January 2022

Power Systems Cascading Failure Analysis And Mitigation Using Discrete Events Systems Approach

Wasseem Al Rousan
Wayne State University

POWER SYSTEMS CASCADING FAILURE ANALYSIS AND MITIGATION USING DISCRETE EVENTS SYSTEMS APPROACH

by

WASSEEM ALROUSAN

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2022

MAJOR: ELECTRICAL ENGINEERING

Approved By:

Advisor Date

DEDICATION

To my parents and my teachers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Power systems cascading failures

A cascading failure is a sequence of events that happens in a power system when a single event or a combination of events causes the propagation of failures. This sequence in some cases stops before it causes a large failure to the system. However, in other cases this sequence can continue and cause a blackout [1]. Many researchers have worked on modeling and mitigation of the problem. Some related studies will be reviewed in the following section.

1.2 Literature review

1.2.1 Cascading failure modeling

Cascading failures have been investigated extensively in the literature. Various models and analysis methods have been proposed to study the problem. The details and complexity of a given model vary based on the context where the model is being used. In [2], the authors reviewed the progress and the methods in the assessment of cascading failures and recommended several directions of future work to enhance the analysis of cascading outages, such as using steady-state analysis and load flow models for more accurate modeling of protection devices. A survey of the methods used in cascading failure risk assessment was also given in [2]. The methods were grouped into two main categories: modeling and simulation methods and bulk analysis methods. Each category was further divided into several techniques. The authors concluded by recommending developing more accurate models to better understand the problem and to capture the interactions between the protection of transmission systems and the power flow dynamics of the actual network. The authors in [3] presented an online quantification of the security level, based on a definition of the system's risk given a forecasted condition, which incorporated the possible loading conditions and the severity of each terminal state. Several severity models were evaluated for the contingency occurring under given operating condition, and one of these models is the severity of cascading overloads. In [4], the authors presented an overview of the measures used in the mitigation and prevention of cascading failures in power systems. In the paper, various methods for mitigation were analyzed and categorized depending on the type of cascading event. Different examples of Remedial Action Scheme (RAS) are introduced by different Transmission System Operators (TSOs). The authors concluded with several recommendations and one of them is to enhance the coordination between different TSOs.

Several approaches have been developed to model the process of failure cascade based on joint dynamics between discrete states and continuous time variables, to extract useful information to increase system reliability. In [5], a simplified model was developed containing the probability of hidden failures of protection devices, based on a DC power model of the network. The authors of [6] suggested a statistical model of cascading failure. Two indices were formulated to describe the probability functions related to initial branch outages and those that occur consequently. This model was then translated into an influence graph, in which nodes and edges reflect these two indices. Critical system components that have a higher probability of initiating cascades were identified. In [7], the authors used hybrid system modeling for the analysis of a cascading outage and applied it to a case study in Europe. The results can be used to tune the protection relays settings to mitigate the failure cascading. A similar dynamic model that takes into consideration of machine dynamics was developed in [8]. Prior to the work in [7] and [8], the authors of [9] presented a framework to analyze and model complex networks at an abstract level, which was achieved by first modeling each component or site in the network as an automaton. The state of each of these automata was determined by transition probabilities within each site and by transition probabilities among neighboring sites or nodes. In [10], the authors introduced a stochastic model that describes the cascading failure process based on Markov chain. The state space of the model was derived by abstracting the network states into a smaller number of states. The transition rates and probability mass function of blackout size were derived. The model can provide information about critical operating conditions that may lead to cascading failure behavior.

Many researchers proposed approaches to mitigate cascading failure, such as the work in [11], where the authors proposed a modular model predicted control (MPC) to control the cascading event. In this approach, the control agents were based on substations, nodes in the power system, where each node has the ability to communicate with its neighboring nodes, and issue optimal control actions not only based on its goals but also the goals of these neighboring nodes. A similar approach based on distributed (MPC) was also used in [12]. In [13], via a graph partitioning approach, the authors presented an online approach to mitigating cascading failures. By employing power transfer distribution factor (PTDF) for generation redispatch and load shedding. The method reduces the stress from overloaded lines. In [14], an algorithm based on multiagents and sensitivities acquired from heuristic data for the power system. This approach requires communication with the central controller and the agents. Control actions were reduced only to generation redispatch without the use of load shedding.

In [15], the authors formulated the process of cascading failure as a Markov model. A cascading failure simulation strategy is derived based on sequential importance sampling benefiting from the derived model.

The authors in [16] proposed an approach to identify cascading failure patterns (CFP).

This approach is based on sequential pattern mining. Fault Chains (FCs) is used to mine CFPs. The influence of CFPs on power systems blackouts is analyzed, and critical CFPs are identified.

In [17], based on chain theory, authors presented a Cascading Faults Graph (CFG). The constructed graph was able to capture the fault propagation mechanism. The developed CFG was used for power system vulnerability assessment.

In [18], based on historical outage data, the authors constructed a Markovian influence graph, which describes the transition probabilities between generations of cascading failure. Each generation represents a single or multiple transmission line outages. The authors used the influence graph to reproduce a cascading failure, to get the probability of the failure, and to extract useful information about the most vulnerable lines that contribute frequently in the process. This information can be used to upgrade the lines to mitigate the problem.

Benchmarking of different models have been introduced in the literature to simulate a cascading failure [19], [20] for continuous variable part, where the authors bench-marked different Quasi Steady State (QSS) models from different aspects such as: degree of stochasticity and power flow model used. In this paper, we used deterministic QSS for the continuous variable part, where AC load flow carried out after each iteration.

The authors in [21] and [22] characterized transmission lines failure propagation using network graph structure. The authors presented their results in two cases: when the post contingency network remains connected and when the failure propagation results in separating the network into multiple islands.

1.2.2 DES Modeling for power systems

Previous research has led to the development of a framework that combines both the continuous dynamics and discrete dynamics of systems. The theory of Discrete Event Systems (DES) and supervisory control developed in [23], [24], [25] and [26] can be used to model such dynamics, where the systems dynamics are abstracted to a higher level, and supervisory control approach can be implemented to achieve control goals or specifications. Moreover, the concept of failure and failure diagnosis in DES was developed by the authors in [27] and [28], which can present the failure modes of the DES.

Several researchers have developed models and control strategies based on DES in power systems applications, such as the work in [29], where the switching nature of Dynamic Flow Controller (DFC) was the matter of interest. Different modes of operation were illustrated in the paper, and the objective was to control line flow within its desired region. The components of DFC were modeled in the DES framework, and a supervisory controller was implemented with the specification reflecting the control objective. The automaton model of the supervisor was presented as a control strategy for the DFC. In [30], the authors introduced the formulation of DES, timed DES (TDES) and their supervisory control methods. For TDES, timing information was introduced for the events was added, and lower and upper bounds were defined. Then, a DES model for Under load Tap Changer (ULTC) was proposed. The control specification was formulated, where automatic and manual modes were incorporated. The authors also modeled the system as TDES, and the supervisor was formulated similarly. The authors of [31] introduced supervisory control of timed networked discrete event systems with communications delay. The derived method was applied on a distribution network with PHEVs as loads, where the loads can communicate with the central node where the controller is located. And with a communication delay. The objective was maintaining the main supply through the transformer bellow its maximum limits.

The problem of modeling a system from the DES perspective is that if the number of system components increases, the number of states increases, which will lead to a quick increase in computational complexity. Modeling the system as a whole and using the conventional centralized will not be efficient for large scale systems. Methods to overcome this issue have been developed in the DES framework. Two main methods for this issue are the matter of interest in this report: on-line and modular control approaches. To overcome the issue of a large number of states in large scale DES, the authors in [32] introduced an on-line limited-look-ahead control approach for supervisory control of DES, where the next control actions are determined for the next N-step based on the projection of the system process. Two attitudes are adopted for the calculations of control actions: conservative and optimistic. As an extension of the previous work, the authors in [33] incorporated the knowledge of the systems states in an attempt to improve the efficiency and quality of computation of the control actions. On-line controls of DES have been used in several domains in the literature in different domains, such as the work in [34], where the authors formulated a Finite State Machine with Variables (FSMwV) and the corresponding supervisory control. For the FSMwV, the traditional definition of DES 5-tuple was modified and extended to be a 7-tuple. Offline and online controllers were formulated using supervisory control theory. Offline supervisors were implemented, and the method introduced was applied to a distribution network with PHEV loads. Based on the developed model and supervisory control, the charging of PHEVs was successfully controlled to accommodate the capacity of the existing network.

Modular control approach can also be used to resolve the issue of a large number of states of DES by dividing control tasks among supervisors. Modular control for DES was first introduced by [35] and [36] and been the studied and developed by many researchers in DES domains. This approach can be efficient when the system is the result of a synchronous product of many components. In [37], the authors presented a modular DES approach to control a high voltage DC transmission system, where two supervisors we devised for two stations at the ends of a transmission line. As an extension to this work in [38], the authors implemented the control approach. However, the approach was limited to a small system consisting of two stations at the end of a transmission line. The authors in [39] proposed a supervisory control algorithm based hybrid automata to control an islanded microgrid with energy storage system and renewable energy resources.

1.3 Motivation and scope of the study

The previous introduction reviewed abstract models for cascading failures, where graph theory and Markov chains were used to derive these models. Although it modeled cascading failure behaviors, remediable actions and on-line control were not employed directly from these models. A unified framework for both modeling and controlling cascading failures on an abstract level has not been presented yet. The work on DES, on the other hand, has not been extended to study cascading failures. The need to implement control strategies that mitigate cascading failures, isolate the faulty components, and restore the system to a normal operating condition must be fulfilled. This dissertation presents a unified DES based framework for modeling and mitigating cascading failures in power systems. A model is first developed to capture the interactions between the protection tripping of a power network and the power flow dynamics at an abstract higher level. After modeling the system, a supervisory control approach is proposed to mitigate the failure cascade. An on-line supervisory control approach is also introduced and implemented to mitigate the problem of cascading failures. Moreover, a Modular Control of DES approach is introduced and proposed to mitigate cascading failures as well. The theoretical framework for the problem and implementation is presented in this dissertation.

1.4 Dissertation organization

This dissertation is organized as follows: Chapter 2 gives an introduction to DES and describes the proposed generic model. Chapter 3 introduces supervisory control theory in DES. Supervisory control approach with forcible events was then proposed to mitigate the failure cascade using a DES approach. To resolve the issue of large number of states, an on-line supervisory control method with a limited lookahead policy approach and forcible events is proposed. A simulation model that combines both the continuous and DES dynamics is developed in Chapter 4. The possible directions for risk assessment of cascading failure are outlined in DES are also introduced. At the end of Chapter 4, results and discussion for the on-line control approach are presented. In Chapter 5, a general architecture for modular supervisory control is introduced to mitigate the risk of cascading failure. Then, the proposed implementation of the modular control approach is introduced. Finally, in Chapter 6, conclusion are drawn, and future work is presented.

CHAPTER 2: DES MODEL FOR POWER SYSTEM CASCADING FAILURE

Cascading failures involve tripping of a large number of power systems components. These tripping events happen due to the automatic operation of protection relays of the power system. To analyze and mitigate failure cascading, a higher abstract model is needed to consider each of the modular components of the system. Furthermore, a mitigation strategy based on this model is also needed. This chapter will briefly introduce fundamentals of power systems operations and cascading failures. Then we will introduce DES and automata, which can be used as a basis for power systems cascading failure modeling and mitigation.

2.1 Power system cascading failure

The basic structure of the power system contains the main components responsible for generating, transmitting, and delivering electricity to consumers. Power systems control is necessary to maintain the power system components at their nominal operating points, such as voltage, current, and frequency. These control devices can be local for the power system components, such as generators. Or, they can be used centrally to control the network and ensure that the network operates economically and in a reliable manner. Finally, the protection systems respond faster than the control devices to faults. Protection systems control the circuit breakers in the power network. Circuit breakers are designed to interrupt high currents and voltages, clear the faults to protect equipment from any damage and to minimize the impact. The main difference between control and protection devices is that control devices usually control continuous-time variables such as voltage and current in the network, while the operation of protection devices usually results in the operation of circuit breakers, thus changing the structure of the power system [40].

Cascading failures that lead to blackouts have been analyzed in the literature. The authors in [41] presented the following sequence for blackouts based on events analysis from several blackouts:

1. The power system is stressed beyond its nominal point; this can result from high loading conditions because of severe weather conditions or unplanned outages.

2. Transmission lines or generators outages due to faults. The power system is designed to withstand such faults, but on top of the operating conditions in point 1, the system will be further stressed.

3. Inappropriate control actions either from control centers as human errors or automatic control actions, which can be the result of incomplete information about the status of the system.

4. Failure cascade. Repetition of points 1,2 and 3 causing additional components outages in the system.

5. Loss of synchronism.

6. Blackout.

The process of cascading can be divided into two phases from a timeline point of view [42]: slow and fast phases. The slow phase can extend from several minutes up to hours, where the components in the system trip usually due to natural disasters or hidden failures in protection devices. The cascade escalates quickly from tens of seconds to milliseconds for the fast phase, where quick lines overload propagation and power system dynamics related to frequency and voltage stability are present. The intervention in this phase is mainly by the remedial or automatic reaction. System operators cannot take any actions in this short time frame. To this end, our approach tends to take control actions related mainly to tripping due to overloads. Power system components are protected from abnormal behavior by protection relays designed to eliminate faults as quickly as possible. Fig. 2.1 shows a system diagram of a node in the power system, where each of the components circuit breakers is connected to certain protection relays.

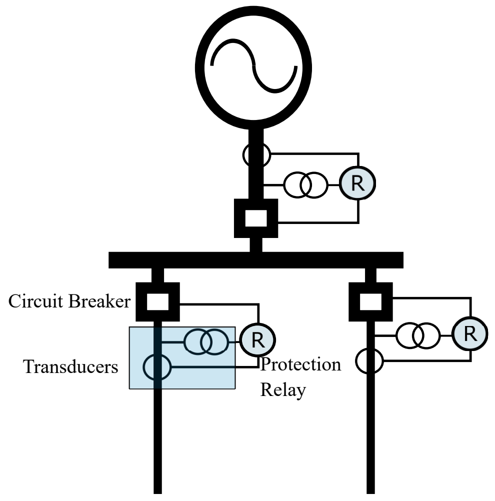

Figure 2.1: System diagram of a simple node in a power system including protection relay.

In [43], the authors provided an overview of the main blackout events in North America and Europe, where the initial events that caused the outage were discussed, and the subsequent events afterward which led to the failure of the power system were also analyzed. The initial event caused the outage of several hundred transmission lines and generators, causing a blackout in the region. The causes of these blackouts were presented, such as the lack of situational awareness and the failure of the control actions taken to prevent the failure cascade. The authors listed the recommendations proposed by the committees that were studying these blackouts. Similarly, in [44], several blackouts events worldwide were analyzed, and the common causes were presented. Transmission lines outages are analyzed in [45], where North America Reliability Council (NERC) reported leading causes of transmission outages during extreme events for two regions as shown in Fig. 2.2.

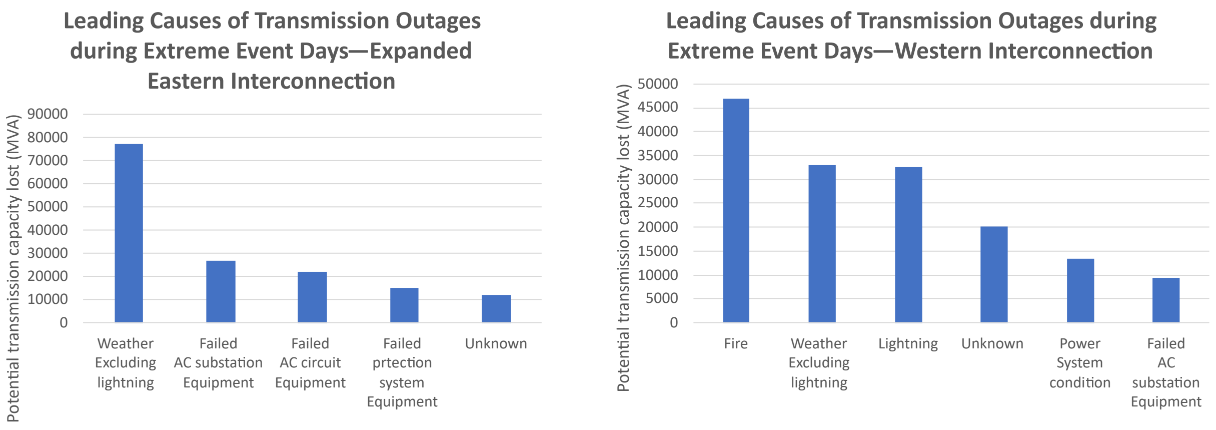

Figure 2.2: Leading causes of transmission outages during extreme events.

For transmission lines, tripping involves mainly over-current and distance protection relays. Undervoltage, over-voltage, and under-frequency relays are also involved for other components. Fig. 2.3 shows inverse time characteristics of an overcurrent relay. From a protection devices point of view, three main concepts are considered to build our model: pick-up time to trip, drop-off time to trip, and the re-closing of a transmission line circuit breaker. The inverse time characteristic equation for overcurrent relays is as follows [46]:

$$\begin{aligned} t(I) &= \frac{A}{M^p - 1} + B \\ M &= \frac{I_{input}}{I_{pickup}} \end{aligned} \tag{2.1}$$

Where $t(I)$ is the operation time in seconds. M is the current expressed in multiplies of pickup current I_{input}/I_{pickup}. A, B and p are constants for selected curve characteristics. The currents flowing in transmission lines that cause them to trip can be derived from the power flow equations. Power flow-based approaches study the steady-state model of the power system. For a network with N buses and G generators, the AC power flow

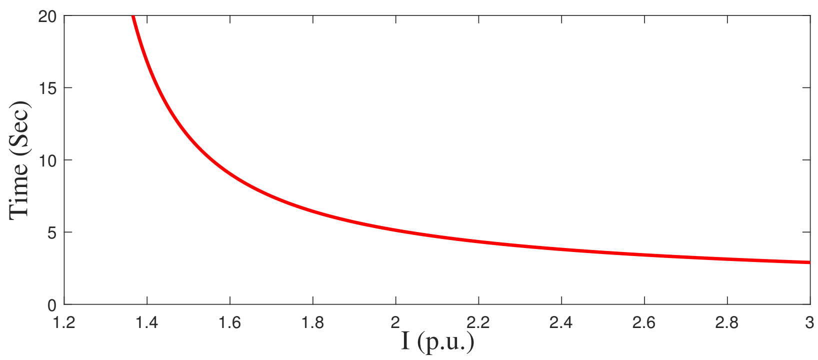

Figure 2.3: Standard inverse time-current characteristic curve of an over-current relay.

equations are as follows [47]:

$$P_{ij} = V_i^2 B_{ij} - V_i V_j (G_{ij} cos(\theta_{ij}) + B_{ij} sin(\theta_{ij})); \forall i, j \in N \tag{2.2}$$

$$Q_{ij} = -V_i^2 G_{ij} - V_i V_j (G_{ij} sin(\theta_{ij}) + B_{ij} cos(\theta_{ij})); \forall i, j \in N \tag{2.3}$$

$$\textstyle\sum_{j \in N} P_{i,j} = Pg_{i|i \in G} - Pd_{i|i \in D}; \forall i, j \in N \tag{2.4}$$

$$\textstyle\sum_{j \in N} Q_{i,j} = Qg_{i|i \in G} - Qd_{i|i \in D}; \forall i, j \in N \tag{2.5}$$

Where equations (2.2) and (2.3) represent transmission line flows of active and reactive powers between bus i and bus j, respectively. P_{gi} and Q_{gi} are generated active and reactive powers at bus i. P_{di} and Q_{di} are active and reactive demands at bus i, respectively. $Y_{ij} = G_{ij} + jB_{ij}$ is the admittance between buses i and j. Eqs. (2.4) and (2.5) are the nodal power balance equations for active and reactive powers, respectively. Linear approximations can be used for the previous set of equations to simplify the calculations, such as DC load flow solution, which will be discussed in later.

Optimal power flow (OPF) is used to find the optimal point of operation of a power system given several constraints. These constraints can be met by enforcing the limits of active and reactive power generations, transmission line capacity, voltage magnitude, and voltage angle difference, as shown in the following equations [48] respectively.

$$Pg_i^{min} \leqslant Pg_i \leqslant Pg_i^{max}; \forall i \in G$$

$$Qg_i^{min} \leqslant Qg_i \leqslant Qg_i^{max}; \forall i \in G$$

$$P_{ij}^2 + Q_{ij}^2 \leqslant (S_{ij}^{max})^2; \forall i, i \in N$$

$$V_i^{min} \leqslant V_i \leqslant V_i^{max}; \forall i \in N$$

$$\theta_{ij}^{min} \leqslant \theta_{ij} \leqslant \theta_{ij}^{max}; \forall i, j \in N$$

In order to analyze and model a cascading failure, several models were presented in section 1.2.1 in the introduction. These models were mainly considering AC and DC load flow-based modeling approaches. Other methods of modeling have been introduced in recent years, such as the work in [8] and [49]. These models consider power systems dynamics analysis. While dynamic models [8], [49] agree with flow-based models at the beginning of the failure events, can also produce power-law distribution provide more insight for power system cascading failure, flow-based models diverge in later stages where machine dynamics govern the sequence of the events. In this work, we are more interested in the beginning of the failure cascade, where our control approaches can interfere more efficiently. In addition to the previous analysis, North American Electric Reliability Corporation (NERC) reviewed and analyzed reliability risks and events in recent years in their report in [45]. Power system protection failure and misoperation can be the cause of cascading failure. The work for wide-area protection and remedial action schemes (RAS) in power systems [50] gained attention from many researchers in power systems, which was also used for cascading failure mitigation. In this dissertation, over-current tripping of transmission lines is considered in the simulation of the cascading failures, considering generation adjustment after each trip. The model used in [51] is considered in this dissertation, where power flow models are used to simulate the process of failure cascade.

In order to study the system on an abstract level, the cascading sequence is analyzed, where tripping of each component is modeled in the form of state transition, along with other transitions that represent major events for that component from the network point of view. The overall interaction between components based on this model is represented for the whole network. The power network is represented based on this component-level model considering load flow solution and stressed transmission lines and other components. After the analysis is carried out from a component level into a network level, the system behavior needs to be altered in case of a high risk of failure cascade. This requires issuing control actions. The computation of control actions will be synthesized from the developed model.

2.2 The DES model

Power system load flow models were discussed in the previous section, where the system's variables like voltages and currents are continuous-state and time-driven systems. As discussed earlier in the literature review, the main automatic control strategies developed for power system dynamics mainly rely on modeling the system in terms of differential and difference equations, where the system's variables are assumed to be continuous-state and time driven. In power system cascading failure, this is not practical. Since in cascading failure, the system's state change with each trip, and natural or external events may change the system topology or one of the system's variables. We can describe this change in the system as a change in the state, and systems that inhibit this behavior can be described as a discrete-state systems. In discrete-state systems, state changes through instantaneous transitions, and these state transitions are also denoted as events. This type of system can also be called an event-driven system. Events that drive state transitions in event-driven systems can happen at various time instances. These events, as mentioned earlier, may be viewed as external causes that affect the system or specific control actions taken by a controller, or they can be viewed as the result of several conditions describing the system that are all met. Events can be described by a discrete set, for example a specific event denoted by σ_1, belongs to a larger set of events Σ, i.e., $\sigma_1 \in \Sigma$.

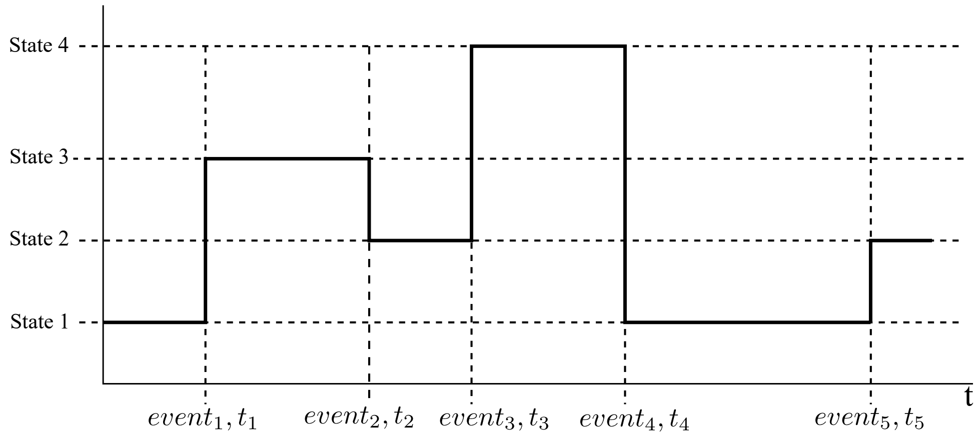

Figure 2.4: Sample path of event based system.

Event-driven systems have gained attention in recent years since they can be combined with a continuous-time system as higher abstract level supervisors, DES-based controllers. Many automated systems that contain microprocessor-based controllers contain both continuous-time feedback control logic associated with event-driven dynamics. Lower level continuous-time controllers and systems being controlled are abstracted on a higher level, where information sent to a DES-based controller in the form of events and control actions are sent back from the supervisor also in the form of events. This is present in many systems, such as power systems. Discrete Event Systems (DES) describe a class of systems that satisfy the following two properties [52]: 1. State space is described by a discrete set. 2. The state transition mechanism is event-driven. These DES properties indicate that the state of the system can change only at discrete points of time, which corresponds to the occurrence of asynchronous events. If sets of events are identified as the cause of state transitions, then time can not be driving the system and may not be an appropriate independent variable. As shown in Fig. 2.4, events occurrences can cause a jump from a state to another, and in some cases, some events do not necessarily cause state transition, resulting in the system staying at its current state. This behavior can not be represented through the difference or differential equation of the general form since no mechanism can be provided to specify how events may interact over time.

Any DES has a set of events associated with it. We denote the set of events as an alphabet Σ. A sequence of events taken from Σ is denoted as a string s. The length of a string is the number of events in it, denoted by $|s|$. A language is defined over the event set Σ as a set of finite-length strings. In the previous content, a language defines the set of all possible trajectories for a DES. Hence it provides a model for the system. Language can be specified using an automaton. A compact structure that defines DES language is needed. This structure also needs to be manipulated and handled by different operations. For continuous-time systems, where the system can be described by differential equations, calculus and algebra are used to perform operations on such systems.

A DES language can be represented using an automaton, the automaton model [23] $\mathcal{A}$ is defined by the following 4 tuple:

$$\mathcal{A} = (Q, \Sigma, \delta, q_o),$$

where Q is the set of states; Σ is the set of events; $\delta : Q \times \Sigma \to Q$ is the (partial) transition function; and q_o is the initial state. We use Σ^* to denote the set of all strings over Σ. The transition function δ can be extended to strings, that is, $\delta : Q \times \Sigma^* \to Q$, as follows. For the empty string ε, $\delta(q, \varepsilon) = q$; for $s \in \Sigma^*$ and $\sigma \in \Sigma$, $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$. We use $\delta(q, s)!$ to denote that $\delta(q, s)$ is defined. The language *generated* by $\mathcal{A}$ is the set of all strings defined in $\mathcal{A}$ from the initial state:

$$L(\mathcal{A}) = \{s \in \Sigma^* : \delta(q_o, s)!\}.$$

In general, a language $K \subseteq \Sigma^*$ is a set of strings. For a string $s \in \Sigma^*$, we use $s' \leq s$ to denote that s' is a prefix of s. The (prefix) closure of K is the set of prefixes of strings in K. A language is (prefix) closed if it equals its prefix closure. By definition, $L(\mathcal{A})$ is closed. We need to consider non-closed language if nonblocking supervisory control is considered. For a string $s \in \Sigma^*$, we use $|s|$ to denote its length. For a set $x \subseteq Q$, we use $|x|$ to denote its cardinality (the number of its elements). In general, a language represent all admissible sequences of events (strings or words) that the DES can generate. After DES language and automaton have been defined, in the next section we will present part of the operations that can be performed on automata.

2.3 Operations on Automata

To analyze a DES modeled by automata, a set of operations are needed to modify its state transition diagram. These operations can either be on a single or a group of automatas.

Accessible part

An accessible part of an automaton is defined as the states in an automaton that are accessible from an initial state. Formally,

$$Ac(\mathcal{A}) = (Q_{ac}, \Sigma, \delta_{ac}, q_0, Q_{m,ac})$$

Where

$Q_{ac} = q \in Q : (\exists s \in \Sigma^*)\delta(q_0, s) = q$ are the states that are accessible from the initial state.

$Q_{m,ac} = Q_{ac} \cap Q_m$

$\delta_{ac} = \delta|_{Q_{ac} \times \Sigma}$ is the restriction of δ to $Q_{ac} \times \Sigma$

Projection

Natural projection or simply projection. The operation is described as mapping from large set of events Σ_l to smaller set of events Σ_s. Denote the operation by θ such that,

$$\theta : \Sigma_l^* \to \Sigma_s^*$$

Formally, the projection is defined as follows:

$$\theta(\varepsilon) = \varepsilon$$

$$\theta(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_s^* \\ \varepsilon & \text{if } \sigma \in \Sigma_l^* - \Sigma_s^* \end{cases}$$

$$\theta(s\sigma) = \theta(s)\theta(\sigma)$$

An inverse projection is defined as follows:

$$\theta^{-1} : \Sigma_s^* \to \Sigma_l^*$$

Parallel composition operation

We define two operations performed on automata: Product and parallel composition, denoted by $||$ and $\times$, respectively. These two operations describe joint behavior between several automata. The standard approach of building models of entire systems from models of individual system components is by parallel composition, which will be used in later sections to represent larger systems out of their components.

Parallel composition between two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ is defined by:

$$\mathcal{A}_1||\mathcal{A}_2 := Ac(Q_1 \times Q_1, \Sigma_1 \cup \Sigma_2, \delta, (q_{01}, q_{02}), Q_{m1} \times Q_{m2})$$

Where

$$\delta((q_1, q_{2)}), \sigma) = \begin{cases} (\delta_1(q_1,\sigma), \delta_2(q_2), \sigma) & \text{if } \sigma \in (\Sigma_1 \cap \Sigma_2) \\ (\delta_1(q_1), q_2) & \text{if } \sigma \in (\Sigma_1 - \Sigma_2) \\ (q_1, \delta_2(q_2)) & \text{if } \sigma \in (\Sigma_2 - \Sigma_1) \\ undefined & otherwise \end{cases}$$

In other words, a transition is possible in a parallel composition if it is possible in one of the automata. The properties of the parallel composition ar:

1. $\mathcal{A}_1 || \mathcal{A}_2 = \mathcal{A}_2 || \mathcal{A}_1$
2. $(\mathcal{A}_1 || \mathcal{A}_2) || \mathcal{A}_3 = \mathcal{A}_1 || (\mathcal{A}_2) || \mathcal{A}_3)$
3. $L(\mathcal{A}_1 || \mathcal{A}_2) = \theta_1^{-1} L(\mathcal{A}_1) \cap \theta_2^{-1} L(\mathcal{A}_2)$
4. $L_m(\mathcal{A}_1 || \mathcal{A}_2) = \theta_1^{-1} L_m(\mathcal{A}_1) \cap \theta_2^{-1} L_m(\mathcal{A}_2)$

Observer Automaton

We can transform a nondeterministic automaton into a language-equivalent deterministic automaton. The state-space of the deterministic automaton is a subset of the power set of the nondeterministic one. This analysis is important in studying the observability of DES, which will be discussed later since unobservable events may result in nondeterministic behavior. The equivalent deterministic automaton is called the observer automaton, denoted by $\mathcal{A}_{obs}$. We assume that only a subset of events are observable. The set of observable events defined by $\Sigma_o \subseteq \Sigma$. Unobservable by $\Sigma_{uo} = \Sigma - \Sigma_o$. The observation is then a natural projection $\theta : \Sigma^* \to \Sigma_o^*$.

Assume a nondeterministic system under partial observation. We may not know the system is in which state by observing observable events. It is possible, however, to find a subset of states the system may be in. This subset set is denoted a state estimate. Formally after observing a string $t \in \theta(L(\mathcal{A}))$. State estimate denoted by

$E(q_o, t) = q \in Q : (\exists s \in \Sigma^*)\theta(s) = t \wedge q = \delta(q_o, s)$

$E(q_o, t) = q \in Q : (\exists q' \in q_o)(\exists s \in \Sigma^*)\theta(s) = t \wedge q = \delta(q_o, s)$

It follows that a nondeterministic automaton is constructed $\mathcal{A}_{nd}$, by replacing unobservable events with an empty string ϵ. $\mathcal{A}_{nd}$ is then converted to an equivalent deterministic automaton, denoted by $\mathcal{A}_{obs}$, or the observer automaton. A state in $\mathcal{A}_{obs}$ is a set of states in $\mathcal{A}_{nd}$.

2.4 Power system Automata model

To analyze power systems on an abstract high level, information regarding the network needs to be constructed, and a model that represents the operational modes of the network is needed. In addition, this abstract higher model needs to be linked through state transitions with a lower continuous-time model. On the other hand, the network model needs to be structured such that it can be decomposed into smaller models that represent the system's components. In other words, the smaller systems models that reflect its components can be coupled together to represent the larger network in a modular fashion [53].

A modular approach is used to obtain the automaton model $\mathcal{A}$: We start with components of a power systems. Each component is modeled by a (small) automaton

$$\mathcal{A}_i = (Q_i, \Sigma_i, \delta_i, q_{o,i}), \ \ i = 1, 2, ..., C.$$

The overall system with C components is then obtained by taking parallel composition (shuffle) of all components [52], that is,

$$\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2 || ... || \mathcal{A}_C.$$

The parallel composition is defined as

$$\begin{aligned}\mathcal{A} &= (Q, \Sigma, \delta, q_o) \\ &= \mathcal{A}_1 || \mathcal{A}_2 || ... || \mathcal{A}_C \\ &= (Q_1 \times Q_2 \times ... \times Q_C, \Sigma_1 \cup \Sigma_2 \cup ... \cup \Sigma_C, \\ &\delta, (q_{o,1}, q_{o,2}, ..., q_{o,C})),\end{aligned}$$

where the transition $\delta : Q \times \Sigma \to Q$ is defined as follows. For $(q_1, q_2, ..., q_C) \in Q$ and $\sigma \in \Sigma$, $\delta((q_1, q_2, ..., q_C), \sigma)$ is defined if and only if $(\forall i \in \{1, 2, ..., C\})\sigma \in \Sigma_i \Rightarrow \delta_i(q_i, \sigma)$ is defined. If it is defined, then $\delta((q_1, q_2, ..., q_C), \sigma) = (q_1', q_2', ..., q_C')$, where

$$q_i' = \begin{cases} \delta_i(q_i, \sigma) & \text{if } \sigma \in \Sigma_i \\ q_i & \text{otherwise} \end{cases}, i = 1, 2, ..., C. \tag{2.6}$$

The power system is modeled based on its main components, with one automaton for each component. The automaton for a transmission line is shown in Fig. 2.1. It has two states, normal (N) and tripped (T), and three events, which are β_1: Line k is tripped, β_2: loading on Line k is changed, and β_3: Line k is back on line. The initial state i denoted by $\rightarrow$. The automata for generation units and loads are similar and also shown in Fig. 2.2.

To model a power system with various components, we take the parallel composition of automata of the components. Taking the IEEE 6 bus system as an example, shown in Fig. 2.6, there are 11 lines, 3 generators and 3 loads. Hence there are total 17 automata. The IEEE 6bus system can modeled by

$$\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2 || ... || \mathcal{A}_{17}$$

$\mathcal{A}$ has 2^{17} states, which is a large number. To overcome this state explosion, we will use two approaches: On-line control that significantly reduces the number of states to

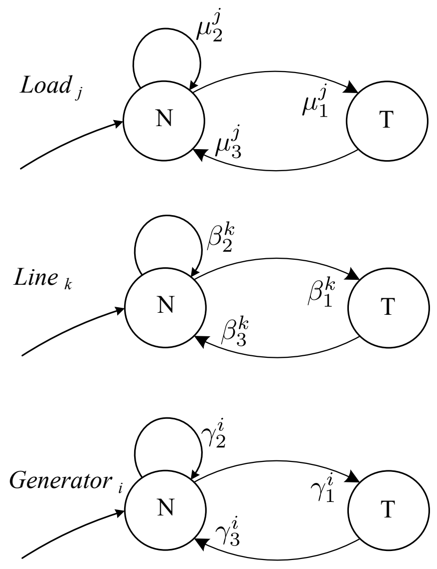

Figure 2.5: Power system components modeled as automata.

be considered, which will be discussed in Chapter 3. And modular control which will be discussed in Chapter 5.

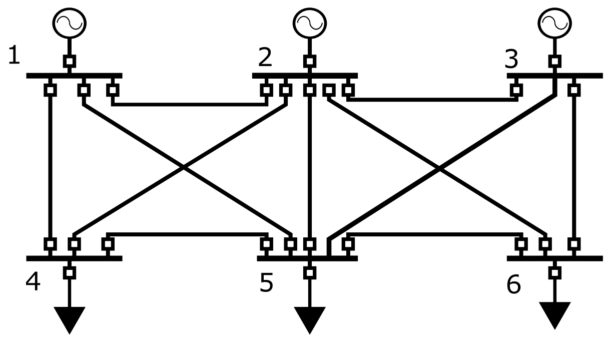

Figure 2.6: Single line diagram of the IEEE-6 Bus system

2.5 Conclusion

In this chapter, we defined power system cascading failure. The failure happens due to sequential tripping of protection relays in power systems. We then described the failure cascade on a higher level from the events occurrences point of view. Events transitions from a state to another can be represented by a DES model, where languages and state spaces are defined in DES framework. The Automata model of DES was then introduced to describe languages in a compact and structural fashion. Operations on automata were then presented to handle and manipulate automata. Finally, a DES model for power systems was introduced that can describe cascading failure.

CHAPTER 3: SUPERVISORY CONTROL OF CASCADING FAILURES

Once a system is modeled in the DES framework, its behavior, denoted by $L(\mathcal{A})$, may be unsatisfactory and reach illegal states. It is desired to control the system via feedback control to achieve certain specifications.

3.1 DES supervisory control

These specifications represents a sublanguage of $L(\mathcal{A})$, denoted by L_a. It is desired to restrict the behavior of the controlled system between the admissible language L_a and the minimum required language L_r, i.e., $L_r \subseteq L_a \subseteq L(\mathcal{A})$. In order to achieve these requirements, a supervisor denoted by $\mathcal{S}$ is required as a feedback controller of the system. The language generated by the supervised system is denoted by $L(S/\mathcal{A})$. Where $Lr \subseteq L(\mathcal{S}/\mathcal{A}) \subseteq L_a$. The admissible language L_a can be generated by an automaton, called specification automaton. Assuming H_a is the specification automaton. then $L(H_a) = L_a$.

The typical definition of events in $L(\mathcal{A})$ is divided into controllable and uncontrollable events, The set of controllable events is denoted by $\Sigma_c \subseteq \Sigma$. The events in $\Sigma_{uc} = \Sigma - \Sigma_c$ are called uncontrollable events. It results that a language can be controllable or not, which will be illustrated in the next section. Assuming that a language K, where $K \in L(\mathcal{A})$, is controllable, then there exist a supervisor $L(S/\mathcal{A}) = \overline{K}$. If K is not controllable, then we would like to find a sublanguage of k that is controllable, which will also be the largest controllable sublanguage. Take the union of all controllable sublanguages in $C_{in}(K)$.

$$K^{\uparrow c} = \bigcup_{L \in C_{in}(K)} L.$$

$K^{\uparrow c}$ is called the supremal controllable sublanguage. If it is desired to define a class of closed and controllable superlanguages of K as $CC_{out}(K))$ L is closed and controllable. We want the smallest closed and controllable superlangauge,

$$K^{\downarrow c} = \bigcap_{L \in CC_{out}(K)} L.$$

$K^{\downarrow c}$ is called the infimal closed and controllable superlanguage. Figure 3.1 illustrate the relation of languages in a supervised DES.

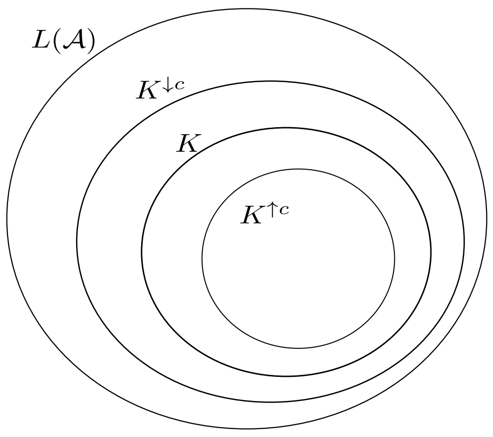

Figure 3.1: DES Languages and sublanguages representation.

As mentioned earlier, $K^{\uparrow c}$ represent the supremal controllable sublangauge, which represent the desired behavior of the supervised system. In order to realize he supervisor in DES, it is represented by an automaton. Supervisor realization will require the information of the plant model $\mathcal{A}$ and the specification automaton H_a. Many researchers developed algorithms and tools to synthesize DES supervisors. such as the work in

3.2 Supervisory Control with Forcible Events

We use a controller, also called supervisor, to control a power system so that certain control objective is achieved. We make the following assumptions on the events. (1) Some events in Σ are forcible; that is, a controller can force them to occur. The set of forcible events is denoted by $\Sigma_f \subseteq \Sigma$. (2) Some events in Σ are controllable in the sense that their occurrences can be disabled. The set of controllable events is denoted by $\Sigma_c \subseteq \Sigma$. The events in $\Sigma_{uc} = \Sigma - \Sigma_c$ are called uncontrollable events. (3) Some events in Σ are observable in the sense that their occurrences can be observed. The set of observable events is denoted by $\Sigma_o \subseteq \Sigma$. The events in $\Sigma_{uo} = \Sigma - \Sigma_o$ are called unobservable events. We further assume that forcible events can pre-empt uncontrollable events. Events such as transmission lines overload are uncontrollable events. Events such as shedding loads are forcible events. We can prevent a transmission line from overload by shedding the system load.

If a string $s \in L(\mathcal{A})$ occurs in $\mathcal{A}$, then the supervisor will observe $\theta(s)$, where $\theta : \Sigma^* \to \Sigma_o^*$ is the (natural) *projection*, defined iteratively as follows. For $s \in \Sigma^*$ and $\sigma \in \Sigma$,

$$\theta(\varepsilon) = \varepsilon,$$

$$\theta(s\sigma) = \begin{cases} \theta(s)\sigma & \text{if } \sigma \in \Sigma_o \\ \theta(s) & \text{otherwise.} \end{cases} .$$

The inverse projection $\theta^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ is given by

$$\theta^{-1}(w) = \{s \in \Sigma^* : \theta(s) = w\}.$$

The projection θ and inverse projection θ^{-1} can be extended from a string to a language. Hence, $\theta(L(\mathcal{A}))$ represents the set of all possible observations.

Formally, a controller is a mapping

$$\mathcal{S} : \theta(L(\mathcal{A})) \to 2^{\Sigma}.$$

$\mathcal{S}$ controls $\mathcal{A}$ as follows: After a string $s \in L(\mathcal{A})$ occurs in $\mathcal{A}$, $\mathcal{S}$ observes $\theta(s)$. Based on its observation, $\mathcal{S}$ issues control $\mathcal{S}(\theta(s))$. Only events in $\mathcal{S}(\theta(s)) \subseteq \Sigma$ are allowed to occur next. The controlled (or closed-loop) system is illustrated in Fig. 2 and denoted by $\mathcal{S}/\mathcal{A}$. The language (set of strings) generated by the controlled system, denoted by $L(\mathcal{S}/\mathcal{A})$, is defined recursively as

$$\begin{aligned} &(1) \quad \varepsilon \in L(\mathcal{S}/\mathcal{A}), \\ &(2) \quad (\forall s \in L(\mathcal{S}/\mathcal{A}))(\forall \sigma \in \Sigma) s\sigma \in L(\mathcal{S}/\mathcal{A}) \\ &\qquad \Leftrightarrow (s\sigma \in L(\mathcal{A}) \wedge \sigma \in \mathcal{S}(\theta(s))). \end{aligned}$$

In other words, if $s \in L(\mathcal{S}/\mathcal{A})$ occurred in the controlled system, then event $\sigma \in \Sigma$ is enabled (allowed occur next) if and only if σ is feasible in $\mathcal{A}$ (that is, $s\sigma \in L(\mathcal{A})$), and σ is enabled by $\mathcal{S}$ (that is, $\sigma \in \mathcal{S}(\theta(s))$).

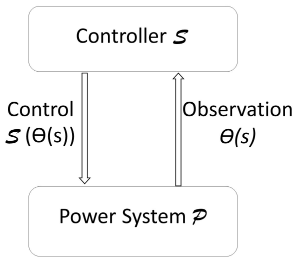

Figure 3.2: A power system controlled by a supervisor.

Since events in Σ_{uc} cannot be disabled (but can be pre-emptied by a forcible event) , for all $s \in L(\mathcal{A})$, $\mathcal{S}(\theta(s))$ must satisfy one of the following two conditions: (1) all events disabled are controllable, that is, $\Gamma(s) - \mathcal{S}(\theta(s)) \subseteq \Sigma_c$, where $\Gamma(s) = \{\sigma \in \Sigma : s\sigma \in L(\mathcal{A})\}$; or (2) some events enabled are forcible, that is, $\mathcal{S}(\theta(s)) \cap \Sigma_f \cap \Gamma(s) \neq \emptyset$. The second condition is due to the assumption that forcible events can pre-empt uncontrollable events. Hence, we require

$$\begin{aligned}&(\forall s \in L(\mathcal{S}/\mathcal{A}))(\Gamma(s) - \mathcal{S}(\theta(s)) \subseteq \Sigma_c)\\ &\vee(\mathcal{S}(\theta(s)) \cap \Sigma_f \cap \Gamma(s) \neq \emptyset).\end{aligned} \tag{3.1}$$

In order to control a system to avoid cascading failures, we need to restrict $L(\mathcal{S}/\mathcal{A})$ to a sublanguage $K \subseteq L(\mathcal{A})$, called a specification language. We assume that, without loss of generality, K is generated by a sub-automaton $\mathcal{H} \sqsubseteq \mathcal{A}$, that is, $K = L(\mathcal{H})$ for some

$$\mathcal{H} = (Q_H, \Sigma, \delta_H, q_o),$$

where $Q_H \subseteq Q$ and $\delta_H = \delta|_{Q_H} \subseteq \delta$ ($\delta|_{Q_H}$ means δ restricted to Q_H). In this way, the state set Q is partitioned into legal/safe state set Q_H and illegal/unsafe state set $Q - Q_H$. A cascading failure can then be modeled as a string of uncontrollable events that lead the system from a legal/safe state to some illegal/unsafe states. To prevent cascading failures is then to design a controller $\mathcal{S}$ such that $L(\mathcal{S}/\mathcal{A}) = K$ or at least $L(\mathcal{S}/\mathcal{A}) \subseteq K$.

To ensure that there exists a controller $\mathcal{S}$ such that $L(\mathcal{S}/\mathcal{A}) = K$ in the case of no forcible events, that is, $\Sigma_f = \emptyset$, the following controllability condition must be satisfied [23].

$$(\forall s \in K)(\forall \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin K) \Rightarrow \sigma \in \Sigma_c.$$

Controllability says that if σ is not allowed after s, then σ must be controllable (so that it can be disabled).

Since we allow forcible events, which are not allowed in [23], the controllability condition needs to be modified as follows.

$$(\forall s \in K)(\forall \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin K)$$
$$\Rightarrow (\sigma \in \Sigma_c \vee (\exists \sigma_f \in \Sigma_f) s\sigma_f \in K).$$

Modified controllability says that if σ is not allowed after s, then either σ is controllable (so that it can be disabled) or there is another forcible event σ_f which is allowed that can preempt σ. We call this modified controllability F-controllability (forcible controllability).

Another condition to ensure that there exists a controller $\mathcal{S}$ such that $L(\mathcal{S}/\mathcal{A}) = K$ is observability [24]. K is observable if

$$(\forall s, s \in K)(\forall \sigma \in \Sigma)$$
$$(\theta(s) = \theta(s) \wedge s\sigma \in K \wedge s'\sigma \in L(\mathcal{A})) \Rightarrow s'\sigma \in K.$$

Observability says that if two string s and s' have the same observation (and hence indistinguishable to the controller), then the fact that σ is allowed after s implies that σ is also allowed after s'. As to be shown in Theorem 1, the existence condition for a controller such that $L(\mathcal{S}/\mathcal{A}) = K$ is that K is F-controllable and observable. When the condition is satisfied, the controller can be designed as follows. After a string $w \in \theta(L(\mathcal{A}))$

is observed by the controller, it calculates the set of all possible states of $\mathcal{H}$ at which the system may be in

$$\begin{aligned} E(w) = \quad & \{q \in Q_H : (\exists s \in L(\mathcal{H})) \\ & w = \theta(s) \wedge \delta_H(q_o, s) = q\}. \end{aligned} \tag{3.2}$$

$E(w)$ is called state estimate after observing w. To calculate $E(w)$, we first replace all unobservable transitions by ε-transitions to obtain $H_\varepsilon = (Q_H, \Sigma_o, \delta_\varepsilon, q_o)$. We then construct the observer as follows [52].

$$H_{obs} = (X, \Sigma_o, \xi, x_o) = Ac(2^{Q_H}, \Sigma_o, \xi, UR(\{q_o\})),$$

where $Ac(.)$ denotes the accessible part; $UR(.)$ is the unobservable reach defined, for $x \subseteq Q_H$, as

$$UR(x) = \{q \in Q_H : (\exists q' \in x)\delta_\varepsilon(q', \epsilon) = q\}.$$

The transition function ξ is defined, for $x \in X$ (note that $x \subseteq Q_H$) and $\sigma \in \Sigma_o$ as

$$\xi(x, \sigma) = UR(\{q \in Q_H : (\exists q' \in x)\delta_\varepsilon(q', \sigma) = q\}).$$

It is well known that $E(w) = \xi(x_o, w)$. A state-estimate-based controller $\mathcal{S}^\diamond$, if exists, is defined as

$$\begin{aligned} \mathcal{S}^\diamond(w) = \quad & \{\sigma \in \Sigma : (\forall q \in E(w)) \\ & \delta(q, \sigma) \in Q \Rightarrow \delta(q, \sigma) \in Q_H\}. \end{aligned} \tag{3.3}$$

The following theorem gives a necessary and sufficient condition for the existence of a controller.

Theorem 1 *Consider a system $\mathcal{A}$ with observable events Σ_o, controllable events Σ_c, and forcible events Σ_f. Given a nonempty and closed specification language $K \subseteq L(\mathcal{A})$, there exists a controller $\mathcal{S} : \theta(L(\mathcal{A})) \to 2^\Sigma$ such that $L(\mathcal{S}/\mathcal{A}) = K$ if and only if K is F-controllable and observable. Furthermore, if such a controller exists, then $\mathcal{S}^\diamond$ defined in Equation (3.3) is such a controller.*

Proof of Theorem 1

(IF) Assume that K is F-controllable and observable. Let us consider the controller $\mathcal{S}^\diamond$ defined in Equation (3.2). We prove $s \in L(\mathcal{S}^\diamond/\mathcal{A}) \Leftrightarrow s \in K$ (that is, $L(\mathcal{S}^\diamond/\mathcal{A}) = K$) by induction on the length of string, $|s|$, as follows.

Base: Since K is nonempty and closed, the empty string $\varepsilon \in K$. By definition, $\varepsilon \in L(\mathcal{S}^\diamond/\mathcal{A})$. Therefore, for $|s| = |\varepsilon| = 0$, we have

$$s \in L(\mathcal{S}^\diamond/\mathcal{A}) \Leftrightarrow s \in K.$$

Induction Hypothesis: Assume that for all $s \in \Sigma^*$, $|s| \leq n$,

$$s \in L(\mathcal{S}^\diamond/\mathcal{A}) \Leftrightarrow s \in K.$$

Induction Step: We show that for all $s \in \Sigma^*$, $\sigma \in \Sigma$, $|s\sigma| = n + 1$,

$$s\sigma \in L(\mathcal{S}^\diamond/\mathcal{A}) \Leftrightarrow s\sigma \in K$$

as follows.

$$\begin{aligned}
& s\sigma \in L(\mathcal{S}^\diamond/\mathcal{A}) \\
\Leftrightarrow\ & s \in L(\mathcal{S}^\diamond/\mathcal{A}) \wedge s\sigma \in L(\mathcal{A}) \wedge \sigma \in \mathcal{S}^\diamond(\theta(s)) \\
& \text{(by the definition of } L(\mathcal{S}^\diamond/\mathcal{A})) \\
\Leftrightarrow\ & s \in K \wedge s\sigma \in L(\mathcal{A}) \wedge \sigma \in \mathcal{S}^\diamond(\theta(s)) \\
& \text{(by Induction Hypothesis)} \\
\Leftrightarrow\ & s \in K \wedge s\sigma \in L(\mathcal{A}) \\
& \wedge(\forall q \in E(\theta(s)))(\delta(q,\sigma) \in Q \Rightarrow \delta(q,\sigma) \in Q_H) \\
& \text{(by the definition of } \mathcal{S}^\diamond\text{, Equation (3.3)).}
\end{aligned}$$

Let us prove

$$\begin{aligned}
& (\forall q \in E(\theta(s)))(\delta(q,\sigma) \in Q \Rightarrow \delta(q,\sigma) \in Q_H) \\
\Leftrightarrow\ & (\forall s' \in \theta^{-1}(\theta(s)) \cap K)(s'\sigma \in L(\mathcal{A}) \Rightarrow s'\sigma \in K)
\end{aligned}$$

by contradiction.

$$\begin{aligned} & \neg(\forall q \in E(\theta(s)))(\delta(q,\sigma) \in Q \Rightarrow \delta(q,\sigma) \in Q_H) \\ \Leftrightarrow\ & (\exists q \in E(\theta(s)))(\delta(q,\sigma) \in Q \wedge \delta(q,\sigma) \notin Q_H). \end{aligned}$$

By the definition of $E(\theta(s))$, Equation (3.2),

$$\begin{aligned} q \in E(\theta(s)) \Leftrightarrow\ & q \in Q_H \wedge (\exists s' \in L(\mathcal{H})) \\ & \theta(s) = \theta(s') \wedge \delta_H(q_o, s') = q. \end{aligned}$$

Hence

$$\begin{aligned} & \neg(\forall q \in E(\theta(s)))(\delta(q,\sigma) \in Q \Rightarrow \delta(q,\sigma) \in Q_H) \\ \Rightarrow\ & (\exists s' \in L(\mathcal{H}))(\theta(s) = \theta(s') \wedge \delta_H(q_o, s') = q \\ & \wedge q \in Q_H \wedge \delta(q,\sigma) \in Q \wedge \delta(q,\sigma) \notin Q_H) \\ \Rightarrow\ & (\exists s' \in K)(\theta(s) = \theta(s') \wedge s'\sigma \in L(\mathcal{A}) \wedge s'\sigma \notin K) \\ \Leftrightarrow\ & \neg(\forall s' \in \theta^{-1}(\theta(s)) \cap K)(s'\sigma \in L(\mathcal{A}) \Rightarrow s'\sigma \in K) \end{aligned}$$

On the other hand,

$$\begin{aligned} & \neg(\forall s' \in \theta^{-1}(\theta(s)) \cap K)(s'\sigma \in L(\mathcal{A}) \Rightarrow s'\sigma \in K) \\ \Leftrightarrow\ & (\exists s' \in K)(\theta(s) = \theta(s') \wedge s'\sigma \in L(\mathcal{A}) \wedge s'\sigma \notin K) \end{aligned}$$

Let $q = \delta_H(q_o, s')$. Then

$$\begin{aligned} & s' \in K \wedge \theta(s) = \theta(s') \Rightarrow q \in E(\theta(s)) \\ & s'\sigma \in L(\mathcal{A}) \Rightarrow \delta(q,\sigma) \in Q \\ & s'\sigma \notin K \Rightarrow \delta(q,\sigma) \notin Q_H. \end{aligned}$$

Hence,

$$\begin{aligned} & \neg(\forall s' \in \theta^{-1}(\theta(s)) \cap K)(s'\sigma \in L(\mathcal{A}) \Rightarrow s'\sigma \in K) \\ \Rightarrow\ & (\exists q \in E(\theta(s)))(\delta(q,\sigma) \in Q \wedge \delta(q,\sigma) \notin Q_H) \\ \Leftrightarrow\ & \neg(\forall q \in E(\theta(s)))(\delta(q,\sigma) \in Q \Rightarrow \delta(q,\sigma) \in Q_H) \end{aligned}$$

Now, we prove

$$\begin{aligned} & s\sigma \in K \\ \Leftrightarrow\ & s \in K \wedge s\sigma \in L(\mathcal{A}) \\ & \wedge(\forall s' \in \theta^{-1}(\theta(s)) \cap K)(s'\sigma \in L(\mathcal{A}) \Rightarrow s'\sigma \in K) \end{aligned}$$

as follows.

($\Leftarrow$): We can prove $\Leftarrow$ by letting $s' = s$.

($\Rightarrow$): We prove $\Rightarrow$ by contradiction. If $\Rightarrow$ is not true, then

$$\begin{aligned} & (\exists s \in \Sigma^*)(\exists \sigma \in \Sigma) s\sigma \in K \wedge \neg(s \in K \wedge s\sigma \in L(\mathcal{A}) \\ & \wedge(\forall s' \in \theta^{-1}(\theta(s)) \cap K)(s'\sigma \in L(\mathcal{A}) \Rightarrow s'\sigma \in K)) \\ \Rightarrow\ & (\exists s \in \Sigma^*)(\exists \sigma \in \Sigma) s\sigma \in K \\ & \wedge\neg((\forall s' \in \theta^{-1}(\theta(s)) \cap K)(s'\sigma \in L(\mathcal{A}) \Rightarrow s'\sigma \in K)) \\ \Rightarrow\ & (\exists s \in \Sigma^*)(\exists \sigma \in \Sigma) s\sigma \in K \wedge (\exists s' \in K) \\ & (\theta(s) = \theta(s') \wedge s'\sigma \in L(\mathcal{A}) \wedge s'\sigma \notin K) \\ \Rightarrow\ & (\exists s, s' \in K)(\exists \sigma \in \Sigma) \\ & \theta(s) = \theta(s') \wedge s\sigma \in K \wedge s'\sigma \in L(\mathcal{A}) \wedge s'\sigma \notin K, \end{aligned}$$

which contradict observability of K. Combining the above derivations, we have

$$s\sigma \in L(\mathcal{S}^\diamond/\mathcal{A}) \Leftrightarrow s\sigma \in K$$

We now show that Equation (3.1) is satisfied as follows.

$$
\begin{aligned}
& (\forall s \in K)(\forall \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin K) \\
& \Rightarrow (\sigma \in \Sigma_c \vee (\exists \sigma_f \in \Sigma_f) s\sigma_f \in K) \\
& \text{(because } K \text{ is controllable)} \\
\Rightarrow \quad & (\forall s \in K)(\forall \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin K) \\
& \Rightarrow (\sigma \in \Sigma_c \vee (\exists \sigma_f \in \Sigma_f) \sigma_f \in \Gamma(s) \wedge s\sigma_f \in K) \\
& \text{(because } s\sigma_f \in L(\mathcal{A})\text{)} \\
\Rightarrow \quad & (\forall s \in L(\mathcal{S}^\diamond/\mathcal{A}))(\forall \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \\
& \wedge s\sigma \notin L(\mathcal{S}^\diamond/\mathcal{A})) \Rightarrow (\sigma \in \Sigma_c \\
& \vee (\exists \sigma_f \in \Sigma_f) \sigma_f \in \Gamma(s) \wedge s\sigma_f \in L(\mathcal{S}^\diamond/\mathcal{A})) \\
& \text{(because } L(\mathcal{S}^\diamond/\mathcal{A}) = K\text{)} \\
\Rightarrow \quad & (\forall s \in L(\mathcal{S}^\diamond/\mathcal{A}))(\forall \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \\
& \wedge \sigma \notin \mathcal{S}^\diamond(\theta(s))) \Rightarrow (\sigma \in \Sigma_c \\
& \vee (\exists \sigma_f \in \Sigma_f) \sigma_f \in \Gamma(s) \wedge \sigma_f \in \mathcal{S}^\diamond(\theta(s))) \\
& \text{(by the definition of } L(\mathcal{S}^\diamond/\mathcal{A})\text{)} \\
\Rightarrow \quad & (\forall s \in L(\mathcal{S}^\diamond/\mathcal{A}))(\forall \sigma \in \Sigma) \\
& (\sigma \in \Gamma(s) - \mathcal{S}^\diamond(\theta(s))) \\
& \Rightarrow (\sigma \in \Sigma_c \vee \mathcal{S}^\diamond(\theta(s)) \cap \Sigma_f \cap \Gamma(s) \neq \emptyset) \\
\Rightarrow \quad & (\forall s \in L(\mathcal{S}^\diamond/\mathcal{A}))(\forall \sigma \in \Sigma) \\
& \neg(\sigma \in \Gamma(s) - \mathcal{S}^\diamond(\theta(s))) \\
& \vee (\sigma \in \Sigma_c \vee \mathcal{S}^\diamond(\theta(s)) \cap \Sigma_f \cap \Gamma(s) \neq \emptyset) \\
\Rightarrow \quad & (\forall s \in L(\mathcal{S}^\diamond/\mathcal{A}))(\forall \sigma \in \Sigma) \\
& (\neg(\sigma \in \Gamma(s) - \mathcal{S}^\diamond(\theta(s))) \vee \sigma \in \Sigma_c) \\
& \vee (\mathcal{S}^\diamond(\theta(s)) \cap \Sigma_f \cap \Gamma(s) \neq \emptyset)
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow\quad & (\forall s \in L(\mathcal{S}^\diamond/\mathcal{A}))(\forall \sigma \in \Sigma) \\
& ((\sigma \in \Gamma(s) - \mathcal{S}^\diamond(\theta(s))) \Rightarrow \sigma \in \Sigma_c) \\
& \vee(\mathcal{S}^\diamond(\theta(s)) \cap \Sigma_f \cap \Gamma(s) \neq \emptyset) \\
\Rightarrow\quad & (\forall s \in L(\mathcal{S}^\diamond/\mathcal{A}))(\Gamma(s) - \mathcal{S}^\diamond(\theta(s)) \subseteq \Sigma_c) \\
& \vee(\mathcal{S}^\diamond(\theta(s)) \cap \Sigma_f \cap \Gamma(s) \neq \emptyset)
\end{aligned}
$$

(ONLY IF) Assume that there exists a controller $\mathcal{S} : \theta(L(\mathcal{A})) \to 2^\Sigma$ such that $L(\mathcal{S}/\mathcal{A}) = K$ and Equation (3.1) is satisfied. Let us first prove that K is F-controllable by contradiction. If K is not F-controllable, then

$$
\begin{aligned}
& (\exists s \in K)(\exists \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin K) \\
& \wedge(\sigma \notin \Sigma_c \wedge (\forall \sigma_f \in \Sigma_f) s\sigma_f \notin K).
\end{aligned}
$$

Let us consider control $\mathcal{S}(\theta(s))$ after the occurrence of $s \in L(\mathcal{S}/\mathcal{A})$.

Case 1 - $\sigma \in \mathcal{S}(\theta(s))$: In this case, $s \in L(\mathcal{S}/\mathcal{A}) \wedge s\sigma \in L(\mathcal{A}) \wedge \sigma \in \mathcal{S}(\theta(s)) \Rightarrow s\sigma \in L(\mathcal{S}/\mathcal{A})$. Therefore, $L(\mathcal{S}/\mathcal{A}) \neq K$, a contradiction.

Case 2 - $\sigma \notin \mathcal{S}(\theta(s))$: In this case, since $\sigma \notin \Sigma_c$ and $s\sigma \in L(\mathcal{A}) \Rightarrow \sigma \in \Gamma(s) - \mathcal{S}(\theta(s))$, we have $\Gamma(s) - \mathcal{S}(\theta(s)) \nsubseteq \Sigma_c$.

On the other hand,

$$
\begin{aligned}
& (\forall \sigma_f \in \Sigma_f) s\sigma \in L(\mathcal{A}) \wedge s\sigma_f \notin K \\
\Rightarrow\quad & (\forall \sigma_f \in \Sigma_f) s\sigma \in L(\mathcal{A}) \wedge s\sigma_f \notin L(\mathcal{S}/\mathcal{A}) \\
& \text{(because } L(\mathcal{S}/\mathcal{A}) = K\text{)} \\
\Rightarrow\quad & (\forall \sigma_f \in \Sigma_f) s\sigma \in L(\mathcal{A}) \wedge \sigma_f \notin \mathcal{S}(\theta(s)) \cap \Gamma(s) \\
& \text{(by the definition of } L(\mathcal{S}/\mathcal{A})\text{)} \\
\Rightarrow\quad & \mathcal{S}(\theta(s)) \cap \Sigma_f \cap \Gamma(s) \neq \emptyset.
\end{aligned}
$$

Therefore, Equation (3.1) is false, a contradiction.

Let us now prove that K is observable by contradiction. If K is not observable, then

$$(\exists s, s \in K)(\exists \sigma \in \Sigma)$$
$$(\theta(s) = \theta(s) \wedge s\sigma \in K \wedge s'\sigma \in L(\mathcal{A})) \wedge s'\sigma \notin K.$$

Let us consider control $\mathcal{S}(\theta(s)) = \mathcal{S}(\theta(s'))$ after the occurrence of $s, s' \in L(\mathcal{S}/\mathcal{A})$.

Case 1 - $\sigma \in \mathcal{S}(\theta(s)) = \mathcal{S}(\theta(s'))$: In this case, $s' \in L(\mathcal{S}/\mathcal{A}) \wedge s'\sigma \in L(\mathcal{A}) \wedge \sigma \in \mathcal{S}(\theta') \Rightarrow s'\sigma \in L(\mathcal{S}/\mathcal{A})$. Therefore, $L(\mathcal{S}/\mathcal{A}) \neq K$, a contradiction.

Case 2 - $\sigma \notin \mathcal{S}(\theta(s)) = \mathcal{S}(\theta(s'))$: In this case, $\sigma \notin \mathcal{S}(\theta(s)) \Rightarrow s\sigma \notin L(\mathcal{S}/\mathcal{A})$. Therefore, $L(\mathcal{S}/\mathcal{A}) \neq K$, a contradiction.

■

If K is not F-controllable and/or not observable, then we want to find a sublanguage $K' \subset K$ such that K' is F-controllable and observable. To see whether such K' is unique or not, we need to investigate whether F-controllable and observable are closed/preserved under arbitrary union. We know that controllability is closed under arbitrary union. The next proposition says that F-controllability is closed under arbitrary union.

Proposition 1 *F-controllability is closed under arbitrary union, that is, if $K_i, i = 1, 2, ...$ are F-controllable, then $\cup_{i=1}^{\infty} K_i$ is also F-controllable.*

Proof of Proposition 1

Assume that $K_i, i = 1, 2, ...$ are F-controllable, that is,

$$(\forall s \in K_i)(\forall \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin K_i)$$
$$\Rightarrow (\sigma \in \Sigma_c \vee (\exists \sigma_f \in \Sigma_f) s\sigma_f \in K_i).$$

We prove

$$(\forall s \in \cup_{i=1}^{\infty} K_i)(\forall \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin \cup_{i=1}^{\infty} K_i)$$
$$\Rightarrow (\sigma \in \Sigma_c \vee (\exists \sigma_f \in \Sigma_f) s\sigma_f \in \cup_{i=1}^{\infty} K_i)$$

by contradiction: If the above is false, then

$$(\exists s \in \cup_{i=1}^{\infty} K_i)(\exists \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin \cup_{i=1}^{\infty} K_i)$$
$$\wedge(\sigma \notin \Sigma_c \wedge (\forall \sigma_f \in \Sigma_f) s\sigma_f \notin \cup_{i=1}^{\infty} K_i)$$

Clearly, $s \in \cup_{i=1}^{\infty} K_i \Rightarrow (\exists K_j) s \in K_j$. For this K_j, we have, $s\sigma \notin \cup_{i=1}^{\infty} K_i \Rightarrow s\sigma \notin K_j$ and $s\sigma_f \notin \cup_{i=1}^{\infty} K_i \Rightarrow s\sigma_f \notin K_j$. Hence,

$$(\exists s \in K_j)(\exists \sigma \in \Sigma)(s\sigma \in L(\mathcal{A}) \wedge s\sigma \notin K_j)$$
$$\wedge(\sigma \notin \Sigma_c \wedge (\forall \sigma_f \in \Sigma_f) s\sigma_f \notin K_j).$$

This contradicts that K_j is F-controllable.

■

By Proposition 1, the supremal F-controllable sublanguage of K exists. We denote it by $K^{\uparrow}$. Similar to that of controllability [23] [52], $K^{\uparrow}$ can be obtained by iteratively removing "bad" states in $\mathcal{H} = (Q_H, \Sigma, \delta_H, q_o)$. A state $q \in Q_H$ is bad if

$$(\exists \sigma \in \Sigma_{uc})\delta(q,\sigma) \in Q \wedge \delta(q,\sigma) \notin Q_H$$
$$\wedge(\forall \sigma_f \in \Sigma_f)\delta(q,\sigma_f) \notin Q_H).$$

Denote the automaton generating $K^{\uparrow}$ as

$$\mathcal{H}^{\uparrow} = (Q_H^{\uparrow}, \Sigma, \delta_H^{\uparrow}, q_o),$$

that is, $K^{\uparrow} = L(\mathcal{H}^{\uparrow})$. Clearly, $Q_H^{\uparrow} \subseteq Q_H$, $\delta_H^{\uparrow} = \delta_H|_{Q_H^{\uparrow}}$, and $K^{\uparrow} \subseteq K$.

It is well known that observability is not closed under arbitrary union. Hence the supremal observable sublanguage of K may not exist. There are works in supervisory control of discrete event systems that investigate properties stronger than observability, such as normality [24] and relative-observability [54], which are closed under arbitrary union. For cascading failures, in this work, we use a simpler approach to control a power system if K is not F-controllable and/or not observable as follows.

Step 1: Find the supremal F-controllable sublanguage $K^{\uparrow}$ of K by calculating

$$\mathcal{H}^{\uparrow} = (Q_H^{\uparrow}, \Sigma, \delta_H^{\uparrow}, q_o).$$

Step 2: Replace all unobservable transitions in $\mathcal{H}^{\uparrow}$ by ε-transitions to obtain

$$H_{\varepsilon}^{\uparrow} = (Q_H^{\uparrow}, \Sigma_o, \delta_{\varepsilon}^{\uparrow}, q_o).$$

Step 3: Construct the observer of $H_{\varepsilon}^{\uparrow}$

$$H_{obs}^{\uparrow} = (X^{\uparrow}, \Sigma_o, \xi^{\uparrow}, x_o).$$

Step 4: Use the following state-estimate-based controller to control the power system. For $w \in \theta(L(\mathcal{A}))$,

$$\begin{aligned} \mathcal{S}^{\uparrow\diamond}(w) = \quad & \{\sigma \in \Sigma : (\forall q \in \xi^{\uparrow}(w)) \\ & \delta(q, \sigma) \in Q \Rightarrow \delta(q, \sigma) \in Q_H^{\uparrow}\}. \end{aligned}$$

The following theorem says that the controller designed above ensures the safety of the system. We can use it to prevent cascading failures.

Theorem 2 *Using the controller designed in the above procedure, $\mathcal{S}^{\uparrow\diamond}$, the language generated by the controlled system is contained in the legal/safe language K, that is, $L(\mathcal{S}^{\uparrow\diamond}/\mathcal{A}) \subseteq K$.*

Proof of Theorem 2

From Step 4 of the procedure, an event $\sigma \in \Sigma$ will be disabled at $\mathcal{S}^{\uparrow\diamond}(w)$ if

$$(\exists q \in \xi^{\uparrow}(w))\delta(q, \sigma) \in Q \wedge \delta(q, \sigma) \notin Q_H^{\uparrow}.$$

Hence the controller disables all event leaving $Q_H^{\uparrow}$. Therefore,

$$L(\mathcal{S}^{\uparrow\diamond}/\mathcal{A}) \subseteq K^{\uparrow} \subseteq K.$$

■

From the above discussions, it is obvious that under full observation, observability is satisfied, and only controllability is needed. The unique maximally permissive supervisor always exists by Proposition 1. While under partial observation, the unique maximally permissive supervisor may not exist. Therefore, the supervisor obtained in Step 4 is a good substitution/approximation of the maximally permissive supervisor.

3.3 On-line Lookahead Control

The control discussed in the previous section can be classified as "off-line" control in the sense that before control is used, we first design the controller $\mathcal{S}$ off-line and then implemented it on-line. Drawbacks of off-line design are as follows. (1) The computation complexity may be too high. As we see in Section 2.2, system $\mathcal{A}$ is usually obtained by taking parallel composition of several components. In terms of the number of states, $|Q| = |Q_1| \times |Q_2| \times ... \times |Q_C|$. Each Q_i is usually small with $|Q_i| = 3 \sim 5$. However, when C is large, $|Q|$ can be very large. Hence, the computation complexity may be too high. (2) The components in a system may change over time. For example, a generator may be turned off when demand is low, or a load may be added to the system. Therefore, $\mathcal{A}_i$, as well as C can change. For off-line control design, when $\mathcal{A}$ changes, the controller needs to be re-designed, which is not convenient.

In this section, we investigate on-line control using lookahead policies. The work in this section can be viewed as extension to the works in [32] [33]. We assume in this section that all events are observable (control under full observation), that is, $\Sigma_o = \Sigma$. On-line control can also be done under partial observation ($\Sigma_o \neq \Sigma$), but the procedure is more complex. Since in most power systems, all events are observable, we restrict the analysis to on-line control under full observation. The on-line control can significantly reduce the computational complexity. This is because on-line control only considers states accessible in the next N steps, as to be shown later in this Section. These states present a small portion of the entire state space that needs to be considered by off-line control. Another benefit of on-line control is that it can also handle flexible and time-varying system configuration, where components in the power system such as generators or loads can be added or removed. This is because on-line control is accomplished by constructing a lookahead tree, which is updated constantly. For on-line control, we do not compute $\mathcal{A}$; rather, we only record the current states q_i of all components $\mathcal{A}_i$. Then the state of $\mathcal{A}$ is given by $q = (q_1, q_2, ..., q_C)$. From the current state $q = (q_1, q_2, ..., q_C)$, we construct a lookahead tree by extending all possible transitions. We do this extension iteratively until the number of levels (the depth of the tree) reaches M, the limit on lookahead steps. Note that we can extend state q to $q' = \delta(q, \sigma)$ without computing $\mathcal{A}$, because $q' = (q'_1, q'_2, ..., q'_C)$ can be calculated element-wise using Equation (2.6).

Formally, a limited lookahead tree is denoted by

$$Tree(q) = (Y, \Sigma, \zeta, y_o)$$

and is defined as follows. Denote the set of all strings defined in $\mathcal{A}$ from state $q \in Q$ as

$$L(\mathcal{A}, q) = \{s \in \Sigma^* : \delta(q, s)!\}.$$

Denote the set of all strings with length less than or equal to M as

$$\Sigma^{\leq M} = \{s \in \Sigma^* : |s| \leq M\}.$$

Then $L(\mathcal{A}, q) \cap \Sigma^{\leq M}$ is the set of all strings defined in $\mathcal{A}$ from state q with length less than or equal to M, Fig. 3.3 shows general structure of $Tree$ with depth M. Clearly, each element (string) in $L(\mathcal{A}, q) \cap \Sigma^{\leq M}$ corresponds to a node in $Tree(q)$. Therefore, we can label the nodes using the corresponding strings. In other words, the set of nodes can be labeled as $Y = L(\mathcal{A}, q) \cap \Sigma^{\leq M}$. The initial node (root) is $y_o = \varepsilon$, the empty string. The transition function is defined as

$$\zeta(y, \sigma) = \begin{cases} y\sigma & \text{if } y\sigma \in Y \\ \text{undefined} & \text{otherwise} \end{cases}.$$

Each node in Y can be mapped into a state in Q. Denote this mapping by $\rho : Y \to Q$, which is given by

$$\rho(y) = \delta(q, y).$$

A node $y \in Y$ is legal/safe if the mapped state $\rho(y)$ is legal/safe, that is, if $\rho(y) \in Q_H$. Hence, the set of legal/safe nodes in $Tree(q)$ is

$$Y_H = \{y \in Y : \rho(y) \in Q_H\}.$$

For the nodes at the Mth level of the tree (leaves at the end of branches), their status of safe/unsafe are more complex: A node may be safe, but a string of uncontrollable events outside the lookahead window may lead to an unsafe node. Since the controller does not see the string yet, it may be too late when controller see it in the future. Thus, the status of safe/unsafe are pending for the nodes at the Mth level. Two attitudes can be used to deal with these pending nodes: conservative and optimistic [32]. We investigate the conservative attitude, since safety is critical to power systems. Formally, the nodes at at the Mth level is denoted by

$$Y_M = \{y \in Y : |y| = M\}.$$

For the conservative attitude, the set of legal/safe nodes in $Tree(q)$ is the nodes in $Y_H - Y_M$ that are accessible from q

$$\begin{aligned} Y_{H-M} &= Ac(Y_H - Y_M) \\ &= \{y \in Y_H - Y_M : (\forall y' \leq y) y' \in Y_H - Y_M\}. \end{aligned} \tag{3.4}$$

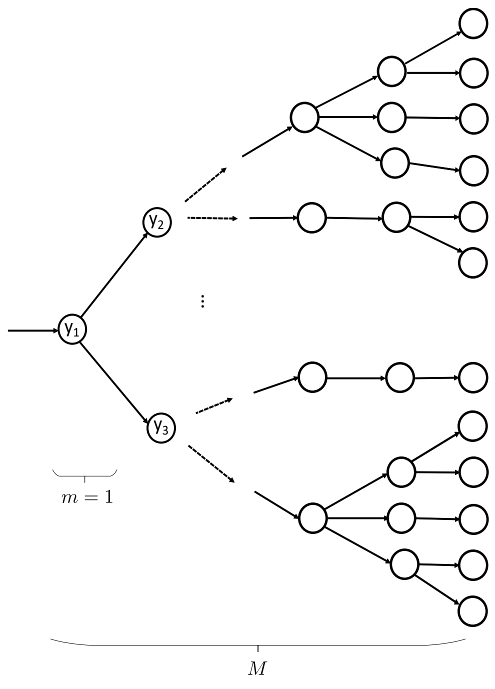

Figure 3.3: Tree structure

To ensure that the system cannot move uncontrollably from nodes in Y_{H-M} to unsafe nodes outside Y_{H-M}, we do a backward search to remove all bad nodes as we do when we calculate the supremal F-controllable sublanguage.

In other words, we iteratively remove nodes in Y_{H-M} that satisfy the following

$$\begin{aligned}&(\exists \sigma \in \Sigma_{uc})\zeta(q,\sigma) \in Y \wedge \zeta(q,\sigma) \notin Y_{H-M}\\ &\wedge(\forall \sigma_f \in \Sigma_f)\zeta(q,\sigma_f) \notin Y_{H-M}).\end{aligned} \tag{3.5}$$

Denote the resulting set $Y^{\uparrow}_{H-M}$. The control at q under the conservative limited lookahead control policy (LLP) with M step lookahead is given by

$$\mathcal{S}^M_{CLL}(q) = \{\sigma \in \Sigma : \zeta(y_o,\sigma) \in Y^{\uparrow}_{H-M}\}.$$

If at the initial state q_o, $\mathcal{S}^M_{CLL}(q_o) = \emptyset$, then we encounter a starting error [32]. This means that either (1) no control exists, that is, $K^{\uparrow} = \emptyset$, or (2) the limit on lookahead steps M is not sufficiently large. For power systems, (1) is unlikely. Hence, we can increase M to avoid a starting error. Note that the choice of the initial state is arbitrary. For example, we can take the current state of a power system as the initial state.

The following theorem says that when there is no starting error, the controller $\mathcal{S}^M_{CLL}$ ensures the safety of the system. In particular, no cascading failures will occur.

Theorem 3 *Assume that at the initial state q_o, $\mathcal{S}^M_{CLL}(q_o) \neq \emptyset$. Then the language generated by the controlled system is contained in the legal/safe language K, that is, $L(\mathcal{S}^M_{CLL}/\mathcal{A}) \subseteq K$.*

Proof of Theorem 3

We prove $L(\mathcal{S}^M_{CLL}/\mathcal{A}) \subseteq K$ by contradiction. Suppose $L(\mathcal{S}^M_{CLL}/\mathcal{A}) \not\subseteq K$. Let $s\sigma$ be the shortest string such that

$$s \in L(\mathcal{S}^M_{CLL}/\mathcal{A}) \cap K \wedge s\sigma \in L(\mathcal{S}^M_{CLL}/\mathcal{A}) \wedge s\sigma \notin K.$$

Let $q = \delta(q_o, s)$ and consider $Tree(q)$ with the root denoted by y_o. We have

$$\begin{aligned}
& s\sigma \in L(\mathcal{S}^M_{CLL}/\mathcal{A}) \wedge s\sigma \notin K \\
\Rightarrow\ & \sigma \in \mathcal{S}^M_{CLL}(s) \wedge s\sigma \notin K \\
& \text{(by the definition of } L(\mathcal{S}^M_{CLL}/\mathcal{A})) \\
\Rightarrow\ & \zeta(y_o, \sigma) \in Y^{\uparrow}_{H-M} \wedge s\sigma \notin K \\
& \text{(by the definition of } \mathcal{S}^M_{CLL}(s) = \mathcal{S}^M_{CLL}(q)) \\
\Rightarrow\ & \zeta(y_o, \sigma) \in Y^{\uparrow}_{H-M} \wedge \zeta(y_o, \sigma) \notin Y_H, \\
& \text{(by the definition of } Y_H)
\end{aligned}$$

a contradiction.

■

In Chapter 4, we use the limited lookahead on-line control to prevent cascading failures in power systems.

3.4 Conclusion

This chapter introduced supervisory control of DES as a framework to control a DES via feedback to achieve specifications and prevent the system from entering illegal states.

Typical supervisory control was then extended to include forcible events since forcible events are used in power systems to achieve control goals and to prevent the system from cascading failures. To overcome the computation complexity that results from large power system, on-line control was then proposed. On-line supervisory control was extended to include forcible events.

CHAPTER 4: DES ON-LINE SUPERVISORY CONTROL IMPLEMENTATION

To implement the on-line control scheme in a power system, the tree $Tree(q) = (Y, \sigma, \zeta, y_o)$ needs to be constructed for the power system. The tree will be updated after each event is observed. Afterward, a backward search is conducted to find possible cascading paths, and to enforce the events (load shedding and generation re-dispatch in this study) if needed. This process will be analyzed in the following two sections. Then, an algorithm is developed to realize this approach. For the DES part, the controller is built in Matlab environment. For this part, the controller operates in three stages: The first stage is to build the lookahead tree, the second stage will search the tree for illegal states and label them based on the most critical lines predicted to trip, and identify the cascading paths if any. The last stage will issue the control actions and inject them back to the power system as forcible events. Fig. 4.1 illustrates a block diagram of the control logic for the system. The operation of the S_{CLL}^{M} block requires translation of the systems C individual components into automata, and real-time knowledge about the system such as the tripped components. For the power system model that simulates the continuous time process of cascading failure, we implemented the algorithm proposed in [51], where the authors introduced an algorithm that simulates the cascading failure process, based on DC power flow. For the first stage of DES controller, expanding the lookahead window with depth M is done based on the automaton model of each of the components. The number of states in the tree is between $1 + C + C^2 + ... + C^M$ and $1 + (2C) + (2C)^2 + ... + (2C)^M$, where C is the number of components in the system according to the modeling approach in Section 2.2. Considering the previous formula for the number of states in the tree, the system will have polynomial complexity of the state space. The individual models of the components $\mathcal{A}_i$ are given in Fig. 2.2. The forcible events are: $\mu_2^k, \gamma_2^k, k = 1, 2, ..., C$. The

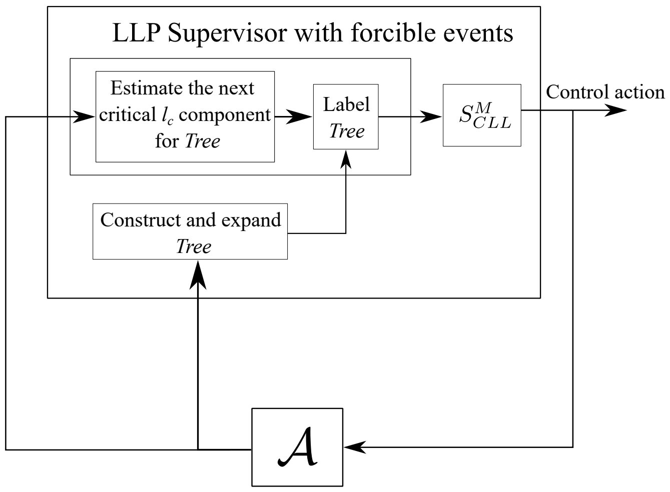
LLP Supervisor with forcible events
Estimate the next critical l_c component for *Tree*
Label *Tree*
S^M_{CLL}
Control action
Construct and expand *Tree*
$\mathcal{A}$

Figure 4.1: Block diagram of on-line controller

following events are uncontrollable: $\beta_1^k, \mu_1^k, \gamma_1^k, k = 1, 2, ..., C$.

After the current system states are observed, a simulation-based risk assessment for the look ahead tree will be conducted. This process label only the paths to a failure with high probabilities, i.e., more likely to happen. We considered two variables to generate random perturbations to the system: initial lines trips, and load demands at the buses. These two initial conditions can be viewed as the new root node y_0, which will determine expansion calculation of the tree with depth M and the most possible paths for the failure cascade in $Tree(q)$. The initial trip will consider $N-k$ contingencies, where $k \in \{1, 2, 3\}$. In this work we will consider the model developed by [55], where the model captures the stochastic process of the cascading failure and predicts the most critical component in the next time interval. The power injection at each load bus $P(t) = [G(t)^T L(t)^T]$ has a mean and covariance. Next, we will formulate the risk index of failure cascade based on DES approach.

4.1 Cascading failure risk definition from DES's perspective

The risk of failure, by determining the probability of a given path that has tripping events (outages) as nodes, is defined as follows [3]:

$$R(X_{t,f}) = \sum_i Pr(E_i)(\sum_j Pr(X_{t,j}|X_{t,f}) \times Sev(E_i, X_{t,j}))$$

Where $X_{t,f}$ is the forecasted condition at time t, $Pr(E_i)$ denotes the ith contingency probability, $Pr(X_{t,j}|X_{t,f})$ is the probability of the jth loading condition given that $X_{t,f}$ forecasted condition happened, and $Sev(E_i, X_{t,j})$ is the severity of the ith contingency occurring under the jth possible operating condition. In this work, we consider the case of cascading overloading events. We denote the severity of failure by the number of tripped transmission line. The risk index definition can be incorporated into the lookahead tree $Tree(q)$ for estimating the cascading failure risk. In the equation above, $Pr(X_{t,j}|X_{t,f}) := Pr(\zeta(y, \beta_2^k))$, and $Sev(E_i, X_{t,j}) := f(V)$. V is the cost function formulated in [33], and is defined $V : X \to \{0, \infty\}$, where ∞ denotes that the state is illegal or the control can not prevent the system from reaching illegal states at this state. We use a similar approach for labeling illegal states. Finally, $Pr(E_i)$ can also defined to be $Pr(E_i) := Pr(\zeta(y_o, \beta_2^k))$, $\beta_2^k \in N - k$, where $N - k$ is contingency list. $Pr(X_{t,j}|X_{t,f})$ assigns the probability for each of the events. The conditional probability of the lth line state at the end of an overload interval can be determined as follows [55]: $p(s_l(t_l^d) = 1) = e^{-\lambda_l^* \tau_l^u(i)}$, where s_l represents the state of the line l, $\tau_l^u(i)$ is the overload interval, and λ_l^* is the transition rate of the line, related to the line l relay settings to trip after an overload event. It follows from the previous model the most critical line at a given stage is defined to be:

$$l_c = arg\,\underset{l}{min}\, a_l \tag{4.1}$$

Where $a_l = \frac{F_l^{Max} - \mu_{Fl}}{\sigma_{Fl}}$. F_l^{Max} , σ_{Fl} and μ_{Fl} are the overloading threshold, the mean of line power flows and covariance respectively. The covariance matrix of the system can be estimated by using Monte Carlo simulations for each contingency applied. We will not consider the probability of the events in implementing LLP control. We will consider however the same probabilistic model to predict the most critical lines to trip in the lookahead tree. The calculation for l_c is essential in applying the on-line control approach, since it is part of calculating the backward search for the illegal states in $Tree(q)$ with depth M.

After the cascading failure risk is estimated by the supervisor, and the tree is labeled and updated accordingly. If the predicted string of events will lead to illegal states, then control actions need to be taken. A new tree with a new layer is generated when a tripping event happens, since the network topology is changed, and the loading is redistributed among the remaining transmission lines. We are only interested in the paths that lead to cascading failures.

4.2 Algorithm for on-line control approach

The implementation of the LLP requires first to translate the power systems components into automata. In [56],we presented a framework that combines a DES analysis tool and MATPOWER [57]. Once the automata are constructed, the DES calculations and procedures are performed in Matlab environment. In this dissertation, however, we implemented the on-line controller directly in Matlab, as shown in Fig. 4.2, with a dedicated function library to expand, label the tree and calculate final control actions. Furthermore, the continuous-time simulation of the cascading failure was implemented as mentioned before based on [57] and [51].

After the critical lines are identified, control (load shedding and generation re-dispatch) actions will be taken, to alleviate the loaded lines, which corresponds to force some events to happen. The goal is to eliminate overloading issues by shedding the minimum amount of loads. Power transfer distribution factor (PTDF) is used to determine the minimum

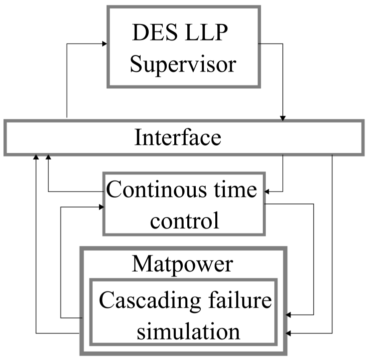

Figure 4.2: MATLAB setup for LLP DES simulation

load shedding after the most critical line for tripping l_c is determined:

$$\begin{aligned} & \min_x \textstyle\sum_j x_j \\ & s.t. \\ & \sqrt{y_t}(A_t^T)^{-1}x \leqslant F_{Max} - P^0 \\ & D \times x \leqslant F_{lMax} - \mu_{F(t)} \end{aligned} \tag{4.2}$$

Where x_j is the amount of injected power at each of the load buses, i.e., x_j corresponds to $\mu_2^j \in P_j$. x is the injected power vector of all of the buses. $D = (PTDF_{i,lc})$ is the value of PTDF of the buses for the critical transmission line l_c.

The DC load flow model can be used to calculate sensitivities to branch flows to changes real power injection at nodes for a given power system. This sensitivity matrix also called power transfer distribution factors (PTDFs), defined $n_l \times n_b$. The PTDF factors are designated and have the following definition:

$$PTDF_{i,j,l} = \frac{\Delta f_l}{\Delta P}$$

where l line index i bus where power is injected j bus where power is taken out Δf : change in megawatt power flow on line when a power transfer of ΔP is made between i and j. ΔP : power transferred from bus i to bus j The PTDF factor represents the sensitivity of the flow on line to power injection on bus i. Suppose one wanted to study the outage of a large generating unit and it was assumed that all the generation lost would be made up by the reference generation. If the generator in question was generating P_i^0 MW and it was lost, we would represent ΔP as $\Delta P = -P_i^0$.

To simplify the AC power flow equation. We omit the Q-V relation. This will result of a linear power flow solution. In the DC power flow, we assume that $|V_i = 1.0$, with 0 angle difference between nodes, and we account for the reactance of the transmission lines and exclude the resistance. These two assumptions come from two observation from typical power systems: 1- the difference in angles of the voltage phasors between any two buses in a power system is less the 10-15 degrees. 2 – the resistance of the transmission lines is significantly less than the lines reactance $r << x$, typically referred to x/r ratio.

The implementation of the overall supervisory control approach is depicted in Fig. 4.1, where each sub block represent a task that was done in Matlab. The first task is to construct and expand the tree with depth M given a root node y_0 and $\zeta(y, \sigma)$. Observed events are received by this block from the power system. This is shown in Algorithm 1.

Algorithm 1 Expand $Tree(q)$

```
*Input:* current state($y_o$)
*Output:* $Tree(y_o)$
  Expand $\zeta(y_o, \sigma)$ with $\rho : Y \to Q$
```

After the tree is expanded, illegal states are labeled based on the calculations for l_c. This is defined by the states reached by the transition function $y_T = \zeta(y, \beta_2^c)$, $y_T \notin Y_{H-M}$. This function is evaluated by the Label Tree block in Fig. 4.1, which also receives an estimate for the most critical line to trip. The implementation of this process is shown in Algorithm 2, where $Tree$ states are checked for each step m with depth M. The last step is to calculate the control actions and compute the $S_{CLL}^M(q)$, shown in Algorithm 3. The results of the above analysis are illustrated in Section 4.3.

Algorithm 2 Label $Tree(q)$

```
Input: $Tree(y_o)$
Output: Labeled $Tree(y_o)$
 1: for m < M do
 2:     if Line is overloaded then
 3:         Calculate $l_c$ using (4.1)
 4:         Label illegal states $y_T$
 5:     end if
 6: end for
```

Algorithm 3 Calculate $S^M_{CLL}(q)$

```
Input: Labeled $Tree(y_o)$
Output: Control Actions
 1: for m < M  do
 2:     if $y_T \notin Y_{H-M}$ then
 3:         Calculate minimum load shedding using (4.2)
 4:         Apply load shedding Actions for selected loads and generators to force events $\mu^j_2$
            and $\gamma^i_2$
 5:         Check if the control actions do not lead to other overloads
 6:         Trim $Tree(y_o)$
 7:     else
 8:         No control action is needed
 9:     end if
10: end for
```

4.3 Simulation results of on-line control

The framework described in Chapters 3 and 4 has been implemented for the IEEE 6 and 30 bus systems. We first illustrate the approach with a simple example of the $Tree(q)$ for the IEEE 6 bus system [58] shown in Fig. 4.3, as an event starts a string of failure cascade. Given a root node y_o that represents a normal operational mode for all the components, the tree is expanded for the next two steps, based on all of the possible transitions.

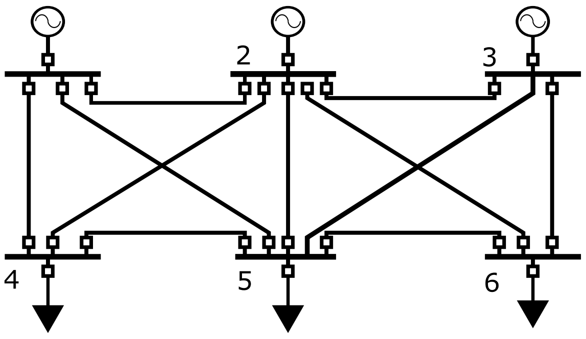

Figure 4.3: Single line diagram of the IEEE-6 Bus system

The number of states for the first window ($m = 1$) is 34, as shown in Fig. 4.4. Assume an $N-2$ line contingency is observed: lines 3 and 7 are tripped. Events β_1^3 and β_1^7 will lead to state number y_{35}. The tree is further expanded and lines are checked for overloading. The most critical line l_c can then be determined. In this case, line 6 is identified as critical line and its tripping state after event β_1^6, is labeled as an illegal state, which will further lead to lines 10 and 4 to trip.

To prevent this situation, control actions are taken at state y_{35} to prevent the system from having cascading failure by enforcing the event μ_2^2 (shedding load 2), γ_2^1 and γ_2^2 (dispatch of generators 1 and 2). These actions depend on the loading conditions For light loading conditions, control actions on load 2 and generator 2 will be sufficient, as event sequence $\mu_2^2\gamma_2^2$ will stop the failure cascade and lead to a safe state. Similarly,

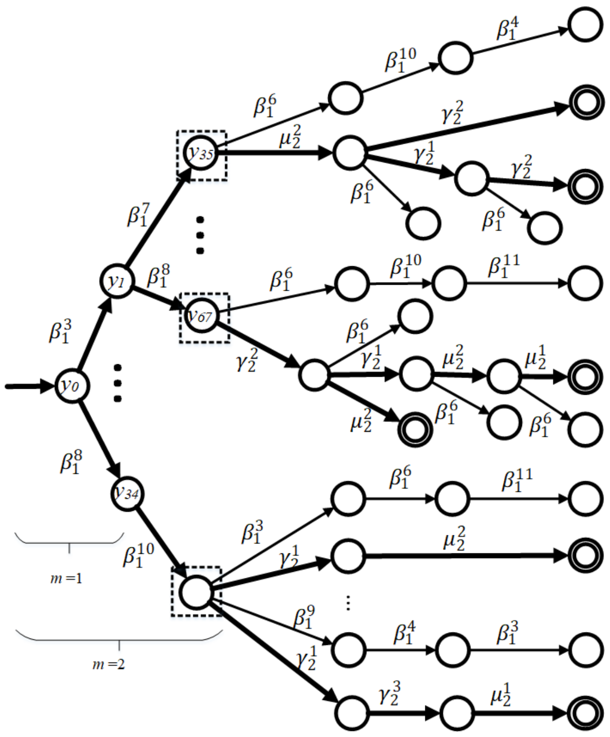

Figure 4.4: LLP illustrative example for IEEE 6 Bus system

other possible trajectories of the systems are also illustrated in the tree. All possible trajectories under control is described by the language $L(\mathcal{S}/\mathcal{A})$. For each of the load buses in each of the $N-k$ cases, where $k \in \{1,2,3\}$. We sampled 10 random values with normal distribution.

Fig. 4.5 illustrates our approach applied to $N-2$ contingencies for the IEEE 6-Bus system, showing the MW lost before and after applying control compared with a centralized emergency control method that was proposed in [59]. Only the cases that lead to cascading failure are included based on an off line simulation for the cases study under the same loading conditions. Fig. 4.6 shows the same results but without MW lost.

Simulation studies have also been carried out on the IEEE 30-bus and 118-bus systems [60], single-line diagrams shown in Fig. 4.7 and 4.12 respectively, to verify the findings of the proposed control. The IEEE 30-bus system contains 41 transmission lines,

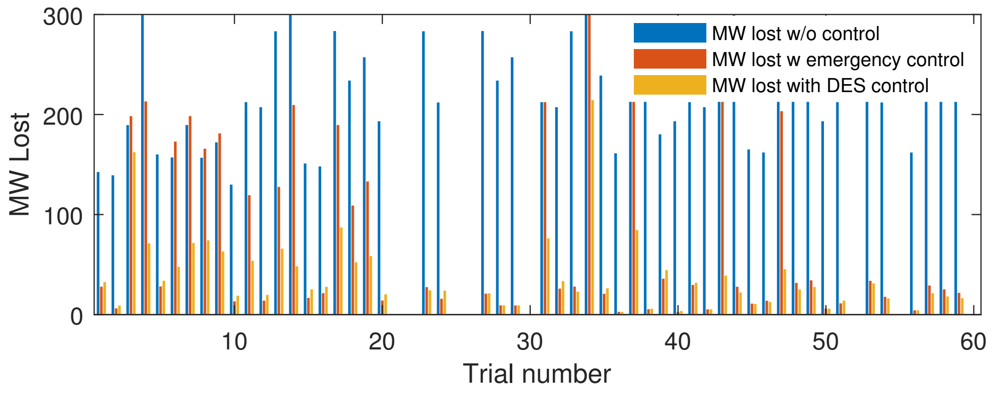

Figure 4.5: MW lost before and after applying control for the IEEE 6-bus system.

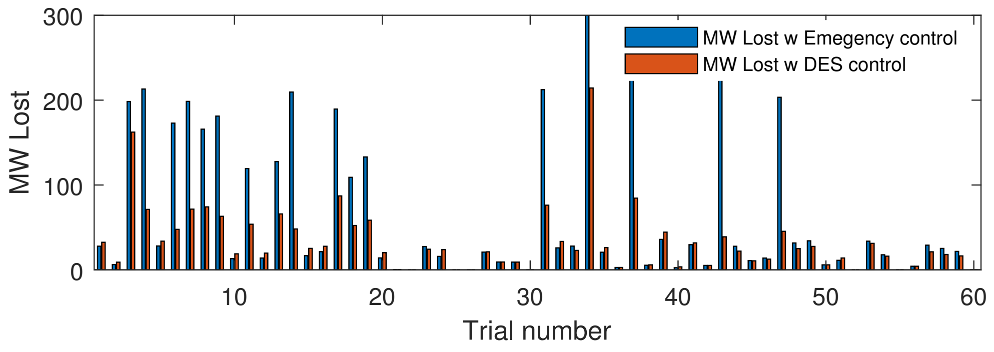

Figure 4.6: MW lost after applying control using DES and emergency control methods for the IEEE 6-bus system.

22 load buses, and six generators. The generation capacity is 186 MW. The IEEE 118-bus system simulated in this dissertation contains 186 transmission lines, 64 load buses, 35 synchronous condensers, and 19 generators. The total generation capacity is 4,377 MW. Fig. 4.8 shows the same comparison as Fig. 4.6 between the proposed control and the emergency control method proposed in the literature, but for four $N-2$ contingencies for the IEEE 30-bus system. For the 30 bus system, the $N-2$ contingencies are: $\{(11,12),(11,19),(11,40),(14,40)\}$, each pair represents two transmission lines numbers which have been tripped as an initial contingency. For each pair, as shown in Fig. 4.8, ten simulations have been carried out. Similarly in Fig. 4.13 and Fig. 4.14 for the 118 bus system, the $N-2$ contingencies are $\{(8,51),(8,96),(33,38)\}$. The uncertainties in

the simulations results shown in Fig. 4.5, 4.6, 4.8, 4.13, and 4.14 are embedded in two parameters: the selected $N-2$ contingencies and the loading conditions of the loads. The set of contingencies is part of a set of consecutive transmission line trippings after that contingency is applied, which may lead to a blackout of the system. The second uncertainty parameter, the loads, was changed based on a normal distribution as mentioned earlier.

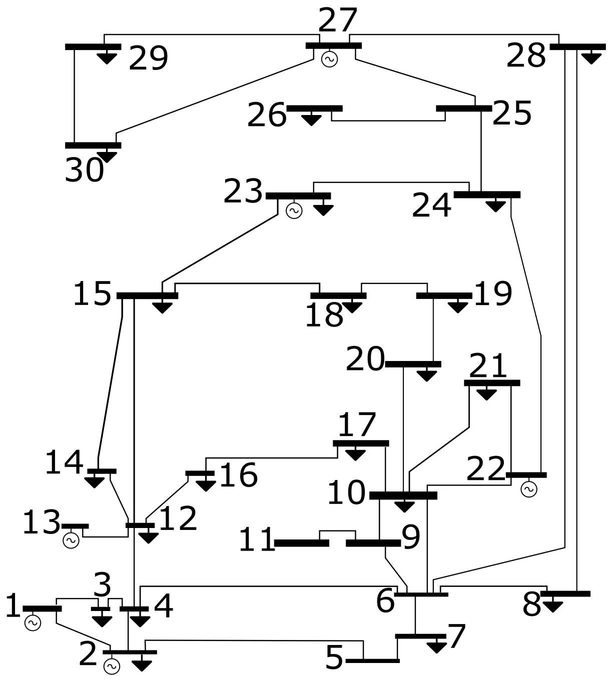

Figure 4.7: Single line diagram of the IEEE 30-bus system.

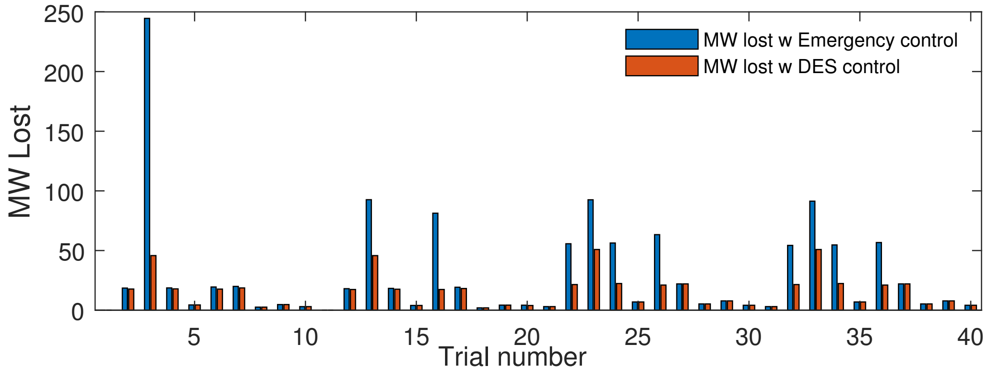

Figure 4.8: MW lost after applying control using DES and emergency control methods for the IEEE 30-bus system.

Fig. 4.9, 4.10 and 4.11 show three specific cases of the IEEE 6 and 30-bus systems under different $N-2$ contingencies. The figures shows the effectiveness of our approach, where the cumulative line tripping was stopped at the original $N-2$ lines (i.e., lines 3 and 7 in the 6 bus system and lines 11 and 30, and lines 11 and 12 in the 30-bus system) that were tripped as an initial trip. The control approach that was proposed in this work can be extended to include the restoration of the initially tripped lines, but this will be left for future work.

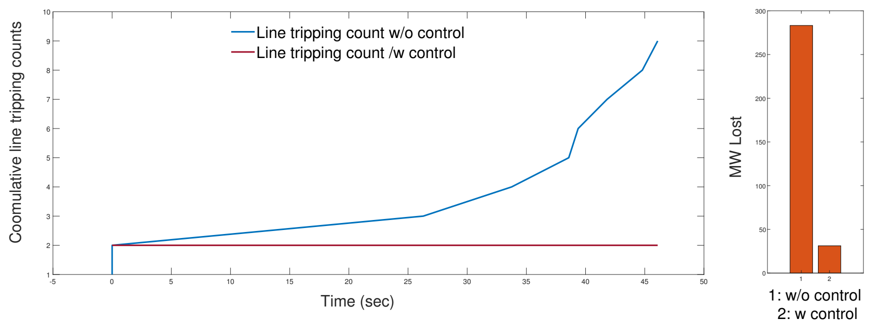

Figure 4.9: Effect of applying forcible events to IEEE 6-Bus system, N-2 contingency is lines 8 and 10 tripped.

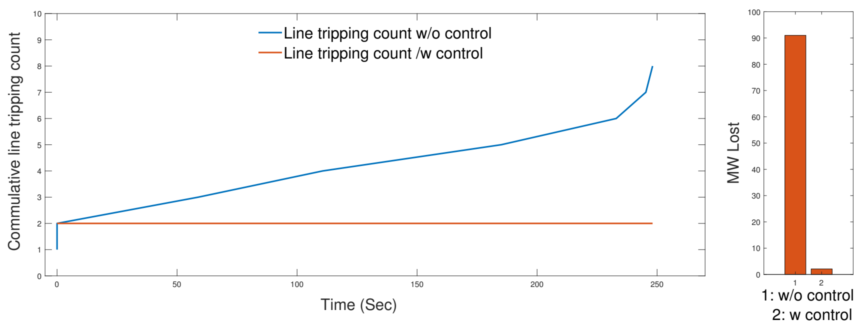

Figure 4.10: Effect of applying forcible events to IEEE 30-Bus system, N-2 contingency is lines 11 and 30 tripped.

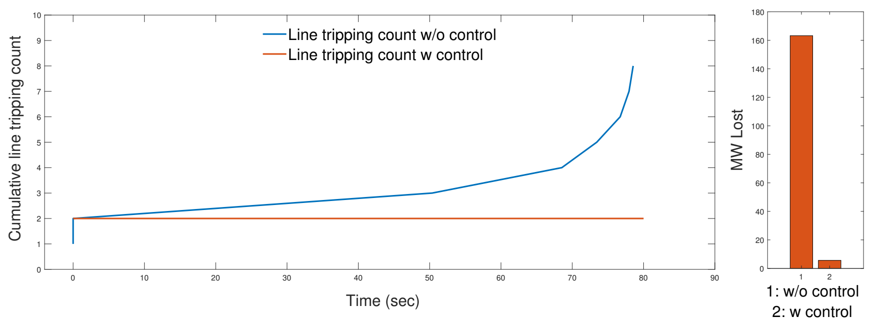

Figure 4.11: Effect of applying forcible events to IEEE 30-Bus system, N-2 contingency is lines 11 and 12 tripped.

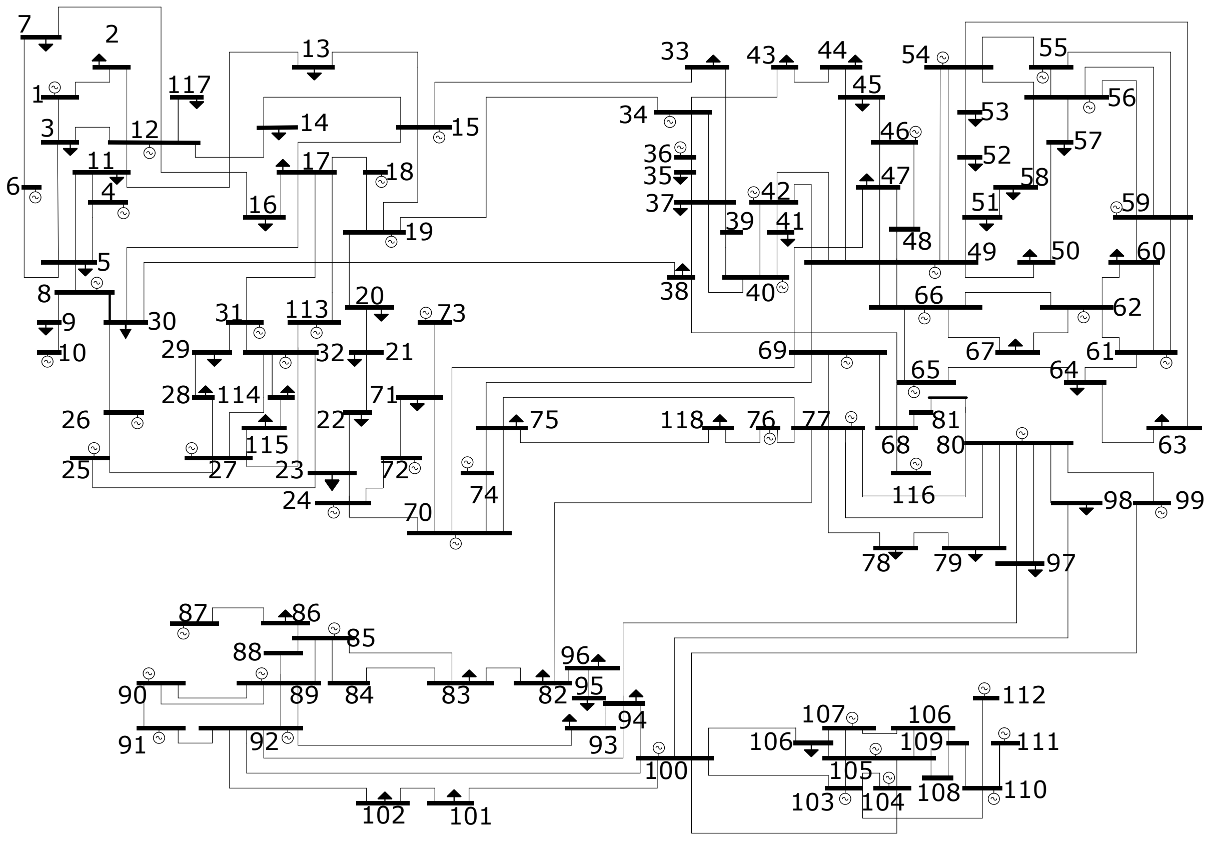

Figure 4.12: single line diagram of 118-Bus system.

To further verify the effectiveness of our approach, we performed 1000 Monte Carlo simulations by applying uncertainty on both the initial $N-2$ tripping and the amount of the loads for the IEEE 118-bus system. The $N-2$ initial trips followed uniform distribution, and the loads were following a normal distribution. The result is shown in Fig. 4.15. Our approach gives better overall performance by preventing the failure

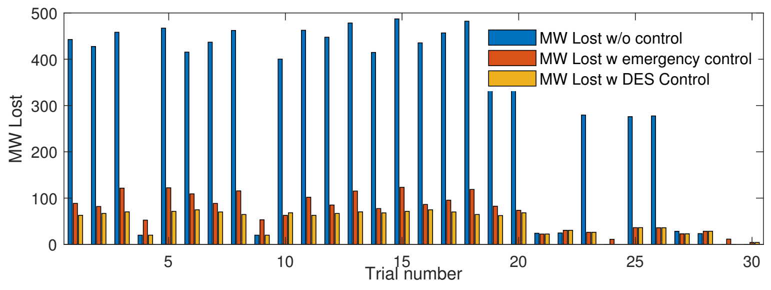

Figure 4.13: MW lost after applying control using DES and emergency control methods for the IEEE 118-bus system.

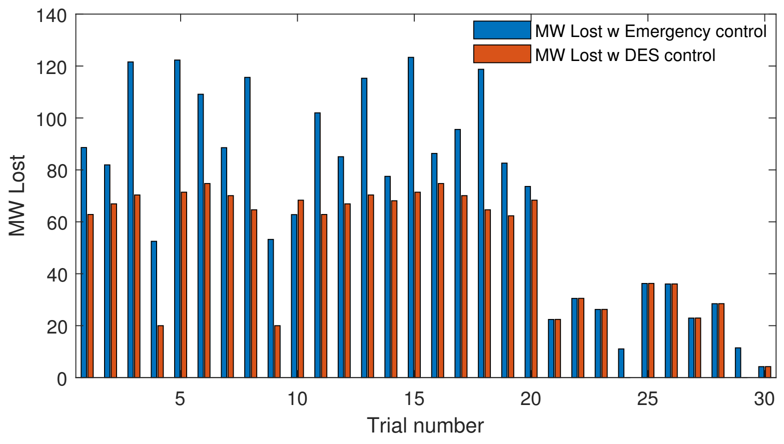

Figure 4.14: MW lost after applying control using DES and emergency control methods for the IEEE 118-bus system.

cascade at lower MW lost.

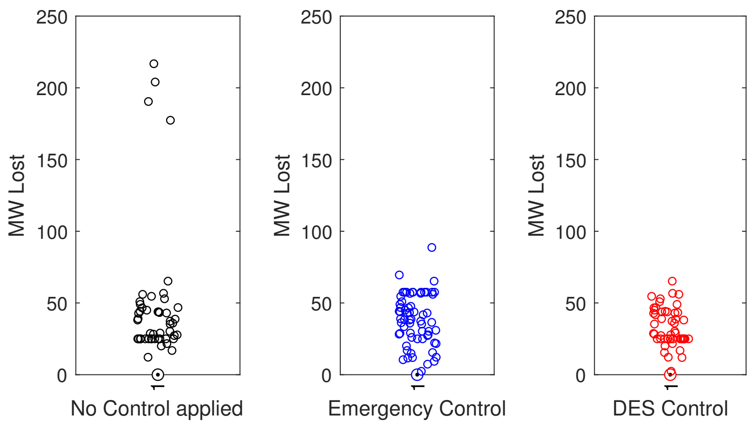

Figure 4.15: MW lost after applying control using DES and emergency control methods for the IEEE 118-bus system.

4.4 Conclusion

This chapter introduced an implementation framework of the formulation in Chapter 3 of on-line control. A Matlab based implementation of on-line control to mitigate cascading failure was proposed based on two levels: The limited lookahead tree calculations. And the continuous-time lower level of the forcible control actions, which represent the values of each of the control actions. The proposed method was then tested on three case studies: The IEEE 6-bus, 30-bus, and 118-bus systems. The results showed that the proposed approach shows that our approach was able to mitigate the failure cascade and gives a better overall performance of other methods used in the literature.

CHAPTER 5: MODULAR SUPERVISORY CONTROL OF DISCRETE EVENT SYSTEMS

One of the problems of modeling a power system via a tree structure is that the complexity of the tree computation increases as the system gets larger. In addition, on-line control (discussed in Chapter 4) assumes that the control is central, which requires that each of the nodes in the system communicate information to the central controller. This control architecture is not reliable as it has a single point of failure. Methods have been developed in the DES framework to deal with this issue. The power system can be presented in DES via a modular approach, where each node can be modeled as a combination of the power system main components attached to this node.

5.1 Introduction to modular supervisory control

Modular supervisory control is more realistic and reliable for large power systems. The supervisory control given in Section 3.2 can be divided into smaller tasks and formulated as a modular supervisory control (MSC) in DES. In this section, we give a brief introduction to MSC for DES, and propose a MSC approach for power systems cascading failures.

As discussed in Section 3.2, $\mathcal{A}$ is a synchronous product of independent automata over disjoint alphabets, $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2||...||\mathcal{A}_n$, where n is the number of modules in the power system, such that $L_m(\mathcal{A}) = L_m(\mathcal{A}_1)||...||L_m(\mathcal{A}_n)$ and $L(\mathcal{A}) = L(\mathcal{A}_1)||...||L(\mathcal{A}_n)$. It follows that by dividing the controller that each sub controller (e.g., for site i) is given by

$$L(\mathcal{S}_i/\mathcal{A}) = L_{ai}^{\uparrow c}$$

Where L_{ai} is the admissible language of site i. Formally, the modular supervisor is given by $\mathcal{S}_i$, $\mathcal{S}_{mod} := \mathcal{S}_1(s) \cap ... \cap \mathcal{S}_n(s)$. Then the language generated by the controlled system is given by

$$L(\mathcal{S}_{mod}/\mathcal{A}) = L_{a1}^{\uparrow c} \cap ... \cap L_{an}^{\uparrow c} \tag{5.1}$$

The above case is extended to case where several projections of $\mathcal{A}$ are controlled concurrently. Assume $\mathcal{A}$ is defined over Σ, and several subalphabets $\Sigma_i \subseteq \Sigma$,$i = 1, ..., n$, are given, describing local events. Let $\mathcal{A}_i$ be an automaton such $L_m(\mathcal{A}_i) = \theta_i L_m(\mathcal{A})$ and $L(\mathcal{A}_i) = \theta_i L(\mathcal{A})$, where $\theta_i : \Sigma^* \to \Sigma_i^*$ are the natural projections. Assume as usual $L_m(\mathcal{A}) = L(\mathcal{A})$. It then follows $L_m(\mathcal{A}_i) = L(\mathcal{A}_i)$. A local control structure is assigned to $\mathcal{A}_i$ as before: $\Sigma_{ic} := \Sigma_i \cap \Sigma_c, \Sigma_{iuc} = \Sigma_i \cap \Sigma_{uc}$. The controllable sublanguage $L_{ai}^{\uparrow c}$ is defined for each $\mathcal{A}_i$ in the same way. Fig. 5.1 shows a block diagram of n supervisors $\mathcal{S}_i, i \in \{1, ..., n\}$ with the projection θ_i over the language of $\mathcal{A}$ for each supervisor.

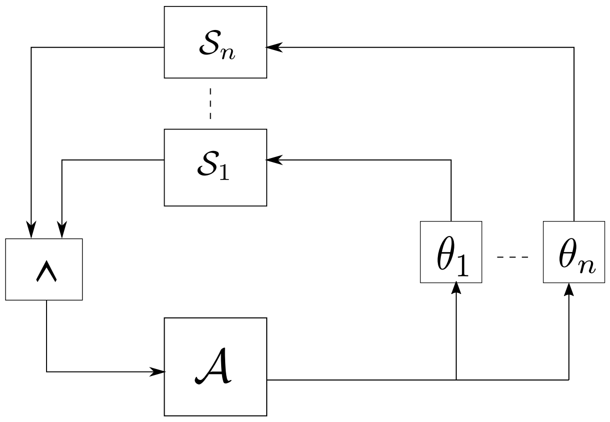

Figure 5.1: Modular approach

Given the local specification languages $L_{ai} \subseteq L_m(\mathcal{A}_i)$, for which we compute $K_i = L_{ai}^{\uparrow c}$. The ith local supervisor (synthesizing K_i) induces global closed controllable behavior $L(\mathcal{S}_i/\mathcal{A}) = \theta_i^{-1}(K_i) \cap L(\mathcal{A})$. The concurrent action of all local supervisors is then the global closed controllable behavior $L(\mathcal{S}/\mathcal{A}) = \bigcap\limits_{i=1:n} L(\mathcal{S}_i/\mathcal{A})$. Now define

$$L_a := \bigcap_{i=1:n} \theta_i^{-1}(L_{ai}) \cap L_m(\mathcal{A})$$
$$K := \bigcap_{i=1:n} \theta_i^{-1}(K_i) \cap L_m(\mathcal{A})$$

As suggested before, $\mathcal{A}$ will often be given as a synchronous product of independent components, say $\mathcal{A}_{j'}(j' \in J)$, where $J = \bigcup_{i=1:n} j_i$. Note that $\mathcal{A}_{j'}$ in this definition can be the union of several sublanguages in J, i.e., $\Sigma_{j'} = \bigcup_{i=1:k} \Sigma_i$, where $i = \{1, ..., k, ..., n\}$. In the typical case considered above, each $\Sigma_{j'}$ will be precisely a disjoint union of form $\Sigma_{j'} = \bigcup_{i \in j'} \Sigma_i$, for some subset $j' \subseteq J$. Thus $\mathcal{A}_i$ can be taken as the synchronous product of the $\mathcal{A}_{j'}$ over $j' \subseteq J$. Then, we can express $\mathcal{A} = \mathcal{A}_{j'} || \mathcal{A}_{j''}$, where $\mathcal{A}_{j''}$ is defined over $\Sigma - \Sigma_{j'} = \bigcup_{j \in J - j'} \Sigma_j$. It follows that,

$$\tilde{L_{aj'}} := \theta_{j'}^{-1}(L_{aj'}) \cap L_m(\mathcal{A}) = L_{aj'} || L_m(\mathcal{A}_{j''})$$

$$\tilde{K_{j'}} := \theta_{j'}^{-1}(K_{j'}) \cap L_m(\mathcal{A}) = K_{j'} || L_m(\mathcal{A}_{j''})$$

In summary, under the above mentioned conditions, the concurrent optimal local control for each group provides global control that is also optimal. In the literature, controllability was discussed from a typical point of view. Forcible events are not introduced, which are vital for the mitigation of cascading failure since the nature of control actions is to preempt tripping events by means of forcible events (e.g., load shedding and generation redispatch). The formulation of modular control with forcible events is a subject of study in this dissertation work. Also, continuous time optimization, prediction model, and the DES modular control approach discussed above need to be combined in one framework, which will be implemented in the dissertation. We model each bus i with all connected components like generators, loads and transmission lines as the results of the shuffle of these components. Taking bus 1 as an example in the power system in Fig. 5.2, one generator and two transmission lines are connected to that bus. It follows then from our previous definition $\mathcal{A}_1 = Geneartor_1 || Line_1 || Line_2$. The result of this shuffle is shown in Fig. 5.2, where the definitions of the events and states for each component is the same as in Fig. 2.5.

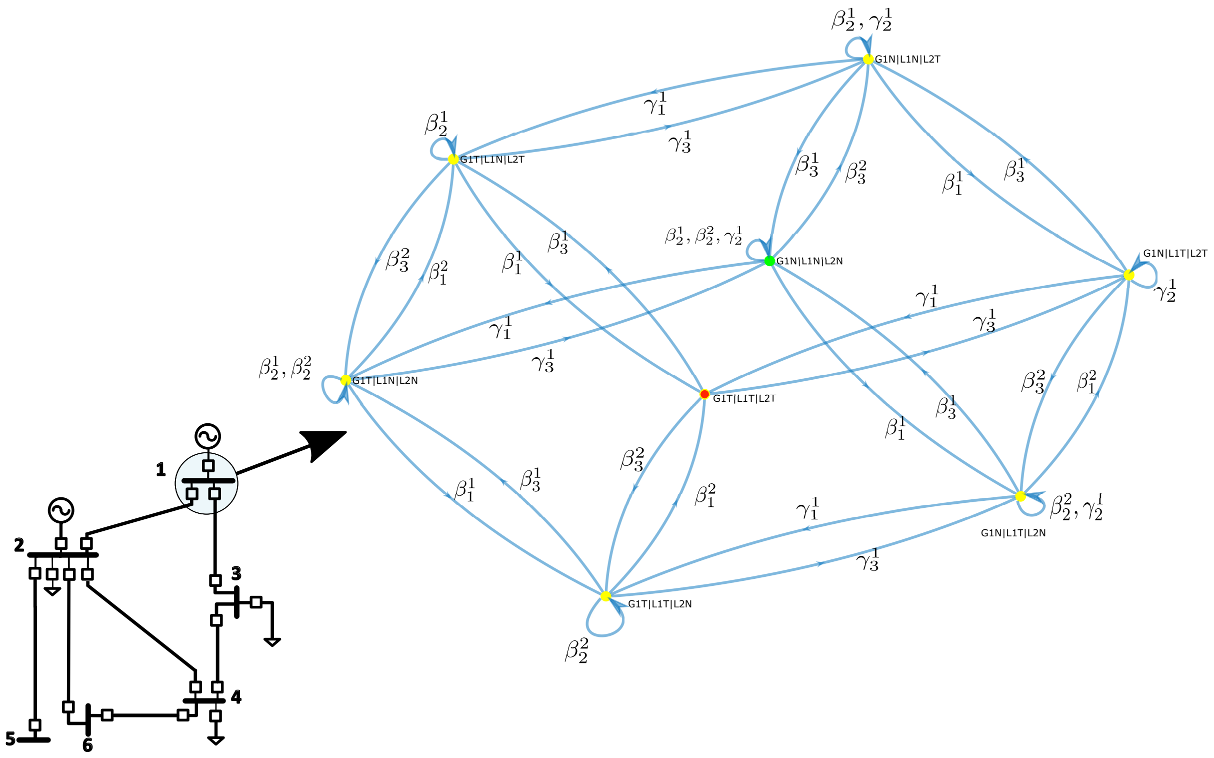

Figure 5.2: Modular approach for Bus 1 of the IEEE 30-Bus system

5.2 Modular supervisory control with extended specification

As previously discussed, the local specifications alone may not ensure the safe operation of a system by the modular supervisors. The supervisor for each node may need the specification from neighboring its nodes. Inspired by the work in [61], we propose to expand the observer projection of the supervisor of each node, so that it will include the events of the neighboring nodes. The authors in [11] proposed an approach similar in principle but for continuous time variables through distributed model predictive control. In our approach the continuous time variables representation will be in a lower layer for each supervisor, while the DES calculations will be used for abstraction at a higher level. The formulation used by the authors in [61] will be used in this dissertation to decompose the specification for the overall system into sublanguages. Consider generators $\mathcal{A}_i$ and $\mathcal{A}_j$ over the alphabets Σ_i and Σ_j, respectively, where $j \in J - J_i$. Given a prefix-closed specification $K \subseteq L(\mathcal{A}_i || \mathcal{A}_j)$. Algorithm 4 in next section finds a coordinator $\mathcal{A}_{cd}$ over Σ_{cd} with $\Sigma_i \cap \Sigma_j \subseteq \Sigma_{cd} \subseteq \Sigma_i \cup \Sigma_j$ such that (1) $\mathcal{A}_{cd} = \theta_{cd}(\mathcal{A}_i)||\theta_{cd}(\mathcal{A}_j)$ and (2) K is conditionally decomposable with respect to Σ_i, Σ_j and Σ_{cd} , that is, $K = \theta_{i+cd}(K)||\theta_{j+cd}(K)$, where $\theta_{i+cd} : (\Sigma_i \cup \Sigma_j)^* \rightarrow \Sigma^*$. Note that $\mathcal{A}_{cd} = \theta_{cd}(\mathcal{A}_i)||\theta_{cd}(\mathcal{A}_j)$ implies $\mathcal{A} = \mathcal{A}_i||\mathcal{A}_j = \mathcal{A}_i||\mathcal{A}_j||\mathcal{A}_{cd}$. Fig. 5.3 shows a block diagram of a system with two supervisors, where each supervisor observes events from different projections of its own as discussed above. $\mathcal{S}_1$ and $\mathcal{S}_2$ are supervisors over the alphabets Σ_i and Σ_j, respectively.

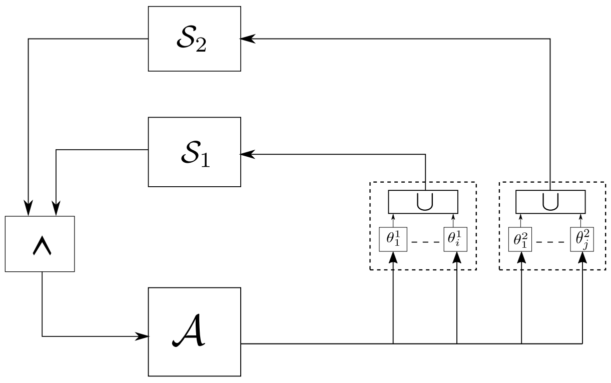

Figure 5.3: Modular approach with several projections from local and neighboring sites for each supervisor

The authors in [62] extended modular supervisory control architecture that implements the control actions of a set of local supervisors. The extension included implementing control actions of a set of local supervisors to use both fusion by intersection and fusion by a union of enabled events. The implementation of a union of control actions as apposite to equation 5.1 can be expressed as follows,

$$L(\mathcal{S}_{mod}/\mathcal{A}) = L_{a1}^{\uparrow c} \cup ... \cup L_{ai}^{\uparrow c} \tag{5.2}$$

The three architectures are shown in Fig. 5.4. In this dissertation, we will consider union of control actions or disjunctive fusion. It may result from the previous discussion that the control agents may pursue only their local control objectives, which may result in not achieving the global objective. Previous research [63] introduced achieving global objectives by different control agents in DES. These control agents are connected through a network, and each one has partial observation of the events of the system. Research has also been done on modular on-line supervisory control in [64]. The analysis discussed in this section considers only the conventional controllability. F-controllability has not been extended to modular control yet. It will be introduced in application in this dissertation, mainly by introducing forcible events preempting uncontrollable events in a modular fashion for different sites i. The formal definition is yet to be introduced in future work.

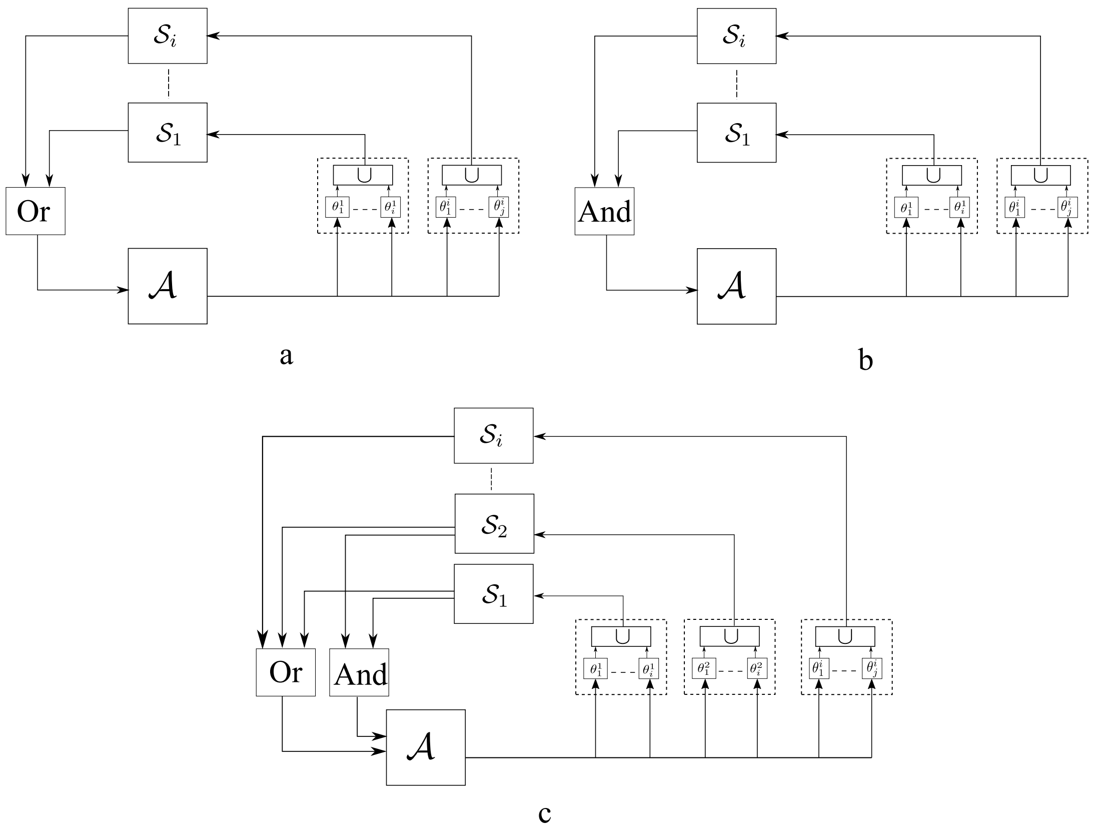
$\mathcal{S}_i$
$\mathcal{S}_1$
$\mathcal{S}_2$
Or
And
$\mathcal{A}$
$\cup$
θ_1^1 θ_i^1
θ_1^2 θ_i^2
θ_1^i θ_j^i
a
b
c

Figure 5.4: Architectures of fusion of enabled events in modular supervisory control: a- Fusion by union, b- Fusion by intersection c- Fusion by union and intersection (general architecture)

5.3 Proposed algorithm for modular control approach implementation

The implementation of the proposed modular control approach can be divided into two main tasks. The first task includes the modeling of continuous time variables for each node i and transforming the status changes of these variables as events to each modular supervisor $\mathcal{S}_i$. This task will also require transforming related events with alphabet Σ_{cd} to $\mathcal{S}_i$. The analysis above for the alphabets $\Sigma_i, \Sigma_j, \Sigma_{cd}$ can be extended such that Σ_{cd} is the set of the events of all neighboring nodes, linked to $\mathcal{A}_i$ as shown in Fig. 5. Σ_j can be considered in this case the alphabets related to the components attached to the neighboring nodes of node i. The computation of Σ_{cd} can be done through Algorithm 4.

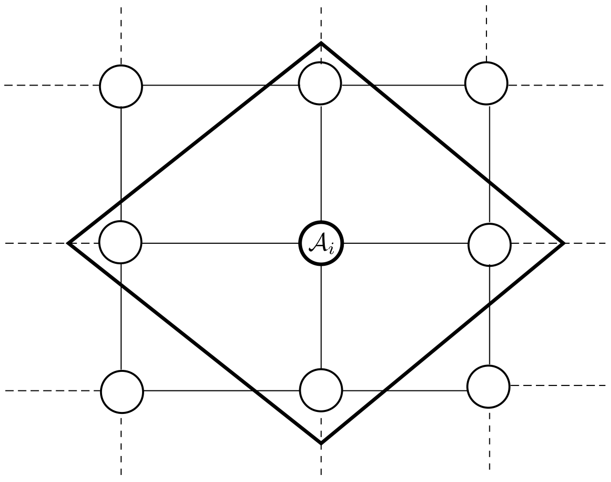

Figure 5.5: $\mathcal{A}_i$ and neighboring nodes

First step is to construct automaton $\mathcal{A}_{cd}$ (coordination automaton) for bus i. To do so, we start by making $\mathcal{A}_{cd}$ to be the overlap between $\mathcal{A}_i$ and its neighboring automata and expanding $\mathcal{A}_{cd}$, such that it may include all of the neighboring nodes events. In studying the problem of cascading failures, neighboring automata include neighboring nodes and their associated components. The overlap between two nodes is the connecting transmission line automata. The goal is to construct the specification language for node i , K_i, which can be found from $\mathcal{A}_i$ and $\mathcal{A}_{cd}$, i.e., $\mathcal{A}_i||\mathcal{A}_{cd} \implies K_i$. The supervisor is then

Algorithm 4 Construction of a Coordinator

```
Input: $\mathcal{A}_i | i \in$ Power system bus domain, $K$ for all system
Output: $\Sigma_{cd}, \mathcal{A}_{cd}$
1: Let $\Sigma_{cd} = \Sigma_i \cap \Sigma_j$ be the set of all shared events of the language generators of nodes
   $i$ and $j$ $\mathcal{A}_i$. i.e., and $\mathcal{A}_j$
2: Extend the alphabet $\Sigma_{cd}$ so that $K_i$ becomes conditional decomposable with respect
   to $\Sigma_i, \Sigma_j$, and $\Sigma_{cd}$
3: Define the coordinator $\mathcal{A}_{cd} = \theta_{cd}(\mathcal{A}_i)||\theta_{cd}(\mathcal{A}_j)$
```

$L(\mathcal{S}_i/\mathcal{A}) = K_i$. As discussed earlier in Section 5.2, each supervisor may need to receive events from neighboring nodes through projection. In addition, each supervisor needs to communicate control actions to neighboring nodes as well.

Continuous-time implementation for the power system node-based controllers is vital for the implementation of the DES modular supervisors for the problem under study. Since this implementation will be responsible for receiving continuous-time variables and issuing final control action, this part will also be constructed in this dissertation. The supervisor language $K_i^{\uparrow c}$ for node i is generated through an automaton $H_i^{\uparrow c}$, $H_i^{\uparrow c} = (Q_{Hi}^{\uparrow c}, \Sigma_i, \delta_{Hi}^{\uparrow c}, q_{oi})$, with feedback mapping, $\psi_i : Q \to 2^{\Sigma_i}$. $\mathcal{S}_i = (H_i^{\uparrow c}, \psi_i)$. Fig. 5.6 shows a block diagram of the proposed modular approach for each node i, where the implementation will be divided into two main parts: The continuous time model and the supervisor realization of each node.

The proposed implementation of the supervisor's conjunction of the overall system is described in Algorithm 5. The proposed implementation will also be implemented in Matlab. The cascading failure process will be simulated using the same algorithm that is used for on-line control approach in Chapter 4. This part is different from the on-line

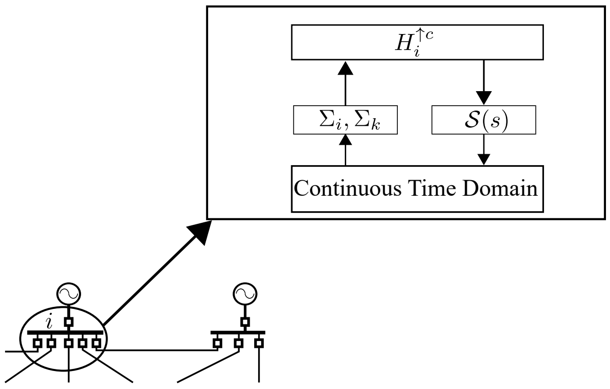

Figure 5.6: Modular control approach for each node i

control procedure that will require dedicated DES software. For this, LibFAUDES [65] is used, which is a C++ library. To encapsulate this library in Matlab environment, Matlab executable function (MEX) is used. The authors in [38] implemented a similar approach for coordination between two supervisors at the ends of a HVDC. Our approach, however, includes the whole network.

Algorithm 5 Supervisors $\mathcal{S}_i/\mathcal{A}$ implementation

```
Input: $\mathcal{A}_i$ , $\Sigma_{cd}$
Output: Control actions
1: Project events $\Sigma_i$ from node $i$ to $\mathcal{S}_i$
2: Communicate events $\Sigma_{cd}$ from the neighboring nodes to $\mathcal{S}_i$ through projection
3: Compute $K_i^{\uparrow c}$ from $H_i^{\uparrow c}$
4: Inject control actions from $\mathcal{S}_i$ back the node $i$ and neighboring nodes through events
   fusion
```

Algorithm 5 can be depicted in Fig. 5.7. which similar to the implementation of the on-line control. The continuous-time model of the power system is abstracted as an automaton for each site ($\mathcal{A}^i$). The information is transmitted from $\mathcal{A}^i$ as observable events. We assume in this dissertation that all events in $\mathcal{A}^i$ and neighboring nodes are observable to $\mathcal{H}_i^{\uparrow c}$. Each $\mathcal{S}_i$ will need to communicate events with neighboring nodes.

This is shown in dashed arrows in Fig. 5.7.

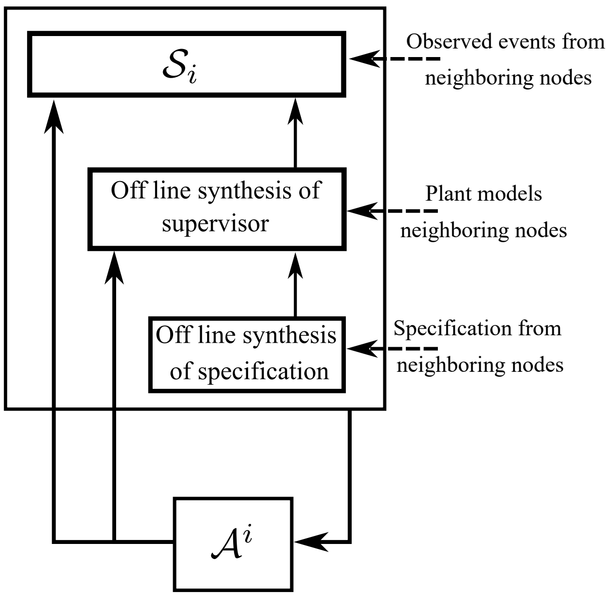
$\mathcal{S}_i$
Observed events from neighboring nodes
Off line synthesis of supervisor
Plant models neighboring nodes
Off line synthesis of specification
Specification from neighboring nodes
$\mathcal{A}^i$

Figure 5.7: Modular control approach for each node i

5.4 DES modular control implementation

In this section, the main results and findings of modular control are discussed analyzed. We first illustrate the proposed approach in a simple two-node example.

5.4.1 Plant model

The plant model is similar to the model developed in Section 2.2. Here, we restrict the model to the matter of interest node and the neighboring nodes. The benefit of the plant model in the modular control approach is that it will be used to construct the specification and synthesize the supervisor. As stated earlier, DES plant model represents the high-level abstraction of the continuous-time of the system. In the modular control approach, the number of states will not suffer state explosion and increasing computation complexity. Take an example from the simple power system shown in Fig. 5.8, which shows node 1 and its neighboring node, node 2.

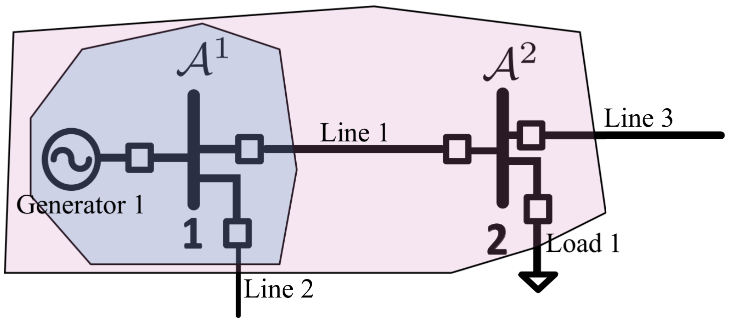

Figure 5.8: Simple power system for with 2 nodes

First, take $\mathcal{A}^1$ as an automaton as shown in Fig. 5.9-b. $\mathcal{A}^1$ represents node one with components of lines 1 and 2, and generator 1. The automaton model of each component from Fig. 2.4 is also shown in Fig. 5.9-a. This automaton represents $\mathcal{A}^i$ in Fig. 5.7. We want to include $\mathcal{A}^2$ plant model to enable specification and supervisor synthesis. The events of this automaton are observed by $\mathcal{H}_i^{\uparrow c}$ in Fig. 5.7 represented by dashed arrow. Similar to $\mathcal{A}^1$, $\mathcal{A}^2$ represents node 2 with components of lines 1 and 3, and load 1.

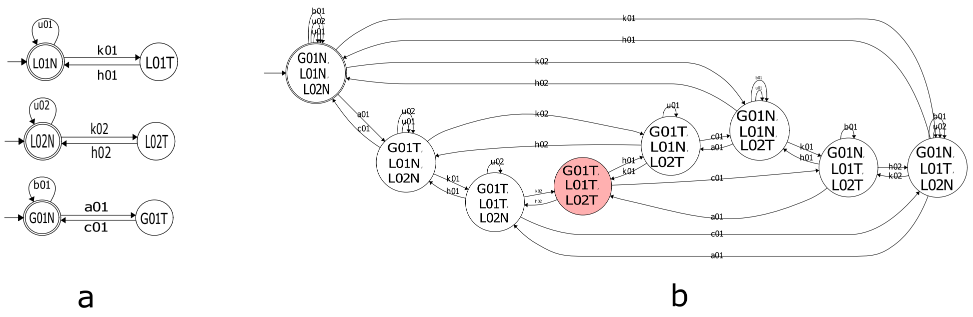

Figure 5.9: Plant model of $\mathcal{A}^1$. a- Automaton models of individual components connected to node 1. b- Parallel composition of all the components in a

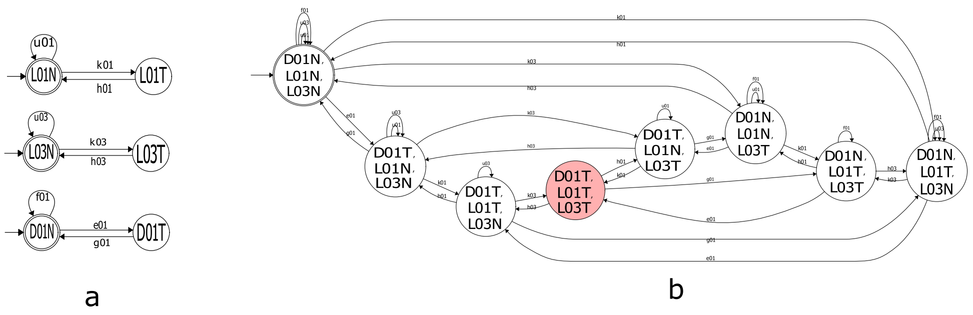

Figure 5.10: Plant model of $\mathcal{A}^2$. a- Automaton models of individual components connected to node 2. b- Parallel composition of all the components in a

Automaton $\mathcal{A}^1$ can be expanded to include a neighboring node to enable specification and supervisor synthesis by setting $\mathcal{A}^1||\mathcal{A}^2$:

$$\mathcal{A} \quad = \mathcal{A}^1||\mathcal{A}^2$$

This will result an automaton shown in Fig 5.11.

5.4.2 Main and extended specification

As mentioned earlier in Sections 3.2 and 5.1, it is desired to construct a specification that is necessary for the supervisor synthesis. The specification will restrict $L(\mathcal{S}_i/\mathcal{A}_i)$ to a sublanguage K_i. The Automaton generating K_i will be based on $\mathcal{A}_i$ but with restricting the behavior not to reach illegal states, i.e., removing the events that will lead to illegal states. In the case of extending $\mathcal{A}_{cd}$ discussed in Section 5.2, the extension of the specification needs to include the neighboring nodes' events. In the illustrative example,

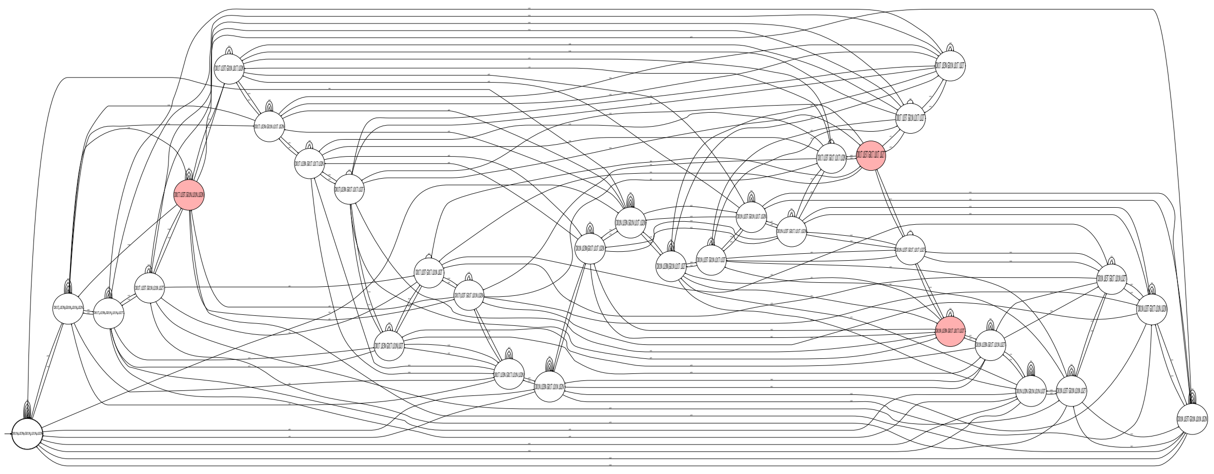
Figure 5.11: Plant model that includes $\mathcal{A}^1$ and $\mathcal{A}^2$

to create a specification automaton, we remove illegal state, i.e., $(Q - Q_H)$, from the extended plant model, i.e., the automaton in Fig. 5.11. In this case, $(Q - Q_H)$ represented by the states $(D01T, L03T, G01T, L01T, L02T)$, $(D01N, L03N, G01T, L01T, L02T)$ and $(D01T, L03T, G01N, L01N, L02N)$, highlighted in red in Fig. 5.11. This will result an automaton that is similar to $\mathcal{H}$ defined in Section 3.2, except in this case $\mathcal{H}$ is defined over $\mathcal{A}_1$ and $\mathcal{A}_2$ only, we denote this automaton as $\mathcal{H}^{1,2}$,

$$\mathcal{H}^{1,2} = (Q_H^{1,2}, \Sigma^{1,2}, \delta_H^{1,2}, q_o)$$

Where $Q_H^{1,2} \subseteq Q^{1,2}$ and $\delta_H^{1,2} = \delta^{1,2}|_{Q_H^{1,2}} \subseteq \delta$ ($\delta^{1,2}|_{Q_H^{1,2}}$ means δ restricted to $Q_H^{1,2}$).
$\mathcal{H}^{1,2}$ can be represented by the specification automaton shown in Fig. 5.12. Specification automaton will generate the admissible language of node 1. In other words,

$$L(\mathcal{H}^{1,2}) = L_a^1$$

5.4.3 Supervisor synthesis

To synthesize the supervisor, an algorithm in [65] is used. We define the controllable and forcible events in order to enable the synthesis. Table 5.1 shows the controllability

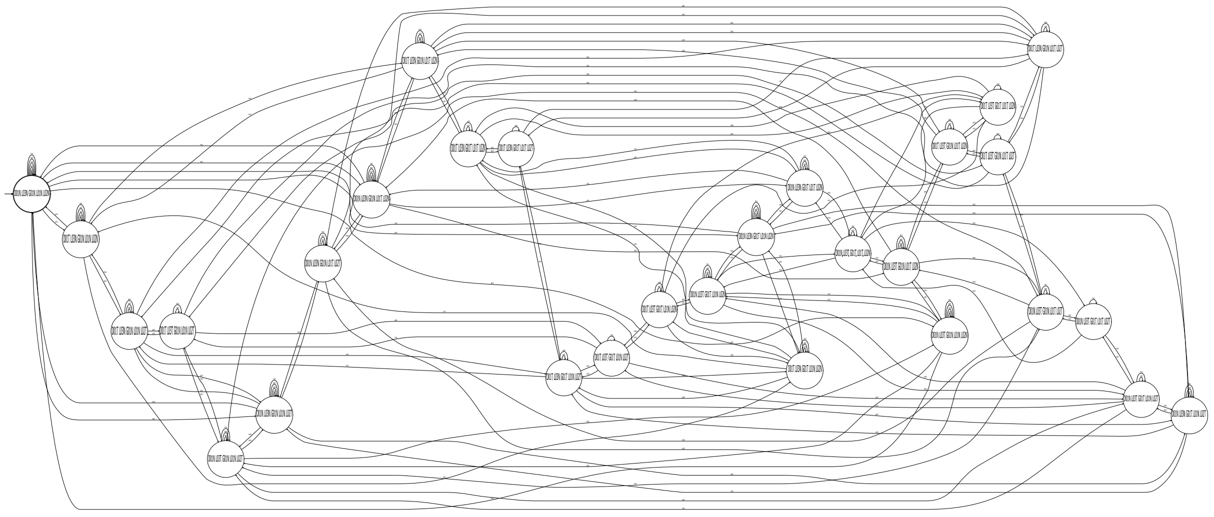

Figure 5.12: Extended Specification Automaton for $\mathcal{A}^1$, $\mathcal{H}^{1,2}$

attributes of the automaton shown in Fig. 5.11. We assume transmission lines tripping events are controllable in the sense that they can be preempted by forcible events.

Table 5.1: Events Attributes definition table

Component designation	Component number	connected to bus	controllability attributes
Transmission line $\Longrightarrow$ k	01	1,2	k01 $\Longrightarrow$ Controllable
Transmission line $\Longrightarrow$ k	02	1	k02 $\Longrightarrow$ Controllable
Transmission line $\Longrightarrow$ k	03	2	k03 $\Longrightarrow$ Controllable
Generator $\Longrightarrow$ b	01	1	a01 $\Longrightarrow$ Forcible
Load $\Longrightarrow$ u	01	2	e01 $\Longrightarrow$ Forcible

After defining controllability attributes and the specification language, and assuming that all events are observable, a supervisor can be constructed for node i, i.e., $\mathcal{H}_i$. The definition of the supervisor that is based on forcible events, as mentioned earlier in Section 3.2,

$$\begin{aligned} \mathcal{S}^{\uparrow\diamond}(w) = \quad & \{\sigma \in \Sigma : (\forall q \in \xi^{\uparrow}(w)) \\ & \delta(q,\sigma) \in Q \Rightarrow \delta(q,\sigma) \in Q_H^{\uparrow}\}. \end{aligned}$$

The supervisor $\mathcal{S}^{\uparrow\diamond}$ realization then will be defined as an automaton of the form: $\mathcal{H} = (Q_H, \Sigma, \delta_H, q_o)$. When synthesizing the supervisor, we need to build a representation of the function $\mathcal{S}_1$. This presentation is called supervisor $\mathcal{S}_1$ realization, which is built as an automaton off-line. We denote $\mathcal{S}_1$ realization as R. Let R be defined as follows

$$R := (Y, \Sigma, g, y_0)$$

Fig. 5.13 shows the supervisor realization of node 1 as an automaton.

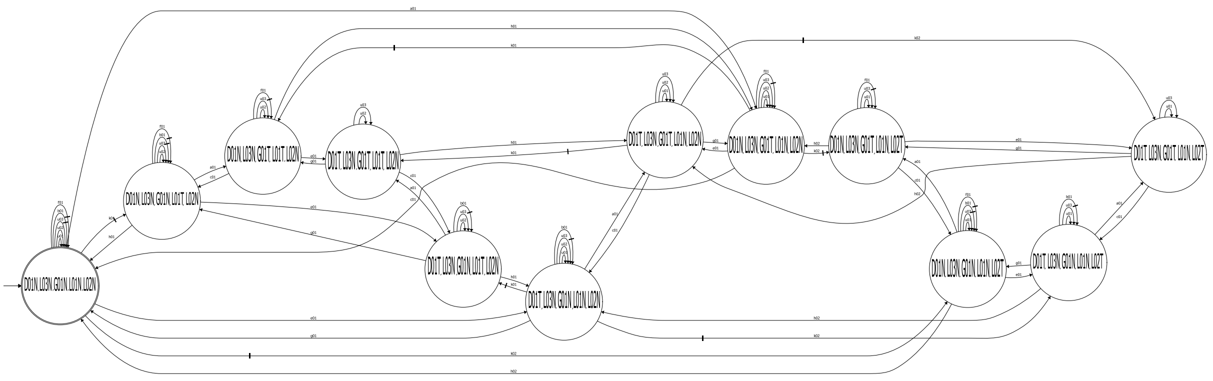

Figure 5.13: $\mathcal{S}_1$: Supervisor realization of node 1

5.5 Simulation results of DES modular control

The framework presented in the previous section is implemented for the IEEE 6-bus and 30-bus systems in this section. The simulation setup was done in Matlab environment, where Matpower is used to simulate the failure cascade and to send and receive information from the supervisors. The supervisors are implemented in Matlab using Matalb executable (MEX), where the DES operations functions are imported from C++ LibFAUDES DES library into Matlab. The framework is explained in Fig. 5.14. The difference between the implementation of modular control in Chapter 5 and on-line control in Chapter 4 is that for the on-line control, we implemented the LLP controller in Matlab directly by constructing the lookahead tree object file based on the automata models of the power systems components. No external DES library was imported to Matlab for the on-line control, such as the one implemented here in Chapter 5.

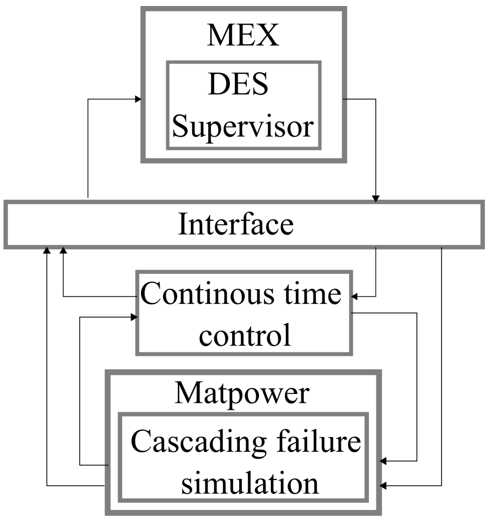

Figure 5.14: MATLAB Simulation setup of Modular DES

As mentioned earlier, the supervisors are synthesized based on the specification and the plant models. The plant models are based on the extended plant model of the nodes that observe faults and the surrounding nodes of this node, i.e., the nodes directly connected via a transmission line. The nodes that observe faulted lines are the nodes

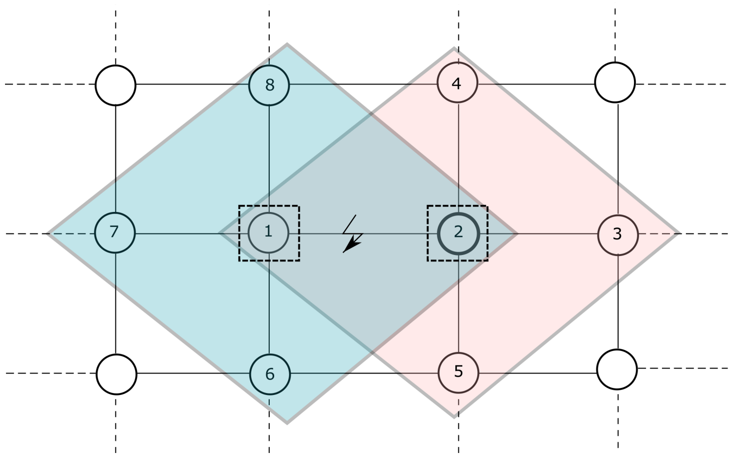

Figure 5.15: An example of two nodes: 1 and 2 observing a faulted line connecting them

that get their supervisors activated and take control actions. Taking an example of the network in Fig. 5.15, if the line connecting nodes 1 and 2 is tripped, then nodes 1 and 2 supervisors only will take action. Node 1 supervisor specification and plant model is including node one and all its surrounding nodes, that is nodes: 2, 6, 7, and 8. Similarly, for the supervisor of node 2, the control actions taken by each supervisor are either actuated on the same node for that supervisor or for one of the surrounding nodes that are included in the specification language. In the case of the two supervisors sending two control actions to the same node, the control actions with the highest value of optimal load shedding or generation re-dispatch are actuated by the target node.

For the optimal control actions of load shedding and generation re-dispatch, a modified approach used for the on-line control is used. The optimization problem

$$\begin{aligned}
&\underset{x}{min} \textstyle\sum_j x_j \\
&s.t. \\
&\sqrt{y_t}(A_t^T)^{-1}x \leqslant F_{Max} - P^0 \\
&D \times x \leqslant F_{lMax} - \mu_{F(t)}
\end{aligned} \tag{5.3}$$

is modified such that only control actions at the nodes of interest are allowed, as mentioned earlier. $D = (PTDF_{i,lc})$ is the value of PTDF of the buses for the critical transmission line l_c. It is calculated based on the original power system assuming that the supervisor does not have any knowledge of any other line is tripped, other than the lines that are directly connected to it.

Simulation studies have been carried out on the IEEE 6-bus, and the IEEE 30-bus systems to verify the findings of the proposed modular control. Fig. 5.16 and 5.18 show the same comparison as in Section 4.3, but for one $N-2$ contingency for the IEEE 6-bus system, which is the lines pair $\{(2, 9)\}$. For the 30-bus system, the $N-2$ contingency lines pair is: $\{(14, 40)\}$. Fig. 5.17 and 5.19 show the results of two specific scenarios from the IEE 6-bus and 30-bus systems simulation results, respectively. The modular control approach for both systems stopped the failure cascading but with higher MW lost than the emergency control method and the LLP DES control method discussed in Chapter 4. For the IEEE 30-bus system cases study at some scenarios, the failure cascade was stopped but with additional transmission lines tripped, as shown in Fig. 5.19. The higher MW lost is because that in modular control, the modular supervisors can only make decisions based on the information received from neighboring nodes and can only send control actions to the neighboring nodes, making the solution local and not global.

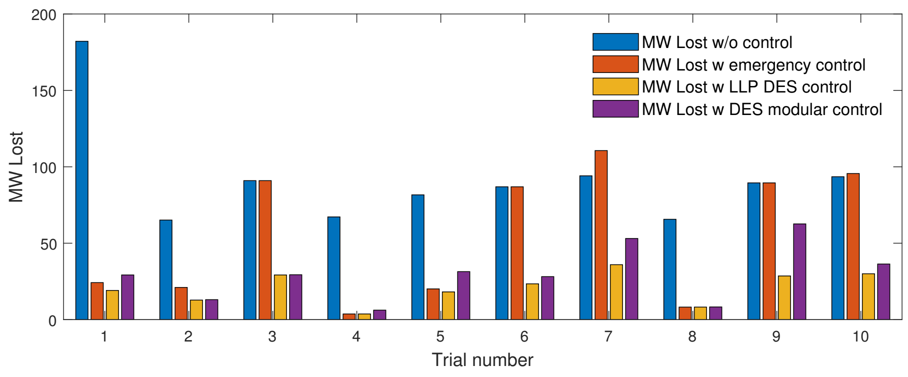

Figure 5.16: MW lost comparison for the IEEE 6-Bus system including Modular DES

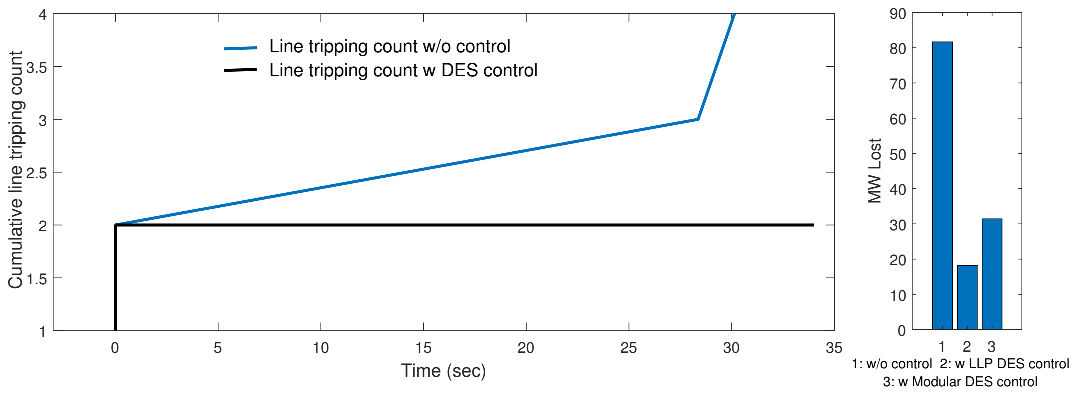

Figure 5.17: Effect of applying modular DES control approach for the IEEE 6-Bus system

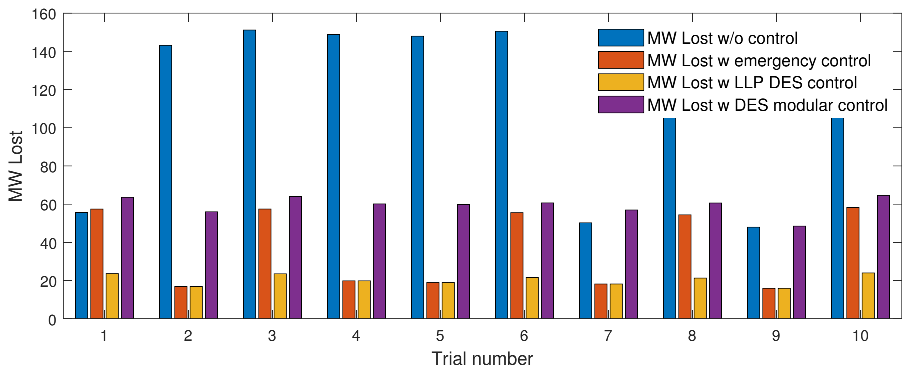

Figure 5.18: MW lost comparison for the IEEE 30-Bus system including Modular DES

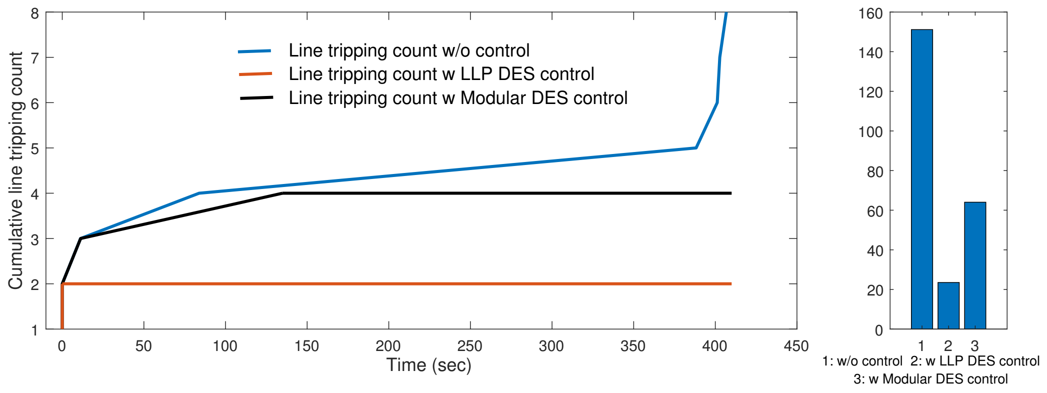

Figure 5.19: Effect of applying modular DES control approach for the IEEE 30-Bus system

5.6 Conclusion

In this chapter, we introduced modular supervisory control as a solution to overcome the computation complexity of large systems. Modular control was extended to include forcible events. In order to achieve better control actions for each of the modular controllers, the specification language was extended to include more events from the neighboring nodes. A framework to implement the modular strategy was proposed based on Matlab environment. The DES part and the continuous-time part of the control were coupled and tested with power systems simulation. Two case studies were simulated to verify the effectiveness of the proposed approach, the IEEE 6-bus, and 30-bus systems. Compared with the LLP approach proposed in Chapter 4, the modular control approach has more MW lost with more lines tripped after $N-2$ contingency observed. Nevertheless, the modular supervisory control can stop cascading failures and can be considered more reliable since it does depend on a central controller and each node requires information only from its neighbors.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Conclusions

This dissertation presents a DES-based approach for modeling, evaluating, and mitigating the risk of cascading failure in power systems. A modified framework of Supervisory Control of DES was developed by introducing F-controllability with forcible events. The necessary and sufficient condition for the existence of such a controller has been given in the paper. To overcome the issue of enormously large state space and the increase in computational complexity, two approaches are presented: the first one is an an on-line supervisory lookahead control $S^M_{CLL}(q)$ with forcible events, which was proposed based on lookahead tree $Tree(q)$ of depth M. The on-line control approach has been implemented for mitigating power system cascading failure problems. The second approach is based on modular supervisory control, where the supervisors are based on individual nodes of the power system. The specification for each supervisor is extended to include the surrounding nodes.

Case studies with joint simulations between continuous variables and discrete events have been carried out in Matlab to verify the effectiveness of the two proposed approaches. $N - k$ contingencies were applied to the IEEE 6-bus, IEEE 30-bus, and the IEEE 118-bus systems to test the proposed control methods. Forcible events have been applied as generation re-dispatch and load shedding actions for mitigating cascading failures. The simulation results show the proposed method is effective in minimizing the risk and impact of cascading failure.

6.2 Future work

As the power grids shift to smart grids, cyberinfrastructure is used for distributed energy resources management, power system market operation, wide-area measurements, etc. Smart grids can be presented as cyber-physical systems (CPS). The power grid and the power system components control represent the physical layer in CPS. The cyber layer represents the wide-area control and communication network between agents. This conjunction between two networks may impact the security and reliability of the system, one of the properties needed for future power systems is to be attack resistance. This is a promising area of research which can include DES. The work in this dissertation can be extended to address cyber security for power systems.

BIBLIOGRAPHY

[1] P. Hines, J. Apt, and S. Talukdar, “Large blackouts in north america: Historical trends and policy implications,” *Energy Policy*, vol. 37, no. 12, pp. 5249 – 5259, 2009.

[2] M. Vaiman, K. Bell, Y. Chen, B. Chowdhury, I. Dobson, P. Hines, M. Papic, S. Miller, and P. Zhang, “Risk assessment of cascading outages: Methodologies and challenges,” *IEEE Transactions on Power Systems*, vol. 27, no. 2, p. 631, 2012.

[3] Ming Ni, J. D. McCalley, V. Vittal, and T. Tayyib, “Online risk-based security assessment,” *IEEE Transactions on Power Systems*, vol. 18, no. 1, pp. 258–265, 2003.

[4] M. Vaiman, P. Hines, J. Jiang, S. Norris, M. Papic, A. Pitto, Y. Wang, and G. Zweigle, “Mitigation and prevention of cascading outages: Methodologies and practical applications,” in *2013 IEEE Power Energy Society General Meeting*, 2013, pp. 1–5.

[5] J. Chen, J. S. Thorp, and I. Dobson, “Cascading dynamics and mitigation assessment in power system disturbances via a hidden failure model,” *International Journal of Electrical Power & Energy Systems*, vol. 27, no. 4, pp. 318–326, 2005.

[6] P. D. Hines, I. Dobson, and P. Rezaei, “Cascading power outages propagate locally in an influence graph that is not the actual grid topology,” *IEEE Transactions on Power Systems*, vol. 32, no. 2, pp. 958–967, 2016.

[7] Y. Susuki, Y. Takatsuji, and T. Hikihara, “Hybrid model for cascading outage in a power system: A numerical study,” *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 92, no. 3, pp. 871–879, 2009.

[8] J. Song, E. Cotilla-Sanchez, G. Ghanavati, and P. D. Hines, “Dynamic modeling of

cascading failure in power systems," *IEEE Transactions on Power Systems*, vol. 31, no. 3, pp. 2085–2095, 2015.

[9] C. Asavathiratham, S. Roy, B. Lesieutre, and G. Verghese, "The influence model," *IEEE Control Systems Magazine*, vol. 21, no. 6, pp. 52–64, 2001.

[10] M. Rahnamay-Naeini, Z. Wang, N. Ghani, A. Mammoli, and M. M. Hayat, "Stochastic analysis of cascading-failure dynamics in power grids," *IEEE Transactions on Power Systems*, vol. 29, no. 4, pp. 1767–1779, 2014.

[11] P. Hines and S. Talukdar, "Reciprocally altruistic agents for the mitigation of cascading failures in electrical power networks," in *2008 First International Conference on Infrastructure Systems and Services: Building Networks for a Brighter Future (INFRA)*, 2008, pp. 1–6.

[12] S. Talukdar, Dong Jia, P. Hines, and B. H. Krogh, "Distributed model predictive control for the mitigation of cascading failures," in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 4440–4445.

[13] B. Li and G. Sansavini, "Effective multi-objective selection of inter-subnetwork power shifts to mitigate cascading failures," *Electric Power Systems Research*, vol. 134, pp. 114 – 125, 2016.

[14] A. A. Babalola, R. Belkacemi, and S. Zarrabian, "Real-time cascading failures prevention for multiple contingencies in smart grids through a multi-agent system," *IEEE Transactions on Smart Grid*, vol. 9, no. 1, pp. 373–385, 2018.

[15] J. Guo, F. Liu, J. Wang, J. Lin, and S. Mei, "Toward efficient cascading outage simulation and probability analysis in power systems," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 2370–2382, 2018.

[16] L. Liu, H. Wu, L. Li, D. Shen, F. Qian, and J. Liu, "Cascading failure pattern identification in power systems based on sequential pattern mining," *IEEE Transactions on Power Systems*, pp. 1–1, 2020.

[17] X. Wei, J. Zhao, T. Huang, and E. Bompard, "A novel cascading faults graph based transmission network vulnerability assessment method," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 2995–3000, 2018.

[18] K. Zhou, I. Dobson, Z. Wang, A. Roitershtein, and A. P. Ghosh, "A markovian influence graph formed from utility line outage data to mitigate large cascades," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 3224–3235, 2020.

[19] J. Bialek, E. Ciapessoni, D. Cirio, E. Cotilla-Sanchez, C. Dent, I. Dobson, P. Henneaux, P. Hines, J. Jardim, S. Miller, M. Panteli, M. Papic, A. Pitto, J. Quiros-Tortos, and D. Wu, "Benchmarking and validation of cascading failure analysis tools," *IEEE Transactions on Power Systems*, vol. 31, no. 6, pp. 4887–4900, 2016.

[20] P. Henneaux, E. Ciapessoni, D. Cirio, E. Cotilla-Sanchez, R. Diao, I. Dobson, A. Gaikwad, S. Miller, M. Papic, A. Pitto, J. Qi, N. Samaan, G. Sansavini, S. Uppalapati, and R. Yao, "Benchmarking quasi-steady state cascading outage analysis methodologies," in *2018 IEEE International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, 2018, pp. 1–6.

[21] L. Guo, C. Liang, A. Zocca, S. H. Low, and A. Wierman, "Line failure localization of power networks part i: Non-cut outages," *IEEE Transactions on Power Systems*, vol. 36, no. 5, pp. 4140–4151, 2021.

[22] ——, "Line failure localization of power networks part ii: Cut set outages," *IEEE Transactions on Power Systems*, vol. 36, no. 5, pp. 4152–4160, 2021.

[23] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[24] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information sciences*, vol. 44, no. 3, pp. 173–198, 1988.

[25] W. Wonham, K. Cai, and K. Rudie, "Supervisory control of discrete-event systems: A brief history," *Annual Reviews in Control*, vol. 45, pp. 250–256, 2018.

[26] W. M. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems.* Springer International Publishing, 2019.

[27] F. Lin, "Diagnosability of discrete event systems and its applications," *Discrete Event Dyn Syst*, vol. 4, no. 2, p. 197–212, 1994.

[28] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[29] A. A. Afzalian, S. A. Nabavi Niaki, M. R. Iravani, and W. M. Wonham, "Discrete-event systems supervisory control for a dynamic flow controller," *IEEE Transactions on Power Delivery*, vol. 24, no. 1, pp. 219–230, 2009.

[30] A. Afzalian, A. Saadatpoor, and W. Wonham, "Systematic supervisory control solutions for under-load tap-changing transformers," *Control Engineering Practice*, vol. 16, no. 9, pp. 1035 – 1054, 2008.

[31] B. Zhao, F. Lin, C. Wang, X. Zhang, M. P. Polis, and L. Y. Wang, "Supervisory control of networked timed discrete event systems and its applications to power

distribution networks," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 146–158, 2017.

[32] S.-L. Chung, S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1921–1935, 1992.

[33] N. B. Hadj-Alouane, S. Lafortune, and F. Lin, "Variable lookahead supervisory control with state information," *IEEE Transactions on Automatic control*, vol. 39, no. 12, pp. 2398–2410, 1994.

[34] J. Zhao, Y.-L. Chen, Z. Chen, F. Lin, C. Wang, and H. Zhang, "Modeling and control of discrete event systems using finite state machines with variables and their applications in power grids," *Systems and Control Letters*, vol. 61, no. 1, pp. 212–222, 2012.

[35] "Decentralized supervisory control of discrete-event systems," *Information Sciences*, vol. 44, no. 3, pp. 199 – 224, 1988.

[36] K. Rudie and W. M. Wonham, "Think globally, act locally: Decentralized supervisory control," in *1991 American Control Conference*, 1991, pp. 898–903.

[37] M. R. Rodríguez, R. Delpoux, L. Piétrac, J. Dai, A. Benchaib, and E. Niel, "Supervisory control for high-voltage direct current transmission systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 326 – 12 332, 2017, 20th IFAC World Congress.

[38] M. Romero-Rodríguez, R. Delpoux, L. Piétrac, J. Dai, A. Benchaib, and E. Niel, "An implementation method for the supervisory control of time-driven systems applied to high-voltage direct current transmission grids," *Control Engineering Practice*, vol. 82, pp. 97 – 107, 2019.

[39] A. Kafetzis, C. Ziogou, K. Panopoulos, S. Papadopoulou, P. Seferlis, and S. Voutetakis, “Energy management strategies based on hybrid automata for islanded microgrids with renewable sources, batteries and hydrogen,” *Renewable and Sustainable Energy Reviews*, vol. 134, p. 110118, 2020.

[40] S. Horowitz, A. Phadke, and J. Niemira, *Power System Relaying.* Wiley, 2013.

[41] S. Horowitz and A. Phadke, “Blackouts and relaying considerations - relaying philosophies and the future of relay systems,” *IEEE Power and Energy Magazine*, vol. 4, no. 5, pp. 60–67, 2006.

[42] “A critical review of cascading failure analysis and modeling of power system,” *Renewable and Sustainable Energy Reviews*, vol. 80, pp. 9 – 22, 2017.

[43] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez-Gasca, R. Schulz, A. Stankovic, C. Taylor, and V. Vittal, “Causes of the 2003 major grid blackouts in north america and europe, and recommended means to improve system dynamic performance,” *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1922–1928, 2005.

[44] O. P. Veloza and F. Santamaria, “Analysis of major blackouts from 2003 to 2015: Classification of incidents and review of main causes,” *The Electricity Journal*, vol. 29, no. 7, pp. 42–49, 2016.

[45] “2020 state of reliability report; an assessment of 2019 bulk power system performance,” The North American Electric Reliability Corporation (NERC), Tech. Rep., July 2020.

[46] Power System Relaying Committee, “Ieee standard for inverse-time characteristics

equations for overcurrent relays," IEEE Power and Energy Society, New York, USA, Standard IEEE Std C37.112-2018, 2018.

[47] H. Saadat, *Power System Analysis.* McGraw Hill, 1999.

[48] J. Zhu, *Optimization of Power System Operation.* IEEE press series on power engineering. John Wiley Sons, 2015.

[49] B. Schäfer, D. Witthaut, M. Timme, and V. Latora, "Dynamically induced cascading failures in power grids," *Nature Communications*, vol. 9, no. 1, p. 1975, 2018.

[50] M. Begovic, D. Novosel, D. Karlsson, C. Henville, and G. Michel, "Wide-area protection and emergency control," *Proceedings of the IEEE*, vol. 93, no. 5, pp. 876–891, 2005.

[51] M. J. Eppstein and P. D. H. Hines, "A "random chemistry" algorithm for identifying collections of multiple contingencies that initiate cascading failure," *IEEE Transactions on Power Systems*, vol. 27, no. 3, 2012.

[52] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems.* Springer Science & Business Media, 2009.

[53] B. Zeigler, A. Muzy, and E. Kofman, *Theory of Modeling and Simulation: Discrete Event and Iterative System Computational Foundations*, 3rd ed. Academic press, 2018.

[54] K. Cai, R. Zhang, and W. M. Wonham, "Relative observability of discrete-event systems and its supremal sublanguages," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 659–670, 2014.

[55] Z. Wang, A. Scaglione, and R. J. Thomas, "A markov-transition model for cascading

failures in power grids," in *2012 45th Hawaii International Conference on System Sciences*. IEEE, 2012, pp. 2115–2124.

[56] W. H. Al-Rousan, C. Wang, and F. Lin, "A discrete event theory based approach for modeling power system cascading failures," in *2019 IEEE Power Energy Society General Meeting (PESGM)*, 2019, pp. 1–5.

[57] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011.

[58] A. Wood, B. Wollenberg, and G. Sheblé, *Power Generation, Operation, and Control*. Wiley, 2013.

[59] R. Pooya, "Cascading failure risk estimation and mitigation in power systems," Ph.D. dissertation, University of Vermont, 2016.

[60] *Online*, https://icseg.iti.illinois.edu/power-cases/.

[61] J. Komenda and F. Lin, "Modular supervisory control of networked discrete-event systems," in *2016 13th International Workshop on Discrete Event Systems (WODES)*, 2016, pp. 85–90.

[62] T. Yoo and S. Lafortune, "A general architecture for decentralized supervisory control of discrete-event systems," *Discrete Event Dynamic Systems*, vol. 12, no. 3, pp. 335–377, 2002.

[63] S.-J. Park and K.-H. Cho, "Achieving a global objective with competing networked agents in the framework of discrete event systems," *International Journal of Control*, vol. 93, no. 4, pp. 889–897, 2020.

[64] Z. Liu, X. Yin, S. Shu, F. Lin, and S. Li, "Online supervisory control of networked

discrete-event systems with control delays," *IEEE Transactions on Automatic Control*, pp. 1–1, 2021.

[65] T. Moor, K. Schmidt, and S. Perk, "libfaudes — an open source c++ library for discrete event systems," in *2008 9th International Workshop on Discrete Event Systems*, 2008, pp. 125–130.

ABSTRACT

POWER SYSTEMS CASCADING FAILURE ANALYSIS AND MITIGATION USING DISCRETE EVENTS SYSTEMS APPROACH

by

WASSEEM ALROUSAN

May 2022

Advisor: Dr. Caisheng Wang

Co-Advisor: Dr. Feng Lin

Major: Electrical Engineering

Degree: Doctor of Philosophy

A power system cascading failure can propagate through sequential tripping of components in the network. As a result, a complete or partial shutdown may occur. Many models were developed to understand the failure propagation mechanism at a higher abstract level and methods were implemented for failure mitigation. This work introduces a unified framework of modeling and mitigating cascading failures. Based on a Discrete Event Systems (DES) approach, a power system is modeled by an automaton via parallel composition of the sub-models of system components. A modified DES supervisory control (SC) strategy is introduced as a solution to mitigating cascading failures and compared with a regular off-line control. The proposed supervisory control strategy is then extended to an on-line based control called limited lookahead policy (LLP) to overcome the increased complexity of large scale systems. An evaluation method and a criterion for assessing cascading failure risk and identifying critical components during failure propagation in the DES framework are then proposed. The proposed on-line control was implemented and verified through joint simulations between continuous time power flow

analysis and discrete events dynamics with supervisory control. An illustrative example of the proposed approach is presented. Simulation studies for the on-line approach are carried out for IEEE 6-bus, 30-bus, and 118-bus systems to verify the effectiveness of the proposed approach. Modular supervisory control is also introduced to mitigate cascading failure for large scale power systems. A framework to implement modular control approach is then proposed. Similar to the on-line LLP approach, simulations based on case studies for the IEEE 6-bus and 30-bus systems are carried out to illustrate the effectiveness of the proposed approach.

AUTOBIOGRAPHICAL STATEMENT

Wasseem Al-Rousan received the B.S. and M.S. degrees in electrical engineering from Yarmouk University, Jordan, and a Ph.D. degree in electrical engineering from Wayne State University, MI, USA. His research interests include Power systems dynamics, Discrete-event systems, supervisory control.