Department of Computing and Software

Faculty of Engineering — McMaster University

Fault Tolerant Controllability and Nonblocking

by

Aos Mulahuwaish, Simon Radel, Oriane Dierikx,
Amal Alsuwaidan, and Ryan J. Leduc

CAS Report Series **CAS-15-12-RL**

Department of Computing and Software **December 2015**
Information Technology Building
McMaster University
1280 Main Street West Hamilton, Ontario, Canada L8S 4K1

Fault Tolerant Controllability and Nonblocking

Aos Mulahuwaish[1], Simon Radel[2], Oriane Dierikx[3], Amal Alsuwaidan[1], and Ryan J. Leduc [1]

[1] Department of Computing and Software, Faculty of Engineering,
McMaster University, Hamilton, Ontario, Canada

[2] Department of Génie Mécanique, ENS DE CACHAN, France

[3] Department of Mechanical Engineering, Technical University of Eindhoven, The Netherlands

Technical Report CAS-15-12-RL
Department of Computing and Software
McMaster University

December 21, 2015

Abstract

In this paper we investigate the problem of fault tolerance in the framework of discrete-event systems (DES). We introduce our setting, and then provide a set of fault tolerant definitions designed to capture different types of fault scenarios and to ensure that our system remains controllable and nonblocking in each scenario. We then present algorithms to verify these properties followed by complexity analyses and correctness proofs of the algorithms. Finally, examples are provided to illustrate our approach.

Keywords: Discrete-Event Systems; Supervisory control; Fault tolerant.

Contents

1 Introduction

Supervisory control theory, introduced by Ramadge and Wonham [1, 2, 3], provides a formal framework for analysing discrete-event systems (DES). In this theory, automata are used to model the system to be controlled and the specification for the desired system behaviour. The theory provides methods and algorithms to obtain a supervisor that ensures the system will produce the desired behaviour.

However, the above typically assumes that the system behavior does not contain faults that would cause the actual system to deviate from the theoretical model. An example is a sensor that detects the presence of an approaching train. If the supervisor relies on this sensor to determine when the train should be stopped in order to prevent a collision, it could fail to enforce its control law if the sensor failed. Our goal in this paper is to develop a way to add fault events to the system's plant model and to categorize some common fault scenarios. We will then develop some properties that will allow us to determine if a supervisor will still be controllable and nonblocking in these scenarios. This paper builds upon our earlier work in Radel et al. [4].

Currently in the DES literature, the most common approach when a fault is detected is to switch to a new supervisor to handle the system in its degraded mode. Such an approach focuses on fault recovery as opposed to fault tolerance. This requires the construction of a second supervisor, and requires that there be a means to detect the occurrence of the fault in order to initiate the switch. In the approach we present in this paper, we use a single supervisor that will behave correctly in the presence of the specified fault scenarios. This method does not rely on detecting the fault, but on fault tolerant supervisors. We will now discuss some relevant previous work.

Lin [5] discussed both robust and adaptive supervisory control in discrete-event systems, including necessary and sufficient conditions for the existence of a robust supervisor. Based on this condition, a robust supervisory control and observation approach for synthesizing a supervisory control was developed. The goal of robust supervision is to synthesize a supervisor that realizes a given desired behavior for all possible systems.

In Park et al. [6], they presented necessary and sufficient conditions for fault tolerant robust supervisory control of discrete-event systems that belong to a set of models. When these conditions are satisfied, fault tolerance can be achieved. In the paper, the results were applied to the design, modelling, and control of a workcell consisting of arc welding (GMAW) robots, a sensor, and a conveyor.

In Paoli et al. [7], the controller was updated based on the information provided by online diagnostics. The supervisor needs to detect the malfunctioning component in the system in order to achieve the desired specification. The authors proposed the idea of safe diagnosability as a step to achieve the fault tolerant control. Two new notations were introduced in this work (safe controllability) and (active fault tolerant system), to characterize the conditions that must be satisfied when solving the fault tolerant control problem using this approach.

Qin Wen et al. [8] introduce a framework for fault-tolerant supervisory control of discrete-event systems. In this framework, plants contain both normal behavior and behavior with faults, as well as a submodel that contains only the normal behavior. The goal of fault-tolerant supervisory control is to enforce a specification for the normal behavior of the plant and to enforce another specification for the overall plant behavior. This includes ensuring

that the plant recovers from any fault within a bounded delay so that after the recovery, the system state is equivalent to a state in the normal plant behavior. They formulate this notion of fault-tolerant supervisory control and provide a necessary and sufficient condition for the existence of such a supervisor. The condition involves notions of controllability, observability and relative-closure together with the notion of stability.

This paper is organized as follows. Section 2 discusses DES preliminaries. Section 3 introduces fault events and the fault scenarios to which they apply. Section 4 presents our fault tolerant controllability definitions while Section 5 presents our fault tolerant nonblocking definitions. Section 6 presents algorithms to verify the fault tolerant controllability and nonblocking properties and provides a complexity analysis. Section 7 presents algorithm correctness proofs and Section 8 provides a manufacturing example to illustrate our approach. Finally, Section 9 provides conclusions and future work.

2 Preliminaries

We now present a summary of the DES terminology that we use in this paper. For more details, please refer to [2].

Let Σ be a finite set of distinct symbols (*events*). Let Σ^+ denote the set of all finite, *non-empty* sequences of events, and Σ^* be the set of all finite sequences of events including ϵ, the *empty string*. We can then define $\Sigma^* := \Sigma^+ \cup \{\epsilon\}$. For $s \in \Sigma^*$, $|s|$ equals the length (number of events) of the string.

Let $L \subseteq \Sigma^*$ be a *language* over Σ. A string $t \in \Sigma^*$ is a prefix of $s \in \Sigma^*$ (written $t \leq s$) if $s = tu$, for some $u \in \Sigma^*$. The *prefix closure* of language L (denoted $\overline{L}$) is defined as $\overline{L} := \{t \in \Sigma^* \,|\, t \leq s \text{ for some } s \in L\}$. Let $\mathrm{Pwr}(\Sigma)$ denote the set of all possible subsets of Σ. For language L, the eligibility operator, $\mathrm{Elig}_L : \Sigma^* \to \mathrm{Pwr}(\Sigma)$, is given by $\mathrm{Elig}_L(s) := \{\sigma \in \Sigma \,|\, s\sigma \in L\}$ for $s \in \Sigma^*$.

A DES automaton is represented as a 5-tuple $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ where Y is the state set, Σ is the event set, the partial function $\delta : Y \times \Sigma \to Y$ is the transition function, y_o is the initial state, and Y_m is the set of marker states. The function δ is extended to $\delta : Y \times \Sigma^* \to Y$ in the natural way. The notation $\delta(y, s)!$ means that δ is defined for $s \in \Sigma^*$ at state y. For DES $\mathbf{G}$, the language generated is denoted by $L(\mathbf{G})$, and is defined to be $L(\mathbf{G}) := \{s \in \Sigma^* |\ \delta(y_o, s)!\}$. The marked behavior of $\mathbf{G}$ is defined as $L_m(\mathbf{G}) := \{\ s \in L(\mathbf{G}) |\ \delta(y_o, s) \in Y_m\}$. The reachable state subset of DES $\mathbf{G}$, denoted Y_r, is $Y_r := \{y \in Y |\ (\exists s \in \Sigma^*)\, \delta(y_o, s) = y\}$. A DES $\mathbf{G}$ is reachable if $Y_r = Y$. We will always assume $\mathbf{G}$ is reachable.

Let $\Sigma = \Sigma_1 \cup \Sigma_2$, $L_1 \subseteq \Sigma_1^*$, and $L_2 \subseteq \Sigma_2^*$. For $i = 1, 2$, $s \in \Sigma^*$, and $\sigma \in \Sigma$, we define the *natural projection* $P_i : \Sigma^* \to \Sigma_i^*$ according to:

$$\begin{aligned} P_i(\epsilon) &= \epsilon, \quad P_i(\sigma) = \begin{cases} \epsilon \text{ if } \sigma \notin \Sigma_i \\ \sigma \text{ if } \sigma \in \Sigma_i \end{cases} \\ P_i(s\sigma) &= P_i(s)P_i(\sigma) \end{aligned}$$

The map $P_i^{-1} : Pwr(\Sigma_i^*) \to Pwr(\Sigma^*)$ is the inverse image of P_i such that for $L \subseteq \Sigma_i^*$, $P_i^{-1}L := \{s \in \Sigma^* | P_i(s) \in L\}$.

Definition 1. *For* $\mathbf{G_i} = (Q_i, \Sigma_i, \delta_i, q_{o,i}, Q_{m,i})$ *$(i = 1, 2)$, we define the* synchronous product

$\mathbf{G} = \mathbf{G_1}||\mathbf{G_2}$ *of the two DES as:*

$$\mathbf{G} := (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{o,1}, q_{o,2}), Q_{m,1} \times Q_{m,2}),$$

where $\delta((q_1, q_2), \sigma)$ *is only defined and equals:*

$$\begin{array}{l}(q_1', q_2') \text{ if } \sigma \in (\Sigma_1 \cap \Sigma_2), \delta_1(q_1, \sigma) = q_1', \delta_2(q_2, \sigma) = q_2' \text{ or} \\ (q_1', q_2) \text{ if } \sigma \in \Sigma_1 - \Sigma_2, \delta_1(q_1, \sigma) = q_1' \text{or} \\ (q_1, q_2') \text{ if } \sigma \in \Sigma_2 - \Sigma_1, \delta_2(q_2, \sigma) = q_2'.\end{array}$$

It follows that $L(\mathbf{G}) = P_1^{-1}L(\mathbf{G_1}) \cap P_2^{-1}L(\mathbf{G_2})$ and $L_m(\mathbf{G}) = P_1^{-1}L_m(\mathbf{G_1}) \cap P_2^{-1}L_m(\mathbf{G_2})$. We note that if $\Sigma_1 = \Sigma_2$, we get $L(\mathbf{G}) = L(\mathbf{G_1}) \cap L(\mathbf{G_2})$ and $L_m(\mathbf{G}) = L_m(\mathbf{G_1}) \cap L_m(\mathbf{G_2})$.

For DES, the two main properties we want to check are *nonblocking* and *controllability.*

Definition 2. *A DES* $\mathbf{G}$ *is said to be* nonblocking *if:*
$(\forall s \in L(\mathbf{G}))\ (\exists s' \in \Sigma^*)\ ss' \in L_m(\mathbf{G})$

For controllability, we assume the standard event partition $\Sigma = \Sigma_u \dot\cup \Sigma_c$, splitting our alphabet into *uncontrollable* and *controllable events.*

Definition 3. *A supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *is* controllable *for plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *if:*

$$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u) s\sigma \in L(\mathbf{G}) \Rightarrow s\sigma \in L(\mathbf{S})$$

We now provide some language definitions that will be useful for this paper. We start with the language L^k. This is the set of strings constructed from any k strings in L.

Definition 4. *Let* $L \subseteq \Sigma^*$ *and* $k \in \{1, 2, 3, \ldots\}$. *We define the* language L^k *to be:*

$$L^k := \{s \in \Sigma^* | s = s_1 s_2 \ldots s_k \text{ for some } s_1, s_2, \ldots, s_k \in L\}$$

We next define the notation for the language constructed from all possible ways to concatenate a string from language L_1, followed by an event from Σ', and a string from language L_2.

Definition 5. *Let* L_1, $L_2 \subseteq \Sigma^*$ *and* $\Sigma' \subseteq \Sigma$. *We define the* language $L_1.\Sigma'.L_2$ *to be:*
$L_1.\Sigma'.L_2 := \{s \in \Sigma^* | s = s_1 \sigma s_2 \text{ for some } s_1 \in L_1, s_2 \in L_2, \sigma \in \Sigma'\}$

3 Fault Tolerant Setting

In this section, we will introduce our concept of fault events and a consistency property that our systems must satisfy. In the following section, we will assume that all DES are deterministic, and that we are given plant $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ and supervisor $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$.

3.1 Fault Events

In this paper, our approach will be to add a set of uncontrollable events to our plant model to represent the possible faults in the system. For example, if we had a sensor to detect when a

train passes, its plant model might originally contain an event such as *trn_sen0* indicating a train is present. We could add a new uncontrollable event, *trnf_sen0*, that will occur instead if the sensor fails to detect the train. This will allow us to model how the system will behave after the occurrence of the fault. Our goal will be to design supervisors that will still behave correctly even if a fault event occurs, even though they can't detect the fault event directly.

We start by defining a group of $m \geq 0$ mutually exclusive sets of *fault events*.

$$\Sigma_{F_i} \subseteq \Sigma_u, i = 1, \ldots, m$$

The idea here is to group related faults into sets such that faults of a given set represent a common fault condition, while faults of a different set represent a different fault condition. For example, two sensors in a row that could each be used to detect the train in time for a given track segment might be in the same fault set, but a sensor in a different part of the track would be in a different set.

Definition 6. *We refer to faults in* Σ_{F_i}, $i = 1, \ldots, m$, *collectively as* standard fault events:

$$\Sigma_F := \dot{\bigcup_{i=1,\ldots,m}} \Sigma_{F_i}$$

We note that for $m = 0$, $\Sigma_F = \emptyset$.

The standard fault events are the faults that will be used to define the various fault scenarios that our supervisors will need to be able to handle. However, there are two additional types of faults that we need to define in order to handle two special cases. The first type is called *unrestricted fault events*, denoted $\Sigma_{\Omega F} \subseteq \Sigma_u$. These are faults that a supervisor can always handle and thus are allowed to occur unrestricted.

The second type is called *excluded fault events*, denoted $\Sigma_{\Delta F} \subseteq \Sigma_u$. These are faults that can not be handled at all and thus are essentially ignored in our scenarios. The idea is that this would allow us to still design a fault tolerant supervisory for the remaining faults. Typically, most systems would have neither excluded or unrestricted faults, but we will include them in our definitions for the systems that do.

For each fault set, Σ_{F_i} $(i = 0, \ldots, m)$, we also need to define a matching set of *reset events*, denoted $\Sigma_{T_i} \subseteq \Sigma$. These events will be explained in Section 3.3, when we describe the resettable fault scenario.

3.2 Fault Tolerant Consistency

We now present a consistency requirement that our systems must satisfy.

Definition 7. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$, *supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$, *and fault sets* $\Sigma_{F_i}\,(i = 0, \ldots, m), \Sigma_{\Delta F}$, *and* $\Sigma_{\Omega F}$, *is* fault tolerant (FT) consistent *if:*

1. $\Sigma_{\Delta F} \cup \Sigma_{\Omega F} \cup \Sigma_F \subseteq \Sigma_u$
2. $\Sigma_{\Delta F}, \Sigma_{\Omega F}, \Sigma_{F_i}$ $(i = 0, \ldots, m)$, *are pair-wise disjoint.*
3. $(\forall i \in 1, \ldots, m)\Sigma_{F_i} \neq \emptyset$

4. $(\forall i \in 1, \ldots, m)\Sigma_{F_i} \cap \Sigma_{T_i} = \emptyset$

5. *Supervisor* $\mathbf{S}$ *is deterministic.*

6. $(\forall x \in X)(\forall \sigma \in (\Sigma_{\Omega F} \cup \Sigma_{\Delta F} \cup \Sigma_F))\xi(x, \sigma) = x$

Point (1) says that fault events are uncontrollable since allowing a supervisor to disable fault events would be unrealistic. **Point (2)** requires that the indicated sets of faults be disjoint since they must each be handled differently. **Point (3)** says that fault sets Σ_{F_i} are non-empty. **Point (4)** says a fault set must be disjoint from its corresponding set of reset events so we can distinguish them.

Points (5) and **(6)** say that $\mathbf{S}$ is deterministic (single initial state and at most a single transition leaving a given state for a given event) and that at every state in $\mathbf{S}$, there is a selfloop for each fault event in the system. This means a supervisor cannot change state (and thus change enablement information) based on a fault event. This is a key concept as it effectively makes fault events unobservable to supervisors. If $\mathbf{S}$ is defined over a subset $\Sigma' \subset \Sigma$ instead, we could equivalently require that Σ' contain no fault events.

3.3 Fault Scenarios

In this paper, we will consider four fault scenarios. The first is the *default fault scenario* where the supervisor must be able to handle any non-excluded fault event that occurs. The second scenario is the $N \geq 0$ *fault scenario* where the supervisor is only required to handle at most N, non-excluded fault events and all unrestricted fault events.

The next scenario is the *non-repeatable* $N \geq 0$ *fault scenario* where the supervisor is only required to handle at most N, non-excluded fault events and all unrestricted fault events, but no more than one fault event from any given Σ_{F_i} $(i = 0, \ldots, m)$ fault set. This definition allows the designer to group faults together in fault sets such that a fault occurring from one set does not affect a supervisors ability to handle a fault from a different set. Particularly for a situation where a supervisor could handle only one fault per fault set, this would allow m faults to occur instead of only one using the previous scenario.

The last scenario we consider is the *resettable fault scenario.* This is designed to capture the situation where at most one fault event from each Σ_{F_i} $(i = 0, \ldots, m)$ fault set can be handled by the supervisor during each pass through a part of the system, but this ability resets for the next pass. For this to work, we need to be able to detect when the current pass has completed and it is safe for another fault event from the same fault set to occur. We use the fault set's corresponding set of reset events to achieve this. The idea is that once a reset event has occurred, the current pass can be considered over and it is safe for another fault event to occur.

4 Fault Tolerant Controllability Definitions

We will now develop some properties that will allow us to determine if a supervisor will still be controllable in the four fault scenarios that we introduced in the previous section.

4.1 Fault Tolerant Controllability

The first fault tolerant property that we introduce is designed to handle the default fault scenario. First, we need to define the *language of excluded faults.* This is the set of all strings that include at least one fault from $\Sigma_{\Delta F}$.

Definition 8. *We define the* language of excluded faults *as:*

$$L_{\Delta F} = \Sigma^*.\Sigma_{\Delta F}.\Sigma^*$$

Definition 9. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$*, supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$*, and fault sets* $\Sigma_{F_i}\,(i = 0, \ldots, m)$ *and* $\Sigma_{\Delta F}$*, is* fault tolerant (FT) controllable *if it is FT consistent and:*

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u)$
$\quad (s\sigma \in L(\mathbf{G})) \wedge (s \notin L_{\Delta F}) \Rightarrow s\sigma \in L(\mathbf{S})$

The above definition is essentially the standard controllability definition but ignores strings that include excluded fault events. As the language $L(\mathbf{S}) \cap L(\mathbf{G})$ is prefix closed, prefixes of these strings that do not contain excluded faults must be checked. This definition is equivalent to blocking all excluded fault events from occurring in the system behavior and then checking the standard controllability definition. This is the most powerful of the fault tolerant definitions as the supervisor must be able to handle a potentially unlimited number of faults that can occur in any order. We note that if $\Sigma_{\Delta F} = \emptyset$, then Definition 9 reduces to the standard controllability definition as $L_{\Delta F}$ reduces to $L_{\Delta F} = \emptyset$.

Typically, the set of excluded faults for a given system is empty. When a system is FT controllable and $\Sigma_{\Delta F} \neq \emptyset$, we say that it is *FT controllable with excluded faults* to emphasize that it is less fault tolerant than if it passed the definition with $\Sigma_{\Delta F} = \emptyset$. We will use a similar expression with the other fault tolerant definitions.

4.2 N-Fault Tolerant Controllability

The next fault tolerant property that we introduce is designed to handle the $N \geq 0$ fault scenario. First, we need to define the *language of N-fault events.* This is the set of all strings that include at most N faults from Σ_F, including those that contain no such faults.

Definition 10. *We define the* language of N-fault events *as:*

$$L_{NF} = (\Sigma - \Sigma_F)^* \cup \bigcup_{k=1}^{N} ((\Sigma - \Sigma_F)^*.\Sigma_F.(\Sigma - \Sigma_F)^*)^k$$

Definition 11. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$*, supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$*, and fault sets* $\Sigma_{F_i}\ (i = 0, \ldots, m)$ *and* $\Sigma_{\Delta F}$*, is* N-fault tolerant (N-FT) controllable *if it is FT consistent and:*

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u)$
$\quad (s\sigma \in L(\mathbf{G})) \wedge (s \notin L_{\Delta F}) \wedge (s \in L_{NF}) \Rightarrow s\sigma \in L(\mathbf{S})$

The above definition is essentially the standard controllability definition but ignores strings that include excluded fault events or more than N faults from fault sets $\Sigma_{F_i}\ (i = 0, \ldots, m)$.

This definition is essentially weaker than the previous one since if we take $N = \infty$ we get the FT controllability definition back. If we set $N = 0$, we get the controllability definition with all fault events from Σ_F excluded as well since L_{NF} will simplify to $L_{NF} = (\Sigma - \Sigma_F)^*$. We also note that if $m = 0$, we get $\Sigma_F = \emptyset$. This means L_{NF} will simplify to $L_{NF} = \Sigma^*$ which means Definition 11 will simplify to Definition 9.

Typically, the set of unrestricted faults for a given system is empty. When a system is N-FT controllable and $\Sigma_{\Omega F} \neq \emptyset$, we say that it is *N-FT controllable with unrestricted faults* to emphasize that it is more fault tolerant than if it passed the definition with $\Sigma_{\Omega F} = \emptyset$. We will use a similar expression with the other fault tolerant definitions.

4.3 Non-repeatable N-Fault Tolerant Controllability

The next fault tolerant property that we introduce is designed to handle the non-repeatable $N \geq 0$ fault scenario. First, we need to define the *language of non-repeatable fault events.* This is the set of all strings that include two or more faults from a single fault set Σ_{F_i} $(i = 0, \ldots, m)$.

Definition 12. *We define the* language of non-repeatable fault events *as:*

$$L_{NRF} = \bigcup_{i=1}^{m} (\Sigma^*.\Sigma_{F_i}.\Sigma^*.\Sigma_{F_i}.\Sigma^*)$$

Definition 13. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$*, supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$*, and fault sets* Σ_{F_i} $(i = 0, \ldots, m)$ *and* $\Sigma_{\Delta F}$*, is* non-repeatable N-fault tolerant (NR-FT) controllable*, if it is FT consistent and:*

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u)$
$\quad (s\sigma \in L(\mathbf{G})) \wedge (s \notin L_{\Delta F} \cup L_{NRF}) \wedge (s \in L_{NF}) \Rightarrow s\sigma \in L(\mathbf{S})$

The above definition is essentially the standard controllability definition, but ignores strings that include excluded fault events, more than N faults from fault sets Σ_{F_i} $(i = 0, \ldots, m)$, or strings that include two or more faults from a single fault set. We note that if $m = 0$, we get $\Sigma_F = \emptyset$. This means L_{NF} simplifies to $L_{NF} = \Sigma^*$ and L_{NRF} simplifies to $L_{NRF} = \emptyset$. This means Definition 13 simplifies to Definition 9.

4.4 Resettable Fault Tolerant Controllability

The next fault tolerant property that we introduce is designed to handle the resettable fault scenario. First, we need to define the *language of resettable fault events.* This is the set of all strings where two faults from the same fault set Σ_{F_i} occur in a row without an event from the corresponding set of reset events in between.

Definition 14. *We define the* language of resettable fault events *as:*

$$L_{TF} = \bigcup_{i=1}^{m} (\Sigma^*.\Sigma_{F_i}.(\Sigma - \Sigma_{T_i})^*.\Sigma_{F_i}.\Sigma^*)$$

Definition 15. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$*, supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$*, and fault sets* Σ_{F_i} $(i = 0, \ldots, m)$ *and* $\Sigma_{\Delta F}$*, is* resettable fault tolerant (T-FT) controllable *if*

it is FT consistent and:

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u)$
$(s\sigma \in L(\mathbf{G})) \wedge (s \notin L_{\Delta F} \cup L_{TF}) \Rightarrow s\sigma \in L(\mathbf{S})$

The above definition is essentially the standard controllability definition, but ignores strings that include excluded fault events and strings where we get two fault events from the same fault set in a row without an event from the corresponding set of reset events in between. We note that if $m = 0$, we get $\Sigma_F = \emptyset$. This means L_{TF} simplifies to $L_{TF} = \emptyset$ which means Definition 15 simplifies to Definition 9.

5 Fault Tolerant Nonblocking Definitions

We will now develop some properties that will allow us to determine if a system will still be nonblocking in the four scenarios that we introduced in Section 3.3.

We use the fault languages from Section 4 and a similar approach to add fault tolerant principles to the standard nonblocking definition.

5.1 Fault Tolerant Nonblocking

The first fault tolerant nonblocking property that we introduce is designed to handle the default fault scenario. We use the language of excluded faults from Section 4.1.

Definition 16. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$*, supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$*, and fault sets* $\Sigma_{F_i}\ (i = 0, \ldots, m)$ *and* $\Sigma_{\Delta F}$*, is* fault tolerant (FT) nonblocking *if it is FT consistent and:*

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))$
$(s \notin L_{\Delta F}) \Rightarrow (\exists s' \in \Sigma^*)(ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})) \wedge (ss' \notin L_{\Delta F})$

We note that if $\Sigma_{\Delta F} = \emptyset$, then Definition 16 reduces to the standard nonblocking definition. Also, if $m = 0$ then Definitions 17, 18, and 19 all simplify to Definition 16.

5.2 N-Fault Tolerant Nonblocking

The next fault tolerant nonblocking property that we introduce is designed to handle the $N \geq 0$ fault scenario. We use the language of excluded faults and the language of N-fault events from Sections 4.1 and 4.2.

Definition 17. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$*, supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$*, and fault sets* $\Sigma_{F_i}\ (i = 0, \ldots, m)$ *and* $\Sigma_{\Delta F}$*, is* N-fault tolerant (N-FT) nonblocking *if it is FT consistent and:*

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))\,(s \notin L_{\Delta F}) \wedge (s \in L_{NF}) \Rightarrow$
$(\exists s' \in \Sigma^*)(ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})) \wedge (ss' \notin L_{\Delta F}) \wedge (ss' \in L_{NF})$

5.3 Non-repeatable N-Fault Tolerant Nonblocking

The next fault tolerant nonblocking property that we introduce is designed to handle the non-repeatable $N \geq 0$ fault scenario. We use the language of excluded faults, the language of

N-fault events, and the language of non-repeatable fault events from Section 4.

Definition 18. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$*, supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$*, and fault sets* Σ_{F_i} $(i = 0, \ldots, m)$ *and* $\Sigma_{\Delta F}$*, is* non-repeatable N-fault tolerant (NR-FT) nonblocking*, if it is FT consistent and:*

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))\,(s \notin L_{\Delta F} \cup L_{NRF}) \wedge (s \in L_{NF}) \Rightarrow$
$\quad (\exists s' \in \Sigma^*)(ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})) \wedge (ss' \notin L_{\Delta F} \cup L_{NRF}) \wedge (ss' \in L_{NF})$

5.4 Resettable Fault Tolerant Nonblocking

The next fault tolerant nonblocking property that we introduce is designed to handle the resettable fault scenario. We use the language of excluded faults and the language of resettable fault events from Section 4.

Definition 19. *A system, with plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$*, supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$*, and fault sets* Σ_{F_i} $(i = 0, \ldots, m)$ *and* $\Sigma_{\Delta F}$*, is* resettable fault tolerant (T-FT) nonblocking *if it is FT consistent and:*

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))\,(s \notin L_{\Delta F} \cup L_{TF}) \Rightarrow$
$\quad (\exists s' \in \Sigma^*)(ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})) \wedge (ss' \notin L_{\Delta F} \cup L_{TF})$

6 Algorithms

In this section, we will present algorithms to construct and verify the eight fault tolerant controllability and nonblocking properties that we defined in Sections 4 and 5. We will not present an algorithm for the FT consistency property as its individual points can easily be checked by adapting various standard algorithms. We assume that our system consists of a plant $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$, supervisor $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$, and fault and reset sets Σ_{F_i}, Σ_{T_i} $(i = 0, \ldots, m)$, $\Sigma_{\Delta F}$, and $\Sigma_{\Omega F}$.

Our approach will be to construct plant components to synchronize with our plant $\mathbf{G}$ such that the new DES will restrict the occurrence of faults to match the given fault tolerant controllability and nonblocking definitions. We can then synchronize the plant components together and then use a standard controllability or nonblocking algorithm to check the property. This approach allows us to automatically take advantage of existing scalability methods such as incremental [9] and binary decision diagram-based (BDD) algorithms [10, 11, 12, 13, 14, 15].

As the controllability, nonblocking, and synchronous product algorithms have already been studied in the literature [16], we will assume that they are given to us. We will use the standard || symbol to indicate the synchronous product operation, *vCont*(**Plant**,**Sup**) to indicate controllability verification, and *vNonb*(**System**) to indicate nonblocking verification. Functions *vCont* and *vNonb* return *true* or *false* to indicate whether the verification passed or failed, and the result will be stored in the Boolean variable *pass*.

In the sections that follow, we will first present algorithms to construct the new plant components that will be shared by the fault tolerant controllable and nonblocking algorithms. We then present the individual fault tolerant controllability and nonblocking algorithms.

6.1 Algorithms to Construct Plants

Algorithms $1-4$ construct the needed plant components for the various fault tolerant algorithms.

6.1.1 Construct Excluded Faults Plant

Algorithm 1 constructs $\mathbf{G_{\Delta F}}$ for fault set $\Sigma_{\Delta F}$. The algorithm constructs a new DES with event set $\Sigma_{\Delta F}$, but no transitions. It also contains only its initial state, which is marked. This will have the effect of removing any $\Sigma_{\Delta F}$ transitions from any DES it is synchronized with.

Please note that all of the constructed DES in these algorithms have every state marked since their goal is to modify the closed behavior by restricting the occurrence of fault events as needed; not to modify the marked behavior of the system directly. Also, when we define our transition functions such as δ, we will define them as a subset of $Y \times \Sigma \times Y$ for convenience. For example, $(y_o, \sigma, y_1) \in \delta$ implies $\delta(y_o, \sigma) = y_1$.

Algorithm 1 construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$

1: $Y_1 \leftarrow \{y_0\}$
2: $Y_{m,1} \leftarrow Y_1$
3: $\delta_1 \leftarrow \emptyset$
4: **return** $(Y_1, \Sigma_{\Delta F}, \delta_1, y_o, Y_{m,1})$

Figure 1 shows an example $\mathbf{G_{\Delta F}}$. In the DES diagrams, circles represent unmarked states, while filled circles represent marked states. Two concentric, unfilled circles represent the initial state. If the initial state is also marked, the inner circle is filled. Note that if a transition is labeled by an event set such as in Figure 2, this is a shorthand for a transition for each event in the event set.

Figure 1: Excluded Faults Plant $\mathbf{G_{\Delta F}}$

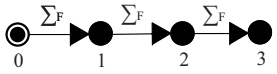

Figure 2: N-Fault Plant $\mathbf{G_{NF}}$, N = 3

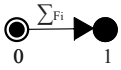

Figure 3: Non-Repeatable N-Fault Plant $\mathbf{G_{F,i}}$

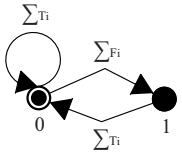

Figure 4: Resettable Fault Plant $\mathbf{G_{TF,i}}$

6.1.2 Construct N-Faults Plant

Algorithm 2 constructs $\mathbf{G_{NF}}$ for max N faults, and standard fault set Σ_F. The algorithm constructs a new DES with event set Σ_F and N states, each state marked. It then creates a transition for each fault event in Σ_F from state y_i to state y_{i+1}. As there are no transitions at

state y_N, synchronizing with this DES will allow at most N faults to occur, and then remove any additional standard fault transitions. Figure 2 shows an example $\mathbf{G_{NF}}$ for $N = 3$.

Algorithm 2 construct-$\mathbf{G_{NF}}(N, \Sigma_F)$

1: $Y_1 \leftarrow \{y_0, y_1, \ldots, y_N\}$
2: $Y_{m,1} \leftarrow Y_1$
3: $\delta_1 \leftarrow \emptyset$
4: **for** $i = 0, \ldots, N-1$
5: **for** $\sigma \in \Sigma_F$
6: $\delta_1 \leftarrow \delta_1 \cup \{(y_i, \sigma, y_{i+1})\}$
7: **end for**
8: **end for**
9: **return** $(Y_1, \Sigma_F, \delta_1, y_o, Y_{m,1})$

We note that if $m = 0$, then $\Sigma_F = \emptyset$. This means that $\mathbf{G_{NF}}$ will contain no events and have unreachable states for $N \geq 1$. As a result, synchronizing with $\mathbf{G_{NF}}$ will have no effect on the closed and marked language of the system. This means that Algorithms 6, 7, 10, and 11 will still work correctly.

We next note that if $N = 0$, $\mathbf{G_{NF}}$ will contain a single state, but no transitions. This will have the desired effect of removing any Σ_F transitions from any DES synchronized with $\mathbf{G_{NF}}$.

6.1.3 Construct Non-repeatable N-Faults Plant

Algorithm 3 constructs $\mathbf{G_{F,i}}$ for $i \in \{1, \ldots, m\}$ and fault set Σ_{F_i}. The algorithm constructs a new DES with event set Σ_{F_i} and two states, both states marked. It then creates a transition for each fault event in Σ_{F_i} from the initial state to state y_1. As there are no transitions at state y_1, synchronizing with this DES will allow at most 1 fault event from the fault set to occur and then remove any additional fault transitions from the fault set. Figure 3 shows an example $\mathbf{G_{F,i}}$.

Algorithm 3 construct-$\mathbf{G_{F,i}}(\Sigma_{F_i}, i)$

1: $Y_i \leftarrow \{y_0, y_1\}$
2: $Y_{m,i} \leftarrow Y_i$
3: $\delta_i \leftarrow \emptyset$
4: **for** $\sigma \in \Sigma_{F_i}$
5: $\delta_i \leftarrow \delta_i \cup \{(y_0, \sigma, y_1)\}$
6: **end for**
7: **return** $(Y_i, \Sigma_{F_i}, \delta_i, y_o, Y_{m,i})$

6.1.4 Construct Resettable Faults Plant

Algorithm 4 constructs $\mathbf{G_{TF,i}}$ for $i \in \{1, \ldots, m\}$, fault set Σ_{F_i}, and reset set Σ_{T_i}. The algorithm constructs a new DES with event set $\Sigma_{F_i} \cup \Sigma_{T_i}$ and two states, both states marked. It then creates a transition for each fault event in Σ_{F_i} from the initial state to state y_1. Next, it creates a transition for each reset event in Σ_{T_i} from state y_1 to the initial state, as well as a

selfloop at the initial state for the event. Figure 4 shows an example $\mathbf{G_{TF,i}}$. Essentially, reset events can occur unrestricted, but once a fault event occurs from Σ_{F_i}, a second event from the set is blocked until a reset event from Σ_{T_i} occurs. Synchronizing with this DES will have the effect of restricting the plant's fault behavior to that which the supervisor is required to handle.

Algorithm 4 construct-$\mathbf{G_{TF,i}}(\Sigma_{F_i}, \Sigma_{T_i}, i)$

1: $Y_i \leftarrow \{y_0, y_1\}$
2: $Y_{m,i} \leftarrow Y_i$
3: $\delta_i \leftarrow \emptyset$
4: **for** $\sigma \in \Sigma_{F_i}$
5: $\quad \delta_i \leftarrow \delta_i \cup \{(y_0, \sigma, y_1)\}$
6: **end for**
7: **for** $\sigma \in \Sigma_{T_i}$
8: $\quad \delta_i \leftarrow \delta_i \cup \{(y_0, \sigma, y_0), (y_1, \sigma, y_0)\}$
9: **end for**
10: **return** $(Y_i, \Sigma_{F_i} \cup \Sigma_{T_i}, \delta_i, y_o, Y_{m,i})$

6.2 Verify Fault Tolerant Controllability

Algorithm 5 shows how to verify fault tolerant controllability for $\mathbf{G}$ and $\mathbf{S}$. Line 1 constructs the *excluded fault plant*, $\mathbf{G_{\Delta F}}$, using Algorithm 1. Line 2 constructs the new plant $\mathbf{G}'$. Line 3 checks that supervisor $\mathbf{S}$ is controllable for plant $\mathbf{G}'$. As $\mathbf{G_{\Delta F}}$ is defined over event set $\Sigma_{\Delta F}$ and contains only a marked initial state and no transitions, synchronizing it with $\mathbf{G}$ creates the original behavior with all excluded fault events removed. Checking that $\mathbf{S}$ is controllable for the resulting behavior will have the effect of verifying fault tolerant controllability.

Algorithm 5 Verify fault tolerant controllability

1: $\mathbf{G_{\Delta F}} \leftarrow$ construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$
2: $\mathbf{G}' \leftarrow \mathbf{G}||\mathbf{G_{\Delta F}}$
3: pass $\leftarrow$ vCont$(\mathbf{G}', \mathbf{S})$
4: **return** pass

We note that if $\Sigma_{\Delta F} = \emptyset$, Algorithm 5 will still produce the correct result. However, it would be more efficient to just check that $\mathbf{S}$ is controllable for $\mathbf{G}$ directly.

6.3 Verify N-Fault Tolerant Controllability

Algorithm 6 shows how to verify N-fault tolerant controllability for $\mathbf{G}$, and $\mathbf{S}$. Line 1 constructs the excluded fault plant, $\mathbf{G_{\Delta F}}$. Line 2 constructs the *N-fault plant*, $\mathbf{G_{NF}}$, using Algorithm 2. Line 3 constructs the new plant $\mathbf{G}'$. Line 4 checks that supervisor $\mathbf{S}$ is controllable for plant $\mathbf{G}'$. As $\mathbf{G_{\Delta F}}$ removes any excluded fault transitions and $\mathbf{G_{NF}}$ prevents strings from containing more than N fault events, checking that $\mathbf{S}$ is controllable for the resulting behavior will have the effect of verifying N-fault tolerant controllability.

Algorithm 6 Verify N-fault tolerant controllability

1: $\mathbf{G_{\Delta F}} \leftarrow$ construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$
2: $\mathbf{G_{NF}} \leftarrow$ construct-$\mathbf{G_{NF}}(N, \Sigma_F)$
3: $\mathbf{G'} \leftarrow \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}$
4: pass $\leftarrow$ vCont$(\mathbf{G'}, \mathbf{S})$
5: **return** pass

We note that if $m = 0$, we have $\Sigma_F = \emptyset$ and that synchronizing with $\mathbf{G_{NF}}$ will have no effect. We will still get the correct result but it would be more efficient to run Algorithm 5 directly instead.

6.4 Verify Non-repeatable N-Fault Tolerant Controllability

Algorithm 7 shows how to verify non-repeatable N-fault tolerant controllability for $\mathbf{G}$ and $\mathbf{S}$. Line 1 constructs the excluded fault plant, $\mathbf{G_{\Delta F}}$. Line 2 constructs the N-fault plant, $\mathbf{G_{NF}}$. For $i \in \{1, \ldots, m\}$, Line 4 constructs the *non-repeatable N-fault plant*, $\mathbf{G_{F,i}}$, using Algorithm 3. Line 6 constructs the new plant $\mathbf{G'}$. Line 7 checks that supervisor $\mathbf{S}$ is controllable for plant $\mathbf{G'}$. As $\mathbf{G_{\Delta F}}$ removes any excluded fault transitions, $\mathbf{G_{NF}}$ prevents strings from containing more than N fault events, and each $\mathbf{G_{F,i}}$ allows at most one fault from their fault set to occur, checking that $\mathbf{S}$ is controllable for the resulting behavior will have the effect of verifying non-repeatable N-fault tolerant controllability. We note that if $m \leq N$, we can safely skip Line 2 (and remove $\mathbf{G_{NF}}$ from line 6) as Lines 3-5 will ensure at most m faults can occur.

Algorithm 7 Verify non-repeatable N-fault tolerant controllability

1: $\mathbf{G_{\Delta F}} \leftarrow$ construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$
2: $\mathbf{G_{NF}} \leftarrow$ construct-$\mathbf{G_{NF}}(N, \Sigma_F)$
3: **for** $i = 1, \ldots, m$
4: $\quad \mathbf{G_{F,i}} \leftarrow$ construct-$\mathbf{G_{F,i}}(\Sigma_{F_i}, i)$
5: **end for**
6: $\mathbf{G'} \leftarrow \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{G_{F,1}}|| \ldots ||\mathbf{G_{F,m}}$
7: pass $\leftarrow$ vCont$(\mathbf{G'}, \mathbf{S})$
8: **return** pass

We note that if $m = 0$, we have $\Sigma_F = \emptyset$, that no $\mathbf{G_{F,i}}$ will be constructed, and that synchronizing with $\mathbf{G_{NF}}$ will have no effect. This means $\mathbf{G'}$ will simplify to $\mathbf{G'} = \mathbf{G}||\mathbf{G_{\Delta F}}$ and we can just evaluate Algorithm 5 instead.

6.5 Verify Resettable Fault Tolerant Controllability

Algorithm 8 shows how to verify resettable fault tolerant controllability for $\mathbf{G}$ and $\mathbf{S}$. Line 1 constructs the excluded fault plant, $\mathbf{G_{\Delta F}}$. For $i \in \{1, \ldots, m\}$, Line 3 constructs the *resettable fault plant* $\mathbf{G_{TF,i}}$, using Algorithm 4. Line 5 constructs the new plant $\mathbf{G'}$. Line 6 checks that supervisor $\mathbf{S}$ is controllable for plant $\mathbf{G'}$. As $\mathbf{G_{\Delta F}}$ removes any excluded fault transitions, and each $\mathbf{G_{TF,i}}$ only allows strings where fault events from Σ_{F_i} are always separated by at least one event from the corresponding set of reset events, Σ_{T_i}, checking that $\mathbf{S}$ is controllable for the resulting behavior will have the effect of verifying resettable fault tolerant controllability.

Algorithm 8 Verify resettable fault tolerant controllability

1: $\mathbf{G_{\Delta F}} \leftarrow$ construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$
2: **for** $i = 1, \ldots, m$
3: $\quad \mathbf{G_{TF,i}} \leftarrow$ construct-$\mathbf{G_{TF,i}}(\Sigma_{F_i}, \Sigma_{T_i}, i)$
4: **end for**
5: $\mathbf{G}' \leftarrow \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{TF,1}}|| \ldots ||\mathbf{G_{TF,m}}$
6: pass $\leftarrow$ vCont$(\mathbf{G}', \mathbf{S})$
7: **return** pass

We note that if $m = 0$, we have $\Sigma_F = \emptyset$ and that no $\mathbf{G_{TF,i}}$ will be constructed. This means $\mathbf{G}'$ will simplify to $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}$ and we can just evaluate Algorithm 5 instead.

6.6 Verify Fault Tolerant Nonblocking

Algorithm 9 shows how to verify fault tolerant nonblocking for $\mathbf{G}$ and $\mathbf{S}$. This algorithm is essentially the same as Algorithm 5, except at Line 2 we calculate the closed loop system $\mathbf{G}'$, and then at Line 3 we verify that it is nonblocking.

Algorithm 9 Verify fault tolerant nonblocking

1: $\mathbf{G_{\Delta F}} \leftarrow$ construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$
2: $\mathbf{G}' \leftarrow \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{S}$
3: pass $\leftarrow$ vNonb$(\mathbf{G}')$
4: **return** pass

We note that if $\Sigma_{\Delta F} = \emptyset$, Algorithm 9 will still produce the correct result. However, it would be more efficient to just check that $\mathbf{S}||\mathbf{G}$ is nonblocking directly.

6.7 Verify N-Fault Tolerant Nonblocking

Algorithm 10 shows how to verify N-fault tolerant nonblocking for $\mathbf{G}$, and $\mathbf{S}$. This algorithm is essentially the same as Algorithm 6, except at Line 3 we calculate the closed loop system $\mathbf{G}'$, and then at Line 4 we verify that it is nonblocking.

Algorithm 10 Verify N-fault tolerant nonblocking

1: $\mathbf{G_{\Delta F}} \leftarrow$ construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$
2: $\mathbf{G_{NF}} \leftarrow$ construct-$\mathbf{G_{NF}}(N, \Sigma_F)$
3: $\mathbf{G}' \leftarrow \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{S}$
4: pass $\leftarrow$ vNonb$(\mathbf{G}')$
5: **return** pass

We note that if $m = 0$, we have $\Sigma_F = \emptyset$ and that synchronizing with $\mathbf{G_{NF}}$ will have no effect. We will still get the correct result but it would be more efficient to run Algorithm 9 directly instead.

6.8 Verify Non-repeatable N-Fault Tolerant Nonblocking

Algorithm 11 shows how to verify non-repeatable N-fault tolerant nonblocking for $\mathbf{G}$ and $\mathbf{S}$. This algorithm is essentially the same as Algorithm 7, except at Line 6 we calculate the closed loop system $\mathbf{G}'$, and then at Line 7 we verify that it is nonblocking.

Algorithm 11 Verify non-repeatable N-fault tolerant nonblocking

1: $\mathbf{G_{\Delta F}} \leftarrow$ construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$
2: $\mathbf{G_{NF}} \leftarrow$ construct-$\mathbf{G_{NF}}(N, \Sigma_F)$
3: **for** $i = 1, \ldots, m$
4: $\quad \mathbf{G_{F,i}} \leftarrow$ construct-$\mathbf{G_{F,i}}(\Sigma_{F_i}, i)$
5: **end for**
6: $\mathbf{G}' \leftarrow \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{G_{F,1}}||\ldots||\mathbf{G_{F,m}}||\mathbf{S}$
7: pass $\leftarrow$ vNonb($\mathbf{G}'$)
8: **return** pass

We note that if $m = 0$, we have $\Sigma_F = \emptyset$, that no $\mathbf{G_{F,i}}$ will be constructed, and that synchronizing with $\mathbf{G_{NF}}$ will have no effect. This means $\mathbf{G}'$ will simplify to $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{S}$ and we can just evaluate Algorithm 9 instead.

We also note that if $N \geq m$, the $\mathbf{G_{F,i}}$ will ensure that no more than m events occur. We thus do not need to add $\mathbf{G_{NF}}$ to $\mathbf{G}'$, which should make the verification more efficient.

6.9 Verify Resettable Fault Tolerant Nonblocking

Algorithm 12 shows how to verify resettable fault tolerant nonblocking for $\mathbf{G}$ and $\mathbf{S}$. This algorithm is essentially the same as Algorithm 8, except at Line 5 we calculate the closed loop system $\mathbf{G}'$, and then at Line 6 we verify that it is nonblocking.

Algorithm 12 Verify resettable fault tolerant nonblocking

1: $\mathbf{G_{\Delta F}} \leftarrow$ construct-$\mathbf{G_{\Delta F}}(\Sigma_{\Delta F})$
2: **for** $i = 1, \ldots, m$
3: $\quad \mathbf{G_{TF,i}} \leftarrow$ construct-$\mathbf{G_{TF,i}}(\Sigma_{F_i}, \Sigma_{T_i}, i)$
4: **end for**
5: $\mathbf{G}' \leftarrow \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{TF,1}}||\ldots||\mathbf{G_{TF,m}}||\mathbf{S}$
6: pass $\leftarrow$ vNonb($\mathbf{G}'$)
7: **return** pass

We note that if $m = 0$, we have $\Sigma_F = \emptyset$ and that no $\mathbf{G_{TF,i}}$ will be constructed. This means $\mathbf{G}'$ will simplify to $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{S}$ and we can just evaluate Algorithm 9 instead.

6.10 Algorithm Complexity Analysis

In this section, we provide a complexity analysis for the fault tolerant controllability and nonblocking algorithms. In the following subsections, we assume that our system consists of a plant $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$, supervisor $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$, and fault and reset sets Σ_{F_i}, Σ_{T_i} $(i = 0, \ldots, m)$, $\Sigma_{\Delta F}$, and $\Sigma_{\Omega F}$.

We will base our analysis on the complexity analysis from Cassandras et al. [17] that states that both the controllability and nonblocking algorithms have a complexity of $O(|\Sigma||Y||X|)$, where $|\Sigma|$ is the size of the system event set, $|Y|$ is the size of the plant state set, and $|X|$ is the size of the supervisor state set. In the analysis that follows, $|Y_{\Delta F}|$ is the size of the state set for $\mathbf{G_{\Delta F}}$ (constructed by Algorithm 1), and $|Y_{NF}|$ is the size of the state set for $\mathbf{G_{NF}}$ (constructed by Algorithm 2).

We note that each FT algorithm first constructs and adds some additional plant components to the system, and then it runs a standard controllability or nonblocking algorithm on the resulting system. Our approach will be to take the standard algorithm's complexity, and replace the value for the state size of the plant with the worst case state size of $\mathbf{G}$ synchronized with the new plant components. As all fault and reset events already belong to the system event set, this means the size of the system event set does not increase.

In the following analysis, we will ignore the cost of constructing the new plant components as they will be constructed in serial with the controllability or nonblocking verification and should be negligible in comparison. We next note that as the base controllability and nonblocking algorithms have the same complexity, the corresponding fault tolerant versions will also have the same complexity (i.e. the FT controllability algorithm will have the same complexity as the FT nonblocking algorithm). As such, we will only present analysis for the FT controllability algorithms.

6.10.1 FT Controllability Algorithm

For Algorithm 5, we replace our plant DES by $\mathbf{G'} = \mathbf{G}||\mathbf{G_{\Delta F}}$. This gives us a worst case state space of $|Y||Y_{\Delta F}|$ for $\mathbf{G'}$. Substituting this into our base algorithm's complexity for the size of our plant's state set gives $O(|\Sigma||Y||Y_{\Delta F}||X|)$. As $|Y_{\Delta F}| = 1$ by Algorithm 1, it follows that our complexity is $O(|\Sigma||Y||X|)$ which is the same as our base algorithm.

6.10.2 N-FT Controllability Algorithm

For Algorithm 6, we replace our plant DES by $\mathbf{G'} = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}$. This gives us a worst case state space of $|Y||Y_{\Delta F}||Y_{NF}|$ for $\mathbf{G'}$. Substituting this into our base algorithm's complexity gives $O(|\Sigma||Y||Y_{\Delta F}||Y_{NF}||X|)$.

We note that $|Y_{\Delta F}| = 1$ by Algorithm 1, and $|Y_{NF}| = N+1$ by Algorithm 2. Substituting in for these values gives $O((N+1)|\Sigma||Y||X|)$. It thus follows that verifying N-FT controllability increases the complexity of verifying controllability by a factor of $(N+1)$.

6.10.3 Non-repeatable N-FT Controllability Algorithm

For Algorithm 7, we replace our plant DES by $\mathbf{G'} = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{G_{F,1}}||\ldots||\ \mathbf{G_{F,m}}$. This gives us a worst case state space of $|Y||Y_{\Delta F}||Y_{NF}||Y_{F_1}|\ldots|Y_{F_m}|$ for $\mathbf{G'}$, where $|Y_{F_i}|$ is the size of the state set for $\mathbf{G_{F,i}}$ $(i = 0, \ldots, m)$, which is constructed by Algorithm 3. Substituting this into our base algorithm's complexity gives $O(|\Sigma||Y||Y_{\Delta F}||Y_{NF}||Y_{F_1}|\ldots|Y_{F_m}||X|)$.

We note that $|Y_{\Delta F}| = 1$ by Algorithm 1, $|Y_{NF}| = N + 1$ by Algorithm 2, and $|Y_{F_i}| = 2$ $(i = 0, \ldots, m)$ by Algorithm 3. Substituting in for these values gives $O(2^m(N+1)|\Sigma||Y||X|)$. It thus follows that verifying non-repeatable N-FT controllability increases the complexity of verifying controllability by a factor of $2^m(N+1)$.

We next note that if $N \geq m$, which we believe will often be the case, it is not necessary to add $\mathbf{G_{NF}}$ to $\mathbf{G}'$. The complexity then reduces to $O(2^m|\Sigma||Y||X|)$.

6.10.4 Resettable FT Controllability Algorithm

For Algorithm 8, we replace our plant DES by $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{TF,1}}||\ldots||\mathbf{G_{TF,m}}$. This gives us a worst case state space of $|Y||Y_{\Delta F}||Y_{TF_1}|\ldots|Y_{TF_m}|$ for $\mathbf{G}'$, where $|Y_{TF_i}|$ is the size of the state set for $\mathbf{G_{TF,i}}$ $(i = 0, \ldots, m)$, which is constructed by Algorithm 4. Substituting this into our base algorithm's complexity gives $O(|\Sigma||Y||Y_{\Delta F}||Y_{TF_1}|\ldots|Y_{TF_m}||X|)$.

We note that $|Y_{\Delta F}| = 1$ by Algorithm 1, and $|Y_{TF_i}| = 2$ $(i = 0, \ldots, m)$ by Algorithm 4. Substituting in for these values gives $O(2^m|\Sigma||Y||X|)$. It thus follows that verifying resettable FT controllability increases the complexity of verifying controllability by a factor of 2^m.

7 Algorithm Correctness

In this section, we introduce several propositions and theorems that show that the algorithms introduced in Section 6 correctly verify that a fault tolerant consistent system satisfies the specified fault tolerant controllability and nonblocking properties defined in Sections 4 and 5.

7.1 Fault Tolerant Propositions

The propositions in this section will be used to support the fault tolerant controllability theorems in Section 7.2. Fault tolerant controllability definitions are essentially controllability definitions with added restriction that a string s is only tested if it is satisfies the appropriate fault tolerant property. The algorithms are intended to replace the original plant with a new plant $\mathbf{G}'$, such that $\mathbf{G}'$ is restricted to strings with the desired property. Propositions $1-4$ essentially assert that string s belongs to the closed behaviour of $\mathbf{G}'$, if and only if s satisfies properties of fault tolerant controllable, N-FT controllable, non-repeatable N-FT controllable, and resettable FT controllable, respectively. These propositions will also be used in the fault tolerant nonblocking theorems in Section 7.3.

Proposition 1. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent, and let* $\mathbf{G}'$ *be the plant constructed in Algorithm 5. Then:*

$$(\forall s \in L(\mathbf{G}))s \notin L_{\Delta F} \iff s \in L(\mathbf{G}')$$

Proof. See Appendix A. □

Proposition 2. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent,* $N \geq 0$*, and let* $\mathbf{G}'$ *be the plant constructed in Algorithm 6. Then:*

$$(\forall s \in L(\mathbf{G}))(s \notin L_{\Delta F}) \wedge (s \in L_{NF}) \iff s \in L(\mathbf{G}')$$

Proof. See Appendix A. □

Proposition 3. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent,* $N \geq 0$*, and let* $\mathbf{G}'$ *be the plant constructed in Algorithm 7. Then:*

$$(\forall s \in L(\mathbf{G}))(s \notin L_{\Delta F} \cup L_{NRF}) \wedge (s \in L_{NF}) \iff s \in L(\mathbf{G}')$$

Proof. See Appendix A. □

Proposition 4. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent, and let* $\mathbf{G}'$ *be the plant constructed in Algorithm 8. Then:*

$$(\forall s \in L(\mathbf{G}))(s \notin L_{\Delta F} \cup L_{TF}) \iff s \in L(\mathbf{G}')$$

Proof. See Appendix A. □

7.2 Fault Tolerant Controllable Theorems

In this section we present theorems that show the fault tolerant controllable algorithms in Section 6 (Algorithms 5-8) will return *true* if and only if the fault tolerant consistent system satisfies the corresponding fault tolerant controllability property.

Theorem 1. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent, and let* $\mathbf{G}'$ *be the plant constructed in Algorithm 5. Then* $\mathbf{S}$ *is* fault tolerant controllable *for* $\mathbf{G}$ *iff* $\mathbf{S}$ *is controllable for* $\mathbf{G}'$.

Proof. See Appendix B. □

Theorem 2. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent,* $N \geq 0$*, and let* $\mathbf{G}'$ *be the plant constructed in Algorithm 6. Then* $\mathbf{S}$ *is* N-fault tolerant controllable *for* $\mathbf{G}$ *iff* $\mathbf{S}$ *is controllable for* $\mathbf{G}'$.

Proof. See Appendix B. □

Theorem 3. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent,* $N \geq 0$*, and let* $\mathbf{G}'$ *be the plant constructed in Algorithm 7. Then* $\mathbf{S}$ *is* non-repeatable N-fault tolerant controllable *for* $\mathbf{G}$ *iff* $\mathbf{S}$ *is controllable for* $\mathbf{G}'$.

Proof. See Appendix B. □

Theorem 4. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent, and let* $\mathbf{G}'$ *be the plant constructed in Algorithm 8. Then* $\mathbf{S}$ *is* resettable fault tolerant controllable *for* $\mathbf{G}$ *iff* $\mathbf{S}$ *is controllable for* $\mathbf{G}'$.

Proof. See Appendix B. □

7.3 Fault Tolerant Nonblocking Theorems

In this section we present theorems that show the fault tolerant nonblocking algorithms in Section 6 (Algorithms 9-12) will return *true* if and only if the fault tolerant consistent system satisfies the corresponding fault tolerant nonblocking property.

Theorem 5. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent, and let* $\mathbf{G}'$ *be the system constructed in Algorithm 9. Then* $\mathbf{S}$ *and* $\mathbf{G}$ *are fault tolerant nonblocking iff* $\mathbf{G}'$ *is nonblocking.*

Proof. See Appendix B. □

Theorem 6. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent,* $N \geq 0$*, and let* $\mathbf{G}'$ *be the system constructed in Algorithm 10. Then* $\mathbf{S}$ *and* $\mathbf{G}$ *are N-fault tolerant nonblocking iff* $\mathbf{G}'$ *is nonblocking.*

Proof. See Appendix B. □

Theorem 7. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent,* $N \geq 0$*, and let* $\mathbf{G}'$ *be the system constructed in Algorithm 11. Then* $\mathbf{S}$ *and* $\mathbf{G}$ *are non-repeatable N- fault tolerant nonblocking iff* $\mathbf{G}'$ *is nonblocking.*

Proof. See Appendix B. □

Theorem 8. *Let system with supervisor* $\mathbf{S} = (X, \Sigma, \xi, x_o, X_m)$ *and plant* $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$ *be FT consistent, and let* $\mathbf{G}'$ *be the system constructed in Algorithm 12. Then* $\mathbf{S}$ *and* $\mathbf{G}$ *are resettable fault tolerant nonblocking iff* $\mathbf{G}'$ *is nonblocking.*

Proof. See Appendix B. □

8 Manufacturing Example

This example is based on the manufacturing testbed from Leduc [18]. The testbed was designed to simulate a manufacturing workcell, in particular problems of routing and collision. Figure 5 shows conceptually the structure of the testbed and sensors.

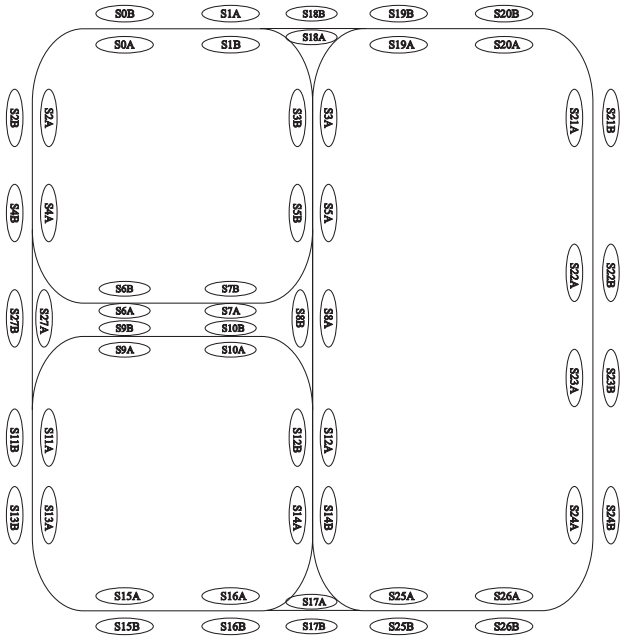

Figure 5: Sensors in the Testbed

In this paper, we will first focus on only a single track loop, shown in Figure 6. The loop contains 8 sensors and two trains (*train 1*, *train 2*). Train 1 starts between sensors 9 and 10, while train 2 starts between sensors 15 and 16. Both trains can only traverse the tracks in a clockwise direction. We will use the simplified version to illustrate our method. We will then report experimental results of applying the method to the full testbed model in Section 8.3.

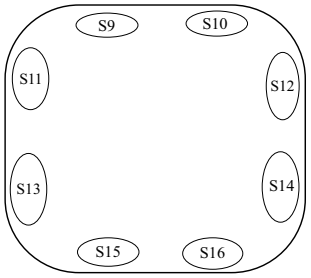

Figure 6: Single Train Loop

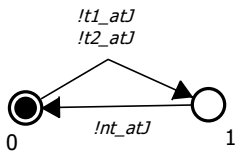

Figure 7: Original Sensor Model

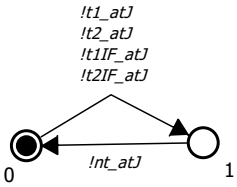

Figure 8: Sensors 9, 10, and 16 with Faults

8.1 Base Plant Models

The plant model for the portion of the testbed we are currently considering consists of the following basic elements: sensors, trains and the relationship between sensors and trains.

8.1.1 Sensor Models

The sensor models indicate when a given train is present, and when no trains are present. Also, they state that only one train can activate a given sensor at a time. Figure 7 shows the original sensor model, for sensor $J \in \{9, \ldots, 16\}$.

To add faults to the model, we assumed that sensors 9 10, and 16 could have an intermittent fault; sometimes the sensor would detect the presence of a train, sometimes it would fail to do so. We modelled this by adding to all the plant models a new event $t1F\_atJ$, $J \in \{9, 10, 16\}$, for each $t1\_atJ$ event. For each $t1\_atJ$ transition in a plant model, we added an identical $t1F\_atJ$ transition. The idea is we can now get the original detection event or the new fault one instead. We made similar changes for train 2. Figure 8 shows the new sensor models with the added fault events. All other sensors will use the original version shown in Figure 7.

For this example, $\Sigma_{\Delta F} = \Sigma_{\Omega F} = \emptyset$. We also set $m = 4$, $\Sigma_{F_1} = \{t1F\_at9, t1F\_at10\}$, $\Sigma_{F_2} = \{t1F\_at16\}$, $\Sigma_{F_3} = \{t2F\_at9, t2F\_at10\}$, $\Sigma_{F_4} = \{t2F\_at16\}$, $\Sigma_{T_1} = \{t1\_at11\}$, $\Sigma_{T_2} = \{t1\_at14\}$, $\Sigma_{T_3} = \{t2\_at11\}$, and $\Sigma_{T_4} = \{t2\_at14\}$.

8.1.2 Sensor Interdependencies

This series of models show the sensor's interdependencies with respect to a given train. With respect to the starting position of a particular train (represented by the initial state), sensors can only be reached in a particular order, dictated by their physical location on the track. This is shown in Figures 9 and 10. Both DES already show the added fault events.

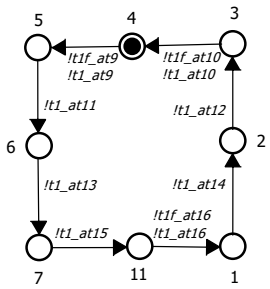

Figure 9: Sensor Interdependencies For Train 1

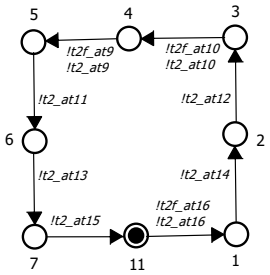

Figure 10: Sensor Interdependencies For Train 2

8.1.3 Train Models

The train models are shown in Figure 11 for train K ($K = 1, 2$). Train K can only move when its enablement event *en_ trainK* occurs, and then it can move at most a single unit of distance (event *umv_ trainK*), before another *en_ trainK* must occur. This allows a supervisor to precisely control the movement of the train by enabling and disabling event *en_ trainK* as needed.

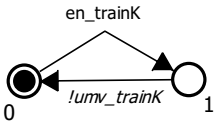

Figure 11: Train K Model

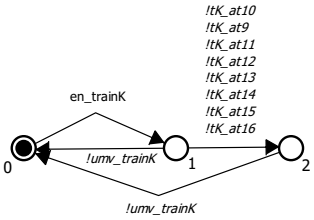

Figure 12: Sensors and Train K

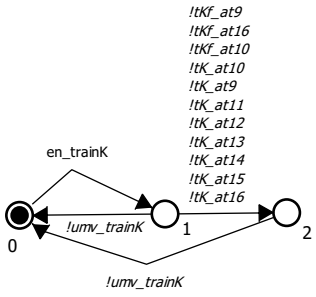

Figure 13: Sensors and Train K with Faults

8.1.4 Relationship Between Sensors and Trains Models

Figure 12 shows the relationship between train K's ($K = 1, 2$) movement, and a sensor detecting the train. It captures the idea that a train can reach at most one sensor during a unit movement, and no sensors if it is disabled. Figure 13 shows the replacement model with fault events added. We now seen that our plant model contains 14 DES in total.

8.2 Modular Supervisors

After the plant models were developed, four supervisors were designed to prevent collisions in the track sections with sensors 11-13, 15-16, 12-14, and 9-10. The idea is to ensure that only one train uses this track section at a time. We will first introduce the original collision protection supervisors that were designed with the assumption of no faults, and then we will introduce new fault tolerant versions with added redundancy.

8.2.1 Collision Protection Supervisors

Figure 14 shows the collision protection supervisor (**CPS-11-13**) for the track section containing sensors 11 and 13. Once a train has reached sensor 11, the other train is stopped at sensor 10 until the first train reaches sensor 15, which indicates it has left the protected area. The stopped train is then allowed to continue. Figures 15, 16, and 17 show similar supervisors for the remaining track sections. Supervisors **CPS-15-16** and **CPS-9-10** have nonstandard initial states in order to reflect the starting locations of the two trains.

It's easy to see that supervisor **CPS-11-13** will not be fault tolerant as it relies solely on sensor 10 to detect when a second train arrives. If sensor 10 fails, the train continues and could collide with the first train. Supervisors **CPS-9-10** and **CPS-12-14** will also not be fault tolerant because of sensor 10. A failure at sensor 10 could cause supervisor **CPS-9-10** to miss a train entering the protected zone, and could cause supervisor **CPS-12-14** to miss a train leaving the protected zone. Using the DES research software tool, DESpot [19], we verified that the system passes $N = 0$ FT controllability and nonblocking (i.e. if all faults are ignored) and fails all eight fault tolerant controllability and nonblocking properties ($N \geq 1$).

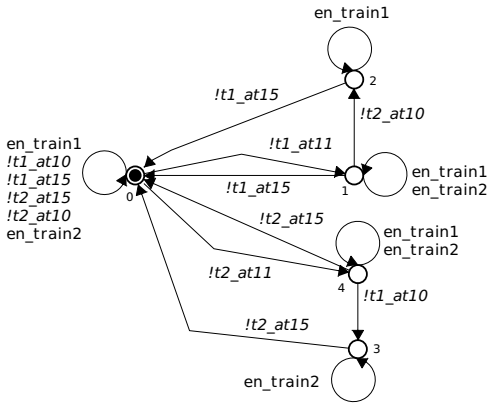

Figure 14: **CPS-11-13** Supervisor

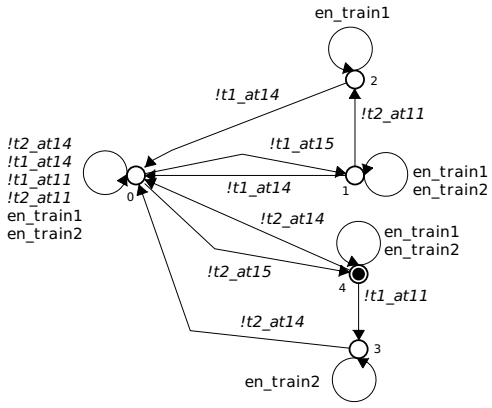

Figure 15: **CPS-15-16** Supervisor

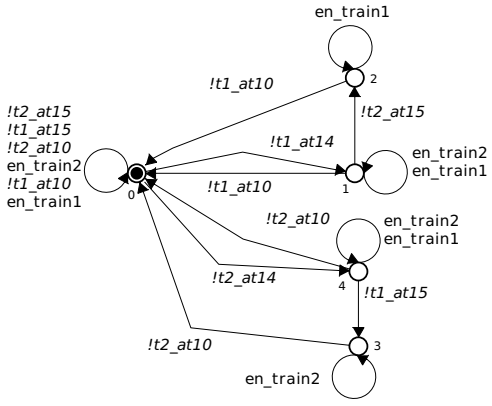

Figure 16: **CPS-12-14** Supervisor

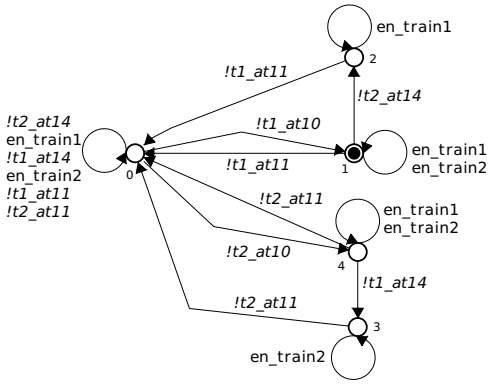

Figure 17: **CPS-9-10** Supervisor

8.2.2 Collision Protection Fault Tolerant Supervisors

We next modified supervisor **CPS-11-13** to make it more fault tolerant. The result is shown in Figure 18. We have added at states 1 and 4 a check for either sensor 9 or sensor 10. That way if sensor 10 fails but sensor 9 doesn't, we can still stop the train at sensor 9 and avoid the collision. We made similar changes to supervisors **CPS-12-14**, and **CPS-9-10**, as shown in Figures 19, and 20. Supervisor **CPS-15-16** did not require any changes as it did not rely on any of the sensors that had faults.

Using DESpot, we can verify that the supervisor is not fault tolerant controllable or nonblocking for the plant. The reason is that if both sensors 9 and 10 fail, the train will not be detected. However, the system can be show to be N-fault tolerant controllable for $N = 1$ (i.e. sensor 10 fails but not sensor 9), non-repeatable N-fault tolerant controllable for $N = 4$, and resettable fault tolerant controllable (as long as both sensors 9 and 10 don't fail in a given pass, all is well). The system also passes the corresponding FT nonblocking properties. It can also be shown that the system fails N-fault tolerant controllable and nonblocking for $N = 2$.

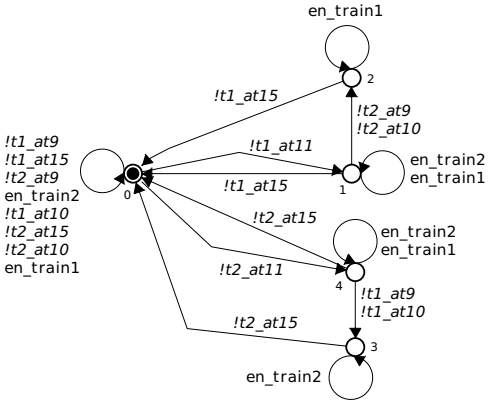

Figure 18: **CPS-11-13FT** Supervisor

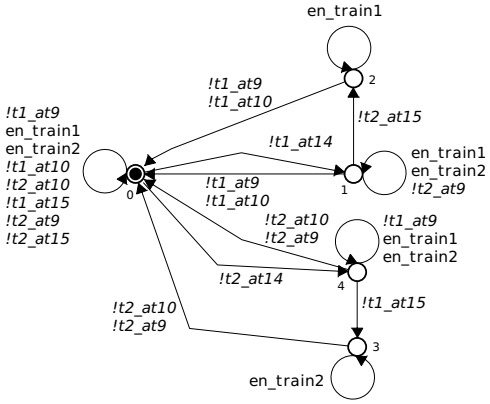

Figure 19: **CPS-12-14FT** Supervisor

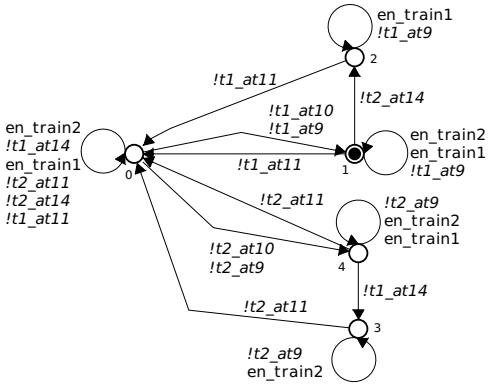

Figure 20: **CPS-9-10FT** Supervisor

8.3 Complete System

We next considered the full plant model for the testbed, as described in Leduc [18]. This model includes all three loops shown in Figure 5, including all of the sensors shown, as well

as six switches for routing, and three cranes, located at sensors 2, 13, and 21, for loading the trains. The full model includes collision protection supervisors for all track sections as well as supervisors for routing trains and stopping each train for loading when they reach a crane. The original system contains 29 supervisors, 110 plant components and has a state space of 7.33×10^9 states.

For this system, we used a similar approach to the one described earlier to add fault events to sensors, and to add fault tolerance to the supervisors. See Dierikx [20] for complete details. For this version of the example, we have $\Sigma_{\Omega F} = \emptyset$ and $\Sigma_{\Delta F} = \cup_{K=1,2}(\cup_{j \in I_\Delta}\{tKF\_atj\})$, where $I_\Delta = \{2, 8, 13, 21, 27\}$. The excluded faults are for key portions of the track where a decision (such as stopping a train in front of a given crane) needs to be made but there does not exist a second physical sensor appropriately located that can be used as a backup. To deal with faults from these sensors, we believe we would need to add additional sensors.

For fault and reset sets, we have $m = 16$. For train 1, we have fault sets $\Sigma_{F_n} = \cup_{j \in I_{Fn}}\{tF1\_atj\}$, $n = 1, \ldots, 8$, where $I_{F1} = \{0, 1, 4\}$, $I_{F2} = \{3, 5, 6, 7\}$, $I_{F3} = \{9, 10, 11\}$, $I_{F4} = \{12, 14\}$, $I_{F5} = \{15, 16\}$, $I_{F6} = \{19, 20, 22\}$, $I_{F7} = \{23, 24\}$, and $I_{F8} = \{25, 26\}$. Sets $\Sigma_{F_9} - \Sigma_{F_{16}}$ are analogous, except that they are for train 2.

For train 1, we have reset sets $\Sigma_{T_n} = \cup_{j \in I_{Rn}}\{t1\_atj\}$, $n = 1, \ldots, 8$, where $I_{R1} = \{6, 7, 27\}$, $I_{R2} = \{0, 1, 19, 20\}$, $I_{R3} = \{15, 16\}$, $I_{R4} = \{8, 9, 10\}$, $I_{R5} = \{12, 14\}$, $I_{R6} = \{23, 24\}$, $I_{R7} = \{25, 26\}$, and $I_{R8} = \{12, 14\}$. Sets $\Sigma_{T_9} - \Sigma_{T_{16}}$ are analogous, except that they are for train 2.

Using our software research tool, DESpot [19], we were able to determine that the system is N-FT controllable and nonblocking (N = 1), non-repeatable N-FT controllable and nonblocking (N = 16), and resettable FT controllable and nonblocking. We ran an FT controllable check on the system but after 33 hours and 1.908×10^9 states and counting, we stopped the computation. See Table 1 for verification times and project state sizes (includes added FT plant components).

We also ran N-FT controllability and nonblocking checks for N = 2. The system passed for controllability and failed for nonblocking. The reason that it passed N-FT controllability is that a switch failed to change state due to a sensor fault and a train derailed taking it to a noncoreachable state before an illegal event could occur. This suggests that the routing supervisors could be made more expressive by adding the uncontrollable train derailing events to their event sets, but without matching transitions.

Table 1: Verification Times for Full System

		Verification Time (seconds)			
Property	**State Size**	**Controllability**		**Nonblocking**	
fault tolerant	1.908×10^9+	-		-	
N-fault tolerant ($N = 1$)	368,548	654	P	3178	P
N-fault tolerant ($N = 2$)	1.961×10^6	13,916	P	26,249	F
nonrepeatable N-FT	1.275×10^{10}	4,230	P	10,956	P
resettable FT	594,448	2,007	P	7,645	P

9 Conclusions and Future Work

In this paper we investigate the problem of fault tolerance (FT) in the framework of discrete-event systems. We introduce a set of eight fault tolerant controllability and nonblocking definitions designed to capture different types of fault scenarios and to ensure that our system remains controllable and nonblocking in each scenario. This approach is different from the typical fault tolerant methodology as the approach does not rely on detecting faults and switching to a new supervisor; it requires a supervisor to work correctly under normal and fault conditions.

We then present a set of algorithms to verify the fault tolerant properties. As these algorithms involve adding new plant components and then checking standard controllability and nonblocking properties, they can instantly take advantage of existing controllability and nonblocking software, as well as scalability approaches such as incremental verification and binary decision diagrams (BDD).

For each algorithm, we provide a complexity analysis showing that the FT algorithms multiply the complexity of the standard algorithms by a factor of one, $N+1$ (N is the number of allowed faults), 2^m (m is the number of fault sets) and $2^m(N+1)$. We then prove the correctness of the algorithms.

We finish with a small manufacturing example that illustrates how the theory can be applied, and then we report on applying our approach to a much larger example.

For future work, we would like to investigate additional fault scenarios as well as additional ways to model faults in the system.

References

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

[2] W. M. Wonham, *Supervisory Control of Discrete-Event Systems*, Department of Electrical and Computer Engineering, University of Toronto, July 2014, Monograph and TCT software can be downloaded at http://www.control.toronto.edu/DES/.

[3] W. M. Wonham and P. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control Optim*, vol. 25, no. 3, pp. 637–659, May 1987.

[4] S. Radel, A. Mulahuwaish, and R. J. Leduc, "Fault tolerant controllability," in *Proc. of 2015 American Control Conference*, Chicago, USA, July 2015, pp. 1603–1610.

[5] F. Lin, "Robust and adaptive supervisory control of discrete event systems," *IEEE Trans. Automatic Control*, vol. 38, no. 12, pp. 1848–1852, Dec. 1993.

[6] S.-J. Park and J.-T. Lim, "Fault-tolerant robust supervisor for discrete event systems with model uncertainty and its application to a workcell," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 386–391, 1999.

[7] A. Paoli, M. Sartini, and S. Lafortune, "Active fault tolerant control of discrete event systems using online diagnostics," *Automatica*, vol. 47, no. 4, pp. 639–649, 2011.

[8] Q. Wen, R. Kumar, J. Huang, and H. Liu, "A framework for fault-tolerant control of discrete event systems," *IEEE Trans. on Automatic Control*, vol. 53, pp. 1839–1849, 2008.

[9] B. A. Brandin, R. Malik, and P. Malik, "Incremental verification and synthesis of discrete-event systems guided by counter-examples," *IEEE Trans. on Control Systems Technology*, vol. 12, no. 3, pp. 387–401, May 2004.

[10] A. E. Bryant, "Symbolic boolean manipulation with ordered binary-decision diagrams," *ACM Computing Surveys*, vol. 24, pp. 293–318, 1992.

[11] C. Ma, "Nonblocking supervisory control of state tree structures," Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Toronto, 2004.

[12] R. Song, "Symbolic synthesis and verification of hierarchical interface-based supervisory control," Master's thesis, Dept. of Comput. and Softw., McMaster University, Hamilton, Ont, 2006.

[13] A. Vahidi, B. Lennartson, and M. Fabian, "Efficient analysis of large discrete-event systems with binary decision diagrams," in *Proc. of the 44th IEEE Conf. Decision Contr. and and 2005 European Contr. Conf.*, Seville, Spain, 2005, pp. 2751–2756.

[14] Y. Wang, "Sampled-data supervisory control," Master's thesis, Dept. of Computing and Software, McMaster University, Hamilton, Ont, 2009.

[15] Z. Zhang, "Smart TCT: an efficient algorithm for supervisory control design." Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 2001.

[16] K. Rudie, "Software for the control of discrete-event systems: A complexity study," Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1988.

[17] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems, 2nd ed.* Springer, 2009.

[18] R. Leduc, "PLC implementation of a DES supervisor for a manufacturing testbed: An implementation perspective," Master's thesis, Dept. of Elec and Comp Eng, University of Toronto, Toronto, Ont, 1996.

[19] DESpot, "`www.cas.mcmaster.ca/~leduc/DESpot.html`. The official website for the DESpot project," 2013.

[20] O. Dierikx, "Fault-tolerance of a DES supervisor for a manufacturing testbed," Technical University of Eindhoven, The Netherlands, Tech. Rep., Feb. 2015.

Appendices

A Proofs of Selected Propositions

Proposition 1:

Proof. Assume initial conditions for proposition.
Let $P_{\Delta F}$: $\Sigma^* \rightarrow \Sigma^*_{\Delta F}$ be a natural projection.
Let $s \in L(\mathbf{G})$. (P1.1)
Must show implies $s \notin L_{\Delta F} \iff s \in L(\mathbf{G}')$.
Sufficient to show (A) $s \notin L_{\Delta F} \Rightarrow s \in L(\mathbf{G}')$ and (B) $s \in L(\mathbf{G}') \Rightarrow s \notin L_{\Delta F}$
First we note that by Algorithm 5, we have $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}$.
We thus have $L(\mathbf{G}') = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$ as $\Sigma_{\Delta F} \subseteq \Sigma$, and $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$ by Algorithm 1. (P1.2)
We next note that by Algorithm 1, $\mathbf{G_{\Delta F}}$ contains an initial state but no transitions. We thus have: $L(\mathbf{G_{\Delta F}}) = \{\epsilon\}$ (P1.3)

Part A) Show $s \notin L_{\Delta F} \Rightarrow s \in L(\mathbf{G}')$
Assume $s \notin L_{\Delta F} = \Sigma^*.\Sigma_{\Delta F}.\Sigma^*$.
Must show implies: $s \in L(\mathbf{G}') = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$
As $s \in L(\mathbf{G})$ from (P1.1), sufficient to show $s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$.
As $s \notin \Sigma^*.\Sigma_{\Delta F}.\Sigma^*$, it follows that $P_{\Delta F}(s) = \epsilon$.
$\Rightarrow P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$, by (P1.3)
$\Rightarrow s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$, as required.

Part B) Show $s \in L(\mathbf{G}') \Rightarrow s \notin L_{\Delta F}$
Assume $s \in L(\mathbf{G}')$.
Must show implies: $s \notin L_{\Delta F}$
We note that $s \in L(\mathbf{G}')$ implies $s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$, by (P1.2).
$\Rightarrow P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$
$\Rightarrow P_{\Delta F}(s) = \epsilon$, by (P1.3)
This implies s does not contain any $\sigma \in \Sigma_{\Delta F}$.
$\Rightarrow s \notin \Sigma^*.\Sigma_{\Delta F}.\Sigma^*$, as required.

By parts (A) and (B), we have: $s \notin L_{\Delta F} \iff s \in L(\mathbf{G}')$ □

Proposition 2:

Proof. Assume initial conditions for proposition.
We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Proposition 1. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.
Let $P_{\Delta F} : \Sigma^* \rightarrow \Sigma^*_{\Delta F}$ and $P_F : \Sigma^* \rightarrow \Sigma^*_F$ be natural projections.
We next note that by Algorithm 6, we have $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}$.
As $\mathbf{G}$ defined over Σ, $\mathbf{G_{\Delta F}}$ over $\Sigma_{\Delta F}$ (by Algorithm 1), and $\mathbf{G_{NF}}$ over Σ_F (by Algorithm 2), we have: $L(\mathbf{G}') = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}})$ (P2.1)

Let $\mathbf{G_1}$ be the plant constructed by Algorithm 1. We thus have: $\mathbf{G_1} = \mathbf{G}||\mathbf{G_{\Delta F}}$

$\Rightarrow L(\mathbf{G_1}) = L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$

$\Rightarrow L(\mathbf{G'}) \subseteq L(\mathbf{G_1})$ (P2.2)

Let $s \in L(\mathbf{G})$ (P2.3)

Must show implies: $s \notin L_{\Delta F} \wedge s \in L_{NF} \iff s \in L(\mathbf{G'})$

Part A) Show $s \notin L_{\Delta F} \wedge s \in L_{NF} \Rightarrow s \in L(\mathbf{G'})$

Assume $s \notin L_{\Delta F}$ and $s \in L_{NF}$. (P2.4)

Must show: $s \in L(\mathbf{G'}) = L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$

By (P2.3), (P2.4), and Proposition 1, we have: $s \in L(\mathbf{G_1}) = L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$ (P2.5)

All the remains is to show $s \in P_F^{-1} L(\mathbf{G_{NF}})$.

As $s \in L_{NF} = (\Sigma - \Sigma_F)^* \cup \bigcup_{k=1}^{N} ((\Sigma - \Sigma_F)^*.\Sigma_F.(\Sigma - \Sigma_F)^*)^k$, there exists $0 \leq j \leq N$, such that $|P_F(s)| = j$.

We note that as $\mathbf{G_{NF}}$ contains an initial state, we have $\epsilon \in L(\mathbf{G_{NF}})$.

If $j = 0$, we immediately have $P_F(s) = \epsilon \in L(\mathbf{G_{NF}})$.

For $j \geq 1$, we can conclude: $(\exists \sigma_0, \ldots, \sigma_{j-1} \in \Sigma_F) P_F(s) = \sigma_0, \ldots, \sigma_{j-1}$

As $j \leq N$, it is easy to see from Algorithm 2, that for $i = 0, \ldots, j-1$, we have: $\delta_1(y_i, \sigma_i, y_{i+1})!$, where δ_1 is the transition function for $\mathbf{G_{NF}}$.

$\Rightarrow \delta_1(y_0, \sigma_0, \ldots, \sigma_{j-1})!$

$\Rightarrow \delta_1(y_0, P_F(s))!$

$\Rightarrow P_F(s) \in L(\mathbf{G_{NF}})$

$\Rightarrow s \in P_F^{-1} L(\mathbf{G_{NF}})$

Combining with (P2.5), we have: $s \in L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}}) = L(\mathbf{G'})$

Part B) Show $s \in L(\mathbf{G'}) \Rightarrow s \notin L_{\Delta F} \wedge s \in L_{NF}$

Assume $s \in L(\mathbf{G'})$. Must show implies $s \notin L_{\Delta F}$ and $s \in L_{NF}$.

As $s \in L(\mathbf{G'})$, we have $s \in L(\mathbf{G_1})$, by (P2.2).

We thus have by Proposition 1 that $s \notin L_{\Delta F}$. (P2.6)

We now need to show $s \in L_{NF}$.

As $L(\mathbf{G'}) = L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$ by (P2.1), we have $s \in P_F^{-1} L(\mathbf{G_{NF}})$.

$\Rightarrow P_F(s) \in L(\mathbf{G_{NF}})$

Let $j = |P_F(s)|$. If $j = 0$, we have $P_F(s) = \epsilon$, thus $s \in (\Sigma - \Sigma_F)^* \subseteq L_{NF}$.

We thus consider $j \geq 1$.

$\Rightarrow (\exists \sigma_0, \ldots, \sigma_{j-1} \in \Sigma_F) P_F(s) = \sigma_0, \ldots, \sigma_{j-1}$

As $P_F(s) \in L(\mathbf{G_{NF}})$, Algorithm 2 implies that for $i = 0, \ldots, j-1$, we have: $\delta_1(y_i, \sigma_i, y_{i+1})!$, where δ_1 is the transition function for $\mathbf{G_{NF}}$.

$\Rightarrow \delta_1(y_0, P_F(s)) = y_j$

As $\mathbf{G_{NF}}$ contains no loops and transitions occur in a strictly increasing order in terms of state labels, we have $j \leq N$.

As we have that s contains at most N events from Σ_F, it is thus clear that:

$$s \in (\Sigma - \Sigma_F)^* \cup \bigcup_{k=1}^{N} ((\Sigma - \Sigma_F)^*.\Sigma_F.(\Sigma - \Sigma_F)^*)^k = L_{NF}$$

Combining with (P2.6), we have $s \notin L_{\Delta F}$ and $s \in L_{NF}$, as required.

By parts (A) and (B), we thus conclude: $s \notin L_{\Delta F} \wedge s \in L_{NF} \iff s \in L(\mathbf{G}')$ $\square$

Proposition 3:

Proof. Assume initial conditions for proposition.

We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Proposition 1. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.

Let $P_{\Delta F} : \Sigma^* \to \Sigma^*_{\Delta F}$, $P_F : \Sigma^* \to \Sigma^*_F$, and $P_{F_i} : \Sigma^* \to \Sigma^*_{F_i}$, $i = 1, \ldots, m$, be natural projections.

We next note that by Algorithm 7, we have: $\mathbf{G'} = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{G_{F,1}}|| \ldots ||\mathbf{G_{F,m}}$

As $\mathbf{G}$ is defined over Σ, $\mathbf{G_{\Delta F}}$ over $\Sigma_{\Delta F}$ by Algorithm 1, $\mathbf{G_{NF}}$ over Σ_F by Algorithm 2, and $\mathbf{G_{F,i}}$ over Σ_{F_i} $(i = 1, \ldots, m)$ by Algorithm 3, we have:

$$L(\mathbf{G'}) = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L(\mathbf{G_{F,m}}) \quad \text{(P3.1)}$$

Let $\mathbf{G_1}$ be the plant constructed by Algorithm 2. We thus have: $\mathbf{G_1} = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}$

$\Rightarrow L(\mathbf{G_1}) = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}})$

$\Rightarrow L(\mathbf{G'}) \subseteq L(\mathbf{G_1})$ (P3.2)

Let $s \in L(\mathbf{G})$. (P3.3)

Must show implies: $s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF} \iff s \in L(\mathbf{G'})$

Part A) Show $s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF} \Rightarrow s \in L(\mathbf{G'})$

Assume $s \notin L_{\Delta F} \cup L_{NRF}$ and $s \in L_{NF}$. (P3.4)

Must show $s \in L(\mathbf{G'})$.

By (P3.3), (P3.4), and Proposition 2, we have: $s \in L(\mathbf{G_1})$

All the remains is to show $s \in P^{-1}_{F_i} L(\mathbf{G_{F,i}}), i = 1, \ldots, m$.

Let $i \in \{1, \ldots, m\}$.

As $s \notin L_{NRF} = \bigcup_{j=1}^{m} (\Sigma^*.\Sigma_{F_j}.\Sigma^*.\Sigma_{F_j}.\Sigma^*)$, it follows that $|P_{F_i}(s)| \leq 1$.

As $\mathbf{G_{F,i}}$ has an initial state (by Algorithm 3), we have $\epsilon \in L(\mathbf{G_{F,i}})$.

By Algorithm 3, we have that for all $\sigma \in \Sigma_{F_i}$, $\delta_i(y_0, \sigma, y_1)!$. This implies $\sigma \in L(\mathbf{G_{F,i}})$.

$\Rightarrow P_{F_i}(s) \in L(\mathbf{G_{F,i}})$

$\Rightarrow s \in P^{-1}_{F_i} L(\mathbf{G_{F,i}})$, as required.

Part B) Show $s \in L(\mathbf{G'}) \Rightarrow s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF}$

Assume $s \in L(\mathbf{G'})$.

Must show implies $s \notin L_{\Delta F} \cup L_{NRF}$ and $s \in L_{NF}$.

As $s \in L(\mathbf{G'})$, we have $s \in L(\mathbf{G_1})$, by (P3.2).

We can thus conclude by Proposition 2 that: $s \notin L_{\Delta F}$ and $s \in L_{NF}$. (P3.5)

We now only need to show $s \notin L_{NRF}$.

As $s \in L(\mathbf{G'})$, we have by (P3.1): $s \in P^{-1}_{F_i} L(\mathbf{G_{F,i}}), i = 1, \ldots, m$.

$\Rightarrow P_{F_i}(s) \in L(\mathbf{G_{F,i}})$, $i = 1, \ldots, m$.

$\Rightarrow P_{F_i}(s) = \sigma \in \Sigma_{F_i}$ or $P_{F_i}(s) = \epsilon$ $(i = 1, \ldots, m)$, by Algorithm 3.

$\Rightarrow s \notin L_{NRF} = \bigcup_{i=1}^{m} (\Sigma^*.\Sigma_{F_i}.\Sigma^*.\Sigma_{F_i}.\Sigma^*)$

Combining with (P3.5), we have $s \notin L_{\Delta F} \cup L_{NRF}$ and $s \in L_{NF}$, as required.

By parts (A) and (B), we thus conclude: $s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF} \iff s \in L(\mathbf{G}')$ $\square$

Proposition 4:

Proof. Assume initial conditions for proposition.

We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Proposition 1. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.

Let $P_{\Delta F} : \Sigma^* \to \Sigma^*_{\Delta F}$ and $P_{TF_i} : \Sigma^* \to (\Sigma_{F_i} \cup \Sigma_{T_i})^*$, $i = 1, \ldots, m$, be natural projections.

We next note that by Algorithm 8, we have: $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{TF,1}}|| \ldots ||\mathbf{G_{TF,m}}$

As $\mathbf{G}$ is defined over Σ, $\mathbf{G_{\Delta F}}$ over $\Sigma_{\Delta F}$ by Algorithm 1, and $\mathbf{G_{TF,i}}$ over $\Sigma_{F_i} \cup \Sigma_{T_i}$ by Algorithm 4, we have:

$$L(\mathbf{G}') = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}}) \quad \text{(P4.1)}$$

Let $\mathbf{G_1}$ be the plant constructed by Algorithm 1. We thus have: $\mathbf{G_1} = \mathbf{G}||\mathbf{G_{\Delta F}}$

$\Rightarrow L(\mathbf{G_1}) = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$

$\Rightarrow L(\mathbf{G}') \subseteq L(\mathbf{G_1})$ (P4.2)

Let $s \in L(\mathbf{G})$. (P4.3)

Must show implies: $s \notin L_{\Delta F} \cup L_{TF} \iff s \in L(\mathbf{G}')$

Part A) Show $s \notin L_{\Delta F} \cup L_{TF} \Rightarrow s \in L(\mathbf{G}')$

Assume $s \notin L_{\Delta F} \cup L_{TF}$. (P4.4)

Must show $s \in L(\mathbf{G}') = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$.

By (P4.3), (P4.4) and Proposition 1, we have: $s \in L(\mathbf{G_1}) = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$

All that remains is to show $s \in P^{-1}_{TF_i} L(\mathbf{G_{TF,i}})$, $i = 1, \ldots, m$.

As $s \notin L_{TF} = \bigcup_{i=1}^{m} (\Sigma^*.\Sigma_{F_i}.(\Sigma - \Sigma_{T_i})^*.\Sigma_{F_i}.\Sigma^*)$, it follows that:

$(\forall i \in \{1, \ldots, m\})\ s \notin \Sigma^*.\Sigma_{F_i}.(\Sigma - \Sigma_{T_i})^*.\Sigma_{F_i}.\Sigma^*$.

Let $i = \{1, \ldots, m\}$.

We will use proof by contrapositive.

Sufficient to show: $P_{TF_i}(s) \notin L(\mathbf{G_{TF,i}}) \Rightarrow s \in \Sigma^*.\Sigma_{F_i}.(\Sigma - \Sigma_{T_i})^*.\Sigma_{F_i}.\Sigma^*$

Assume $P_{TF_i}(s) \notin L(\mathbf{G_{TF,i}})$.

We note that by Algorithm 4 that $\epsilon \in L(\mathbf{G_{TF,i}})$, as $\mathbf{G_{TF,i}}$ has an initial state.

$\Rightarrow (\exists s' \in (\Sigma_{F_i} \cup \Sigma_{T_i})^*)(\exists \sigma \in \Sigma_{F_i} \cup \Sigma_{T_i}) s'\sigma \leq P_{TF_i}(s) \wedge s' \in L(\mathbf{G_{TF_i}}) \wedge s'\sigma \notin L(\mathbf{G_{TF_i}})$

From Algorithm 4, it is clear that all $\sigma' \in \Sigma_{F_i} \cup \Sigma_{T_i}$ are defined at state y_0, all $\sigma' \in \Sigma_{T_i}$ are defined at state y_1, and no $\sigma' \in \Sigma_{F_i}$ are defined at state y_1.

$\Rightarrow \delta_i(y_0, s') = y_1$, and $\sigma \in \Sigma_{F_i}$

Also, as the only way to reach state y_1 is from state y_0 via $\sigma' \in \Sigma_{F_i}$, it follows that string s' ends in an event from Σ_{F_i}.

$\Rightarrow (\exists s'' \in (\Sigma_{F_i} \cup \Sigma_{T_i})^*)(\exists \sigma' \in \Sigma_{F_i})\, s''\sigma'\sigma = s'\sigma \leq P_{TF_i}(s)$

$\Rightarrow s \in \Sigma^*.\Sigma_{F_i}.(\Sigma - \Sigma_{T_i})^*.\Sigma_{F_i}.\Sigma^*$, as required.

Part B) Show $s \in L(\mathbf{G}') \Rightarrow s \notin L_{\Delta F} \cup L_{TF}$

Assume $s \in L(\mathbf{G}')$. Must show implies $s \notin L_{\Delta F} \cup L_{TF}$.

As $s \in L(\mathbf{G'})$, we have $s \in L(\mathbf{G_1})$, by (P4.2).

We can thus conclude by Proposition 1 that: $s \notin L_{\Delta F}$ (P4.5)

We now need to show $s \notin L_{TF}$.

As $s \in L(\mathbf{G'})$, we have by (P4.1): $s \in P^{-1}_{TF_i} L(\mathbf{G_{TF,i}}), i = 1, \ldots, m$

$\Rightarrow (\forall i \in \{1, \ldots, m\}) P_{TF_i}(s) \in L(\mathbf{G_{TF,i}})$

We proceed by proof by contradiction.

Assume $s \in L_{TF}$.

$\Rightarrow (\exists i \in \{1, \ldots, m\}) s \in \Sigma^*.\Sigma_{F_i}.(\Sigma - \Sigma_{T_i})^*.\Sigma_{F_i}.\Sigma^*$

Let $i \in \{1, \ldots, m\}$ be the above index.

This implies string $P_{TF_i}(s)$ contains two events from Σ_{F_i} in a row, without a $\sigma \in \Sigma_{T_i}$ in between.

As it is clear from Algorithm 4 that $\mathbf{G_{TF,i}}$ would never allow two $\sigma \in \Sigma_{F_i}$ to occur in a row, this contradicts $P_{TF_i}(s) \in L(\mathbf{G_{TF,i}})$.

We thus conclude $s \notin L_{TF}$.

Combining with (P4.5) we have $s \notin L_{\Delta F} \cup L_{TF}$, as required.

By parts (A) and (B), we thus conclude: $s \notin L_{\Delta F} \cup L_{TF} \iff s \in L(\mathbf{G'})$ □

B Proofs of Selected Theorems

Theorem 1:

Proof. Assume initial conditions for theorem.
Must show $\mathbf{S}$ is fault tolerant controllable for $\mathbf{G} \iff \mathbf{S}$ is controllable for $\mathbf{G'}$.
From Algorithm 5, we have: $\mathbf{G'} = \mathbf{G}||\mathbf{G_{\Delta F}}$
From Algorithm 1, we know that $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$.
Let $P_{\Delta F} : \Sigma^* \rightarrow \Sigma^*_{\Delta F}$ be a natural projection.
As $\mathbf{G}$ is defined over Σ, we have: $L(\mathbf{G'}) = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$ (T1.1)

Part A) Show ($\Rightarrow$)
Assume $\mathbf{S}$ is fault tolerant controllable for $\mathbf{G}$. (T1.2)
Must show implies: $(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G'}))(\forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G'}) \Rightarrow s\sigma \in L(\mathbf{S})$
Let $s \in L(\mathbf{S}) \cap L(\mathbf{G'})$ and $\sigma \in \Sigma_u$. (T1.3)
Assume $s\sigma \in L(\mathbf{G'})$. (T1.4)
Must show implies $s\sigma \in L(\mathbf{S})$.
To apply (T1.2), we need to show that $s \in L(\mathbf{S}) \cap L(\mathbf{G})$, $s\sigma \in L(\mathbf{G})$ and $s \notin L_{\Delta F}$.
We first note that (T1.1), (T1.3) and (T1.4) imply:
$s \in L(\mathbf{S})$, $s \in L(\mathbf{G})$, and $s\sigma \in L(\mathbf{G})$
As $s \in L(\mathbf{G'})$ by (T1.3), we conclude by Proposition 1 that $s \notin L_{\Delta F}$.
We can now conclude by (T1.2) that $s\sigma \in L(\mathbf{S})$, as required.

Part B) Show ($\Leftarrow$)
Assume $\mathbf{S}$ is controllable for $\mathbf{G'}$. (T1.5)
Must show implies $\mathbf{S}$ and $\mathbf{G}$ are FT consistent (follows automatically from initial assumptions) and that: $(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G}) \wedge s \notin L_{\Delta F} \Rightarrow s\sigma \in L(\mathbf{S})$
Let $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ and $\sigma \in \Sigma_u$. Assume $s\sigma \in L(\mathbf{G})$ and $s \notin L_{\Delta F}$. (T1.6)
Must show implies $s\sigma \in L(\mathbf{S})$.
We have two cases: (1) $\sigma \in \Sigma_{\Delta F}$, and (2) $\sigma \notin \Sigma_{\Delta F}$

Case 1) $\sigma \in \Sigma_{\Delta F}$
As the system is FT consistent, it follows that σ is self-looped at every state in $\mathbf{S}$.
As $s \in L(\mathbf{S})$ by (T1.6), it thus follows that $s\sigma \in L(\mathbf{S})$, as required.

Case 2) $\sigma \notin \Sigma_{\Delta F}$
To apply (T1.5), we still need to show $s \in L(\mathbf{S}) \cap L(\mathbf{G'})$, and $s\sigma \in L(\mathbf{G'})$.
We first note that by (T1.6) and Proposition 1, we can conclude: $s \in L(\mathbf{G'})$ (T1.7)
$\Rightarrow s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$, by (T1.1)
$\Rightarrow P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$
As $\sigma \notin \Sigma_{\Delta F}$, we have $P_{\Delta F}(\sigma) = \epsilon$.
$\Rightarrow P_{\Delta F}(s\sigma) = P_{\Delta F}(s)P_{\Delta F}(\sigma) = P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$
$\Rightarrow s\sigma \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$
Combining with (T1.6), (T1.7), and (T1.1), we have: $s \in L(\mathbf{S}) \cap L(\mathbf{G'})$, $\sigma \in \Sigma_u$, and $s\sigma \in L(\mathbf{G'})$

We can thus conclude by (T1.5) that $s\sigma \in L(\mathbf{S})$, as required.

We thus conclude by cases (1) and (2), that $s\sigma \in L(\mathbf{S})$.

We can now conclude by parts (A) and (B) that $\mathbf{S}$ is fault tolerant controllable for $\mathbf{G}$ iff $\mathbf{S}$ is controllable for $\mathbf{G'}$. □

Theorem 2:

Proof. Assume initial conditions for theorem.

We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Theorem 1. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.

Must show $\mathbf{S}$ is N-fault tolerant controllable for $\mathbf{G} \iff \mathbf{S}$ is controllable for $\mathbf{G'}$.

From Algorithm 6, we have $\mathbf{G'} = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}$.

From Algorithm 1, we know that $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$, and from Algorithm 2, we know that $\mathbf{G_{NF}}$ is defined over Σ_F.

Let $P_{\Delta F} : \Sigma^* \rightarrow \Sigma^*_{\Delta F}$ and $P_F : \Sigma^* \rightarrow \Sigma^*_F$ be natural projections.

As $\mathbf{G}$ is defined over Σ, we have: $L(\mathbf{G'}) = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}})$ (T2.1)

Part A) Show ($\Rightarrow$)

Assume $\mathbf{S}$ is N-fault tolerant controllable for $\mathbf{G}$. (T2.2)

Must show implies: $(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G'}))(\forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G'}) \Rightarrow s\sigma \in L(\mathbf{S})$

Let $s \in L(\mathbf{S}) \cap L(\mathbf{G'})$, and $\sigma \in \Sigma_u$. (T2.3)

Assume $s\sigma \in L(\mathbf{G'})$. (T2.4)

Must show implies $s\sigma \in L(\mathbf{S})$.

To apply (T2.2), we need to show that $s \in L(\mathbf{S}) \cap L(\mathbf{G})$, $s\sigma \in L(\mathbf{G})$ and $s \notin L_{\Delta F} \wedge s \in L_{NF}$.

We first note that (T2.1), (T2.3) and (T2.4) imply $s \in L(\mathbf{S})$, $s \in L(\mathbf{G})$, and $s\sigma \in L(\mathbf{G})$.

As $s \in L(\mathbf{G'})$ by (T2.3), Proposition 2 implies that: $s \notin L_{\Delta F} \wedge s \in L_{NF}$

We can now conclude by (T2.2) that $s\sigma \in L(\mathbf{S})$, as required.

Part B) Show ($\Leftarrow$)

Assume $\mathbf{S}$ is controllable for $\mathbf{G'}$. (T2.5)

Must show implies $\mathbf{S}$ and $\mathbf{G}$ are FT consistent, (follows automatically from initial assumptions) and that:

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G}) \wedge s \notin L_{\Delta F} \wedge s \in L_{NF} \Rightarrow s\sigma \in L(\mathbf{S})$

Let $s \in L(\mathbf{S}) \cap L(\mathbf{G})$, $\sigma \in \Sigma_u$. Assume $s\sigma \in L(\mathbf{G})$ and $s \notin L_{\Delta F} \wedge s \in L_{NF}$. (T2.6)

Must show implies $s\sigma \in L(\mathbf{S})$.

We have two cases: (1) $\sigma \in \Sigma_{\Delta F} \cup \Sigma_F$, and (2) $\sigma \notin \Sigma_{\Delta F} \cup \Sigma_F$

Case 1) $\sigma \in \Sigma_{\Delta F} \cup \Sigma_F$

As the system is FT consistent, it follows that σ is self-looped at every state in $\mathbf{S}$.

As $s \in L(\mathbf{S})$ by (T2.6), it thus follows that $s\sigma \in L(\mathbf{S})$, as required.

Case 2) $\sigma \notin \Sigma_{\Delta F} \cup \Sigma_F$

To apply (T2.5), we still need to show $s \in L(\mathbf{S}) \cap L(\mathbf{G'})$, and $s\sigma \in L(\mathbf{G'})$.

We first note that by (T2.6) and Proposition 2, we can conclude: $s \in L(\mathbf{G'})$. (T2.7)

$\Rightarrow s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}})$, by (T2.1)

$\Rightarrow P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$ and $P_F(s) \in L(\mathbf{G_{NF}})$
As $\sigma \notin \Sigma_{\Delta F}$, we have $P_{\Delta F}(\sigma) = \epsilon$. As $\sigma \notin \Sigma_F$, we have $P_F(\sigma) = \epsilon$.
$\Rightarrow P_{\Delta F}(s\sigma) = P_{\Delta F}(s)P_{\Delta F}(\sigma) = P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$
$\Rightarrow P_F(s\sigma) = P_F(s)P_F(\sigma) = P_F(s) \in L(\mathbf{G_{NF}})$
$\Rightarrow s\sigma \in P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}}) \cap P_F^{-1}L(\mathbf{G_{NF}})$
Combining with (T2.6), (T2.7), and (T2.1), we have: $s \in L(\mathbf{S}) \cap L(\mathbf{G}')$, $\sigma \in \Sigma_u$, and $s\sigma \in L(\mathbf{G}')$.
We can thus conclude by (T2.5) that $s\sigma \in L(\mathbf{S})$, as required.

We thus conclude by cases (1) and (2), that $s\sigma \in L(\mathbf{S})$.
We can now conclude by parts (A) and (B), that $\mathbf{S}$ is N-fault tolerant controllable for $\mathbf{G}$ iff $\mathbf{S}$ is controllable for $\mathbf{G}'$. □

Theorem 3:

Proof. Assume initial conditions for theorem.
We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Theorem 1. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.
Must show $\mathbf{S}$ is non-repeatable N-fault tolerant controllable for $\mathbf{G} \iff \mathbf{S}$ is controllable for $\mathbf{G}'$.
From Algorithm 7, we have: $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{G_{F,1}}||\ldots||\mathbf{G_{F,m}}$
From Algorithm 1, we know that $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$. From Algorithm 2, we know that $\mathbf{G_{NF}}$ is defined over Σ_F, and from Algorithm 3, we know that $\mathbf{G_{F,i}}$ is defined over Σ_{F_i}, $i = 1, \ldots, m$.
Let $P_{\Delta F} : \Sigma^* \to \Sigma_{\Delta F}^*$, $P_F : \Sigma^* \to \Sigma_F^*$, and $P_{F_i} : \Sigma^* \to \Sigma_{F_i}^*$, $i = 1, \ldots, m$, be natural projections.
As $\mathbf{G}$ is defined over Σ, we have that:

$$L(\mathbf{G}') = L(\mathbf{G}) \cap P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}}) \cap P_F^{-1}L(\mathbf{G_{NF}}) \cap P_{F_1}^{-1}L(\mathbf{G_{F,1}}) \cap \ldots \cap P_{F_m}^{-1}L(\mathbf{G_{F,m}}) \quad \text{(T3.1)}$$

Part A) Show ($\Rightarrow$)
Assume $\mathbf{S}$ is non-repeatable N-fault tolerant controllable for $\mathbf{G}$. (T3.2)
Must show implies: $(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}'))(\forall \sigma \in \Sigma_u) s\sigma \in L(\mathbf{G}') \Rightarrow s\sigma \in L(\mathbf{S})$
Let $s \in L(\mathbf{S}) \cap L(\mathbf{G}')$, and $\sigma \in \Sigma_u$. (T3.3)
Assume $s\sigma \in L(\mathbf{G}')$. (T3.4)
Must show implies $s\sigma \in L(\mathbf{S})$.
To apply (T3.2), we need to show that $s \in L(\mathbf{S}) \cap L(\mathbf{G})$, $s\sigma \in L(\mathbf{G})$, $s \notin L_{\Delta F} \cup L_{NRF}$ and $s \in L_{NF}$.
We first note that (T3.1), (T3.3) and (T3.4) imply $s \in L(\mathbf{S})$, $s \in L(\mathbf{G})$, and $s\sigma \in L(\mathbf{G})$.
As $s \in L(\mathbf{G}')$ by (T3.3), we conclude by Proposition 3 that: $s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF}$
We can now conclude by (T3.2) that $s\sigma \in L(\mathbf{S})$, as required.

Part B) Show ($\Leftarrow$)
Assume $\mathbf{S}$ is controllable for $\mathbf{G}'$. (T3.5)
Must show implies $\mathbf{S}$ and $\mathbf{G}$ are FT consistent (follows automatically from initial assumptions) and that:

$$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G}) \wedge s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF} \Rightarrow s\sigma \in L(\mathbf{S})$$

Let $s \in L(\mathbf{S}) \cap L(\mathbf{G})$, $\sigma \in \Sigma_u$. Assume $s\sigma \in L(\mathbf{G})$, and $s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF}$. (T3.6)
Must show implies $s\sigma \in L(\mathbf{S})$.
We have two cases: (1) $\sigma \in \Sigma_{\Delta F} \cup \Sigma_{F_i}$, and (2) $\sigma \notin \Sigma_{\Delta F} \cup \Sigma_{F_i}$

Case 1) $\sigma \in \Sigma_{\Delta F} \cup \Sigma_F$
As the system is FT consistent, it follows that σ is self-looped at every state in $\mathbf{S}$.
As $s \in L(\mathbf{S})$ by (T3.6), it thus follows that $s\sigma \in L(\mathbf{S})$, as required.

Case 2) $\sigma \notin \Sigma_{\Delta F} \cup \Sigma_F$
To apply (T3.5), we still need to show $s \in L(\mathbf{S}) \cap L(\mathbf{G}')$, and $s\sigma \in L(\mathbf{G}')$.
We first note that by (T3.6), and Proposition 3, we can conclude: $s \in L(\mathbf{G}')$ (T3.7)
$\Rightarrow s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{F} L(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L(\mathbf{G_{F,m}})$, by (T3.1)
$\Rightarrow P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}}), P_F(s) \in L(\mathbf{G_{NF}})$ and $P_{F_i}(s) \in L(\mathbf{G_{F,i}})$, $i = 1, \ldots, m$
As $\sigma \notin \Sigma_{\Delta F} \cup \Sigma_F$, we have $P_{\Delta F}(\sigma) = \epsilon, P_F(\sigma) = \epsilon$, and $P_{F_i}(\sigma) = \epsilon$, $i = 1, \ldots, m$.
This implies $P_{\Delta F}(s\sigma) = P_{\Delta F}(s)P_{\Delta F}(\sigma) = P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$, and $P_F(s\sigma) = P_F(s)P_F(\sigma) = P_F(s) \in L(\mathbf{G_{NF}})$, and $P_{F_i}(s\sigma) = P_{F_i}(s)P_{F_i}(\sigma) = P_{F_i}(s) \in L(\mathbf{G_{F,i}}), i = 1, \ldots, m$.
$\Rightarrow s\sigma \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{F} L(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L(\mathbf{G_{F,m}})$
Combining with (T3.6), (T3.7), and (T3.1), we have: $s \in L(\mathbf{S}) \cap L(\mathbf{G}')$, $\sigma \in \Sigma_u$, and $s\sigma \in L(\mathbf{G}')$
We can thus conclude by (T3.5) that $s\sigma \in L(\mathbf{S})$, as required.

We thus conclude by cases (1) and (2), that $s\sigma \in L(\mathbf{S})$.

We can now conclude by parts (A) and (B), that $\mathbf{S}$ is non repeatable N-fault tolerant controllable for $\mathbf{G}$ iff $\mathbf{S}$ is controllable for $\mathbf{G}'$. □

Theorem 4:

Proof. Assume initial conditions for theorem.
We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Theorem 1. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.
Must show $\mathbf{S}$ is resettable fault tolerant controllable for $\mathbf{G} \iff \mathbf{S}$ is controllable for $\mathbf{G}'$.
From Algorithm 8, we have: $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{TF,1}}|| \ldots ||\mathbf{G_{TF,m}}$
From Algorithm 1, we know that $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$, and from Algorithm 4, we know that $\mathbf{G_{TF,i}}$ is defined over $\Sigma_{F_i} \cup \Sigma_{T_i}$, $i = 1, \ldots, m$.
Let $P_{\Delta F} : \Sigma^* \to \Sigma^*_{\Delta F}$ and $P_{TF_i} : \Sigma^* \to (\Sigma_{F_i} \cup \Sigma_{T_i})^*$, $i = 1, \ldots, m$, be natural projections.
As $\mathbf{G}$ is defined over Σ, we have that:
$$L(\mathbf{G}') = L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}}) \quad \text{(T4.1)}$$

Part A) Show ($\Rightarrow$)
Assume $\mathbf{S}$ is resettable fault tolerant controllable for $\mathbf{G}$. (T4.2)
Must show implies: $(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}'))(\forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G}') \Rightarrow s\sigma \in L(\mathbf{S})$
Let $s \in L(\mathbf{S}) \cap L(\mathbf{G}')$, and $\sigma \in \Sigma_u$. (T4.3)
Assume $s\sigma \in L(\mathbf{G}')$. (T4.4)
Must show implies $s\sigma \in L(\mathbf{S})$.
To apply (T4.2), we need to show that $s \in L(\mathbf{S}) \cap L(\mathbf{G})$, $s\sigma \in L(\mathbf{G})$ and $s \notin L_{\Delta F} \cup L_{TF}$.
We first note that (T4.1), (T4.3) and (T4.4) imply $s \in L(\mathbf{S})$, $s \in L(\mathbf{G})$, and $s\sigma \in L(\mathbf{G})$.

As $s \in L(\mathbf{G}')$ by (T4.3), we conclude by Proposition 4 that: $s \notin L_{\Delta F} \cup L_{TF}$
We can now conclude by (T4.2) that $s\sigma \in L(\mathbf{S})$, as required.

Part B) Show ($\Leftarrow$)
Assume $\mathbf{S}$ is controllable for $\mathbf{G}'$. (T4.5)
Must show implies $\mathbf{S}$ and $\mathbf{G}$ are FT consistent, (follows automatically from initial assumptions) and that:
$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))(\forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G}) \wedge s \notin L_{\Delta F} \cup L_{TF} \Rightarrow s\sigma \in L(\mathbf{S})$
Let $s \in L(\mathbf{S}) \cap L(\mathbf{G})$, $\sigma \in \Sigma_u$. Assume $s\sigma \in L(\mathbf{G})$ and $s \notin L_{\Delta F} \cup L_{TF}$. (T4.6)
Must show implies $s\sigma \in L(\mathbf{S})$.
We have two cases: (1) $\sigma \in \Sigma_{\Delta F} \cup \Sigma_F$, and (2) $\sigma \notin \Sigma_{\Delta F} \cup \Sigma_F$

Case 1) $\sigma \in \Sigma_{\Delta F} \cup \Sigma_F$
As the system is FT consistent, it follows that σ is self-looped at every state in $\mathbf{S}$.
As $s \in L(\mathbf{S})$ by (T4.6), it thus follows that $s\sigma \in L(\mathbf{S})$, as required.

Case 2) $\sigma \notin \Sigma_{\Delta F} \cup \Sigma_F$
To apply (T4.5), we still need to show $s \in L(\mathbf{S}) \cap L(\mathbf{G}')$, and $s\sigma \in L(\mathbf{G}')$.
We first note that by (T4.6) and Proposition 4, we can conclude: $s \in L(\mathbf{G}')$ (T4.7)
$\Rightarrow s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$, by (T4.1)
$\Rightarrow P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$ and $P_{TF_i}(s) \in L(\mathbf{G_{TF,i}})$, $i = 1, \ldots, m$ (T4.8)
As $\sigma \notin \Sigma_{\Delta F}$, we have $P_{\Delta F}(\sigma) = \epsilon$.
$\Rightarrow P_{\Delta F}(s\sigma) = P_{\Delta F}(s)P_{\Delta F}(\sigma) = P_{\Delta F}(s) \in L(\mathbf{G_{\Delta F}})$
$\Rightarrow s\sigma \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}})$ (T4.9)
We now have two cases to consider: (a) $\sigma \notin \bigcup_{i=1}^{m} \Sigma_{T_i}$, and (b) $\sigma \in \bigcup_{i=1}^{m} \Sigma_{T_i}$

Case a) $\sigma \notin \bigcup_{i=1}^{m} \Sigma_{T_i}$
As $\sigma \notin \Sigma_F \cup \bigcup_{i=1}^{m} \Sigma_{T_i}$, we have $P_{TF_i}(\sigma) = \epsilon$, $i = 1, \ldots, m$.
$\Rightarrow P_{TF_i}(s\sigma) = P_{TF_i}(s)P_{TF_i}(\sigma) = P_{TF_i}(s) \in L(\mathbf{G_{TF,i}})$, $i = 1, \ldots, m$
$\Rightarrow s\sigma \in P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$

Case b) $\sigma \in \bigcup_{i=1}^{m} \Sigma_{T_i}$
We note that Algorithm 4 states that all $\sigma' \in \Sigma_{T_i}$ are defined at every state in $\mathbf{G_{TF,i}}$, $i = 1, \ldots, m$.
Let $j \in \{1, \ldots, m\}$.
If $\sigma \in \Sigma_{T_j}$, we have $P_{TF_j}(\sigma) = \sigma$. We thus have $P_{TF_j}(s\sigma) = P_{TF_j}(s)\sigma \in L(\mathbf{G_{TF,j}})$ as $P_{TF_j}(s) \in L(\mathbf{G_{TF,j}})$ by (T4.8).
Otherwise, $\sigma \notin \Sigma_{T_j}$. As we also have $\sigma \notin \Sigma_F$, it follows that $P_{TF_j}(\sigma) = \epsilon$. We thus have $P_{TF_j}(s\sigma) = P_{TF_j}(s)P_{TF_j}(\sigma) = P_{TF_j}(s) \in L(\mathbf{G_{TF,j}})$, by (T4.8).
$\Rightarrow s\sigma \in P^{-1}_{TF_j} L(\mathbf{G_{TF,j}})$ for both cases.
$\Rightarrow s\sigma \in P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$

By cases (a) and (b), we can conclude: $s\sigma \in P_{TF_1}^{-1}L(\mathbf{G_{TF,1}}) \cap \ldots \cap P_{TF_m}^{-1}L(\mathbf{G_{TF,m}})$
Combining with (T4.9), we have:
$s\sigma \in P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}}) \cap P_{TF_1}^{-1}L(\mathbf{G_{TF,1}}) \cap \ldots \cap P_{TF_m}^{-1}L(\mathbf{G_{TF,m}})$
Combining with (T4.6), (T4.7), and (T4.1), we have: $s \in L(\mathbf{S}) \cap L(\mathbf{G}')$, $\sigma \in \Sigma_u$, and $s\sigma \in L(\mathbf{G}')$.
We can thus conclude by (T4.5) that $s\sigma \in L(\mathbf{S})$, as required.

We thus conclude by cases (1) and (2), that $s\sigma \in L(\mathbf{S})$.

We can now conclude by parts (A) and (B), that $\mathbf{S}$ is resettable fault tolerant controllable for $\mathbf{G}$ iff $\mathbf{S}$ is controllable for $\mathbf{G}'$. □

Theorem 5:

Proof. Assume initial conditions for theorem.
Must show $\mathbf{S}$ and $\mathbf{G}$ are fault tolerant nonblocking $\iff$ $\mathbf{G}'$ is nonblocking.
From Algorithm 9, we have: $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{S}$
From Algorithm 1, we know that $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$.
Let $P_{\Delta F} : \Sigma^* \to \Sigma_{\Delta F}^*$ be a natural projection.
As $\mathbf{G}$ and $\mathbf{S}$ are defined over Σ, we have that: $L(\mathbf{G}') = L(\mathbf{S}) \cap L(\mathbf{G}) \cap P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}})$ and $L_m(\mathbf{G}') = L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P_{\Delta F}^{-1}L_m(\mathbf{G_{\Delta F}})$. (T5.1)

Part A) Show ($\Rightarrow$)
Assume $\mathbf{S}$ and $\mathbf{G}$ are fault tolerant nonblocking. (T5.2)
Must show implies: $(\forall s \in L(\mathbf{G}'))(\exists s' \in \Sigma^*)ss' \in L_m(\mathbf{G}')$
Let $s \in L(\mathbf{G}')$.
$\Rightarrow s \in L(\mathbf{S}) \cap L(\mathbf{G}) \cap P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}})$ (T5.3)
$\Rightarrow s \in L(\mathbf{G}) \cap P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}})$
$\Rightarrow s \in L(\mathbf{G}||\mathbf{G_{\Delta F}})$
We can thus apply Proposition 1 and conclude that $s \notin L_{\Delta F}$.
As we have $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ from (T5.3), we can apply (T5.2) and conclude that:
$(\exists s' \in \Sigma^*)ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \wedge ss' \notin L_{\Delta F}$ (T5.4)
We now need to show that $ss' \in L_m(\mathbf{G}')$.
Sufficient to show: $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P_{\Delta F}^{-1}L_m(\mathbf{G_{\Delta F}})$
From (T5.4), we have $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})$, so only need to show $ss' \in P_{\Delta F}^{-1}L_m(\mathbf{G_{\Delta F}})$.
We note from Algorithm 1 that since all states in $\mathbf{G_{\Delta F}}$ are marked, we have $L(\mathbf{G_{\Delta F}}) = L_m(\mathbf{G_{\Delta F}})$.
It is thus sufficient to show: $ss' \in P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}})$
As $ss' \in L_m(\mathbf{G})$ by (T5.4), we have $ss' \in L(\mathbf{G})$, since $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$.
From (T5.4), we have: $ss' \notin L_{\Delta F}$
Applying Proposition 1, we can conclude that: $ss' \in L(\mathbf{G}||\mathbf{G_{\Delta F}}) = L(\mathbf{G}) \cap P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}})$
$\Rightarrow ss' \in P_{\Delta F}^{-1}L(\mathbf{G_{\Delta F}})$
We thus have that $\mathbf{G}'$ is nonblocking, as required.

Part B) Show ($\Leftarrow$)
Assume $\mathbf{G}'$ is nonblocking. (T5.5)

Must show implies $\mathbf{S}$ and $\mathbf{G}$ are FT consistent (follows from initial assumptions) and that:
$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))\, s \notin L_{\Delta F} \Rightarrow (\exists s' \in \Sigma^*)\, ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \wedge ss' \notin L_{\Delta F}$
Let $s \in L(\mathbf{S}) \cap L(\mathbf{G})$. (T5.6)
Assume $s \notin L_{\Delta F}$. (T5.7)
To apply (T5.5), we need to show: $s \in L(\mathbf{G}') = L(\mathbf{S}) \cap L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$
As we have $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ from (T5.6), we only still need to show $s \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$.
By (T5.6) and (T5.7), we can apply Proposition 1 and conclude:
$s \in L(\mathbf{G}||\mathbf{G_{\Delta F}}) = L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$
We thus have $s \in L(\mathbf{G}')$. As $\mathbf{G}'$ is nonblocking, we can conclude: $(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{G}')$
$\Rightarrow ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}})$, by (T5.1)
We thus have $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})$, and only need to show that $ss' \notin L_{\Delta F}$.
We first note that we have $ss' \in L(\mathbf{G})$, as $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$.
We next note that $ss' \in P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}})$ implies $ss' \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$ as every state is marked in $\mathbf{G_{\Delta F}}$, by Algorithm 1.
$\Rightarrow ss' \in L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) = L(\mathbf{G}||\mathbf{G_{\Delta F}})$
We can now conclude by Proposition 1 that $ss' \notin L_{\Delta F}$.
We thus conclude that $\mathbf{S}$ and $\mathbf{G}$ are fault tolerant nonblocking.

We can thus conclude by parts (A) and (B), that $\mathbf{S}$ and $\mathbf{G}$ are fault tolerant nonblocking iff $\mathbf{G}'$ is nonblocking. □

Theorem 6:

Proof. Assume initial conditions for theorem.
We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Theorem 5. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.
Must show $\mathbf{S}$ and $\mathbf{G}$ are N-fault tolerant nonblocking $\iff$ $\mathbf{G}'$ is nonblocking.
From Algorithm 10, we have: $\mathbf{G}' = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{S}$
From Algorithm 1, we know that $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$, and from Algorithm 2, we know that $\mathbf{G_{NF}}$ is defined over Σ_F.
Let $P_{\Delta F} : \Sigma^* \to \Sigma_{\Delta F}^*$ and $P_F : \Sigma^* \to \Sigma_F^*$ be natural projections.
As $\mathbf{G}$ and $\mathbf{S}$ are defined over Σ, we have $L(\mathbf{G}') = L(\mathbf{S}) \cap L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$
and $L_m(\mathbf{G}') = L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L_m(\mathbf{G_{NF}})$. (T6.1)

PartA) Show ($\Rightarrow$)
Assume $\mathbf{S}$ and $\mathbf{G}$ are N-fault tolerant nonblocking. (T6.2)
Must show implies: $(\forall s \in L(\mathbf{G}'))(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{G}')$
Let $s \in L(\mathbf{G}')$.
$\Rightarrow s \in L(\mathbf{S}) \cap L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$ (T6.3)
$\Rightarrow s \in L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$
$\Rightarrow s \in L(\mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}})$
We can thus apply Proposition 2 and conclude: $s \notin L_{\Delta F} \wedge s \in L_{NF}$.
As we have $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ from (T6.3), we can apply (T6.2) and conclude that:
$(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \wedge ss' \notin L_{\Delta F} \wedge ss' \in L_{NF}$ (T6.4)

We now need to show that $ss' \in L_m(\mathbf{G}')$.

Sufficient to show: $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L_m(\mathbf{G_{NF}})$.

From (T6.4), we have $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})$, so only need to show $ss' \in P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L_m(\mathbf{G_{NF}})$.

We note from Algorithm 1 that as all states in $\mathbf{G_{\Delta F}}$ are marked, we have $L(\mathbf{G_{\Delta F}}) = L_m(\mathbf{G_{\Delta F}})$. From Algorithm 2, we have that all states in $\mathbf{G_{NF}}$ are marked, thus $L(\mathbf{G_{NF}}) = L_m(\mathbf{G_{NF}})$.

It is thus sufficient to show that: $ss' \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$

As $ss' \in L_m(\mathbf{G})$ by (T6.4), we have $ss' \in L(\mathbf{G})$, since $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$.

From (T6.4), we have: $ss' \notin L_{\Delta F} \wedge ss' \in L_{NF}$

Applying Proposition 2, we can conclude that:

$ss' \in L(\mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}) = L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$

$\Rightarrow ss' \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$

We thus have that $\mathbf{G}'$ is nonblocking, as required.

Part B) Show ($\Leftarrow$)

Assume $\mathbf{G}'$ is nonblocking. (T6.5)

Must show implies $\mathbf{S}$ and $\mathbf{G}$ are FT consistent (follows from initial assumptions) and that:

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))\, s \notin L_{\Delta F} \wedge s \in L_{NF} \Rightarrow$

$(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \wedge ss' \notin L_{\Delta F} \wedge ss' \in L_{NF}$

Let $s \in L(\mathbf{S}) \cap L(\mathbf{G})$. (T6.6)

Assume $s \notin L_{\Delta F} \wedge s \in L_{NF}$. (T6.7)

To apply (T6.5), we need to show: $s \in L(\mathbf{G}') = L(\mathbf{S}) \cap L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$

As we have $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ from (T6.6), we only still need to show:

$s \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$

By (T6.6) and (T6.7), we can apply Proposition 2, and conclude:

$s \in L(\mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}) = L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$

We thus have $s \in L(\mathbf{G}')$. As $\mathbf{G}'$ is nonblocking, we can conclude: $(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{G}')$

$\Rightarrow ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}})$, by (T6.1)

We thus have $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})$, and only need to show that $ss' \notin L_{\Delta F} \wedge ss' \in L_{NF}$.

We first note that we have $ss' \in L(\mathbf{G})$, as $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$.

We next note that $ss' \in P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}})$ implies $ss' \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$ as every state is marked in $\mathbf{G_{\Delta F}}$, by Algorithm 1.

We also note that $ss' \in P_F^{-1} L_m(\mathbf{G_{NF}})$ implies $ss' \in P_F^{-1} L(\mathbf{G_{NF}})$ as every state is marked in $\mathbf{G_{NF}}$, by Algorithm 2.

$\Rightarrow ss' \in L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}}) = L(\mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}})$

We can now conclude by Proposition 2 that $ss' \notin L_{\Delta F}$ and that $ss' \in L_{NF}$.

We thus conclude that $\mathbf{S}$ and $\mathbf{G}$ are N-fault tolerant nonblocking.

We can thus conclude by parts (A) and (B), that $\mathbf{S}$ and $\mathbf{G}$ are N-fault tolerant nonblocking iff $\mathbf{G}'$ is nonblocking. $\square$

Theorem 7:

Proof. Assume initial conditions for theorem.

We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Theorem 5. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.

Must show $\mathbf{S}$ and $\mathbf{G}$ are non-repeatable N-fault tolerant nonblocking $\iff$ $\mathbf{G'}$ is nonblocking.

From Algorithm 11, we have: $\mathbf{G'} = \mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{G_{F,1}}||\ldots||\mathbf{G_{F,m}}||\mathbf{S}$

From Algorithm 1, we know that $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$. From Algorithm 2, we know that $\mathbf{G_{NF}}$ is defined over Σ_F, and from Algorithm 3, we know that $\mathbf{G_{F,i}}$ is defined over $\Sigma_{F_i}, i = 1, \ldots, m$.

Let $P_{\Delta F} : \Sigma^* \to \Sigma^*_{\Delta F}$, $P_F : \Sigma^* \to \Sigma^*_F$, and $P_{F_i} : \Sigma^* \to \Sigma^*_{F_i}$, $i = 1, \ldots, m$, be natural projections.

As $\mathbf{G}$ and $\mathbf{S}$ are defined over Σ, we have that $L(\mathbf{G'}) = L(\mathbf{S}) \cap L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L(\mathbf{G_{F,m}})$ and $L_m(\mathbf{G'}) = L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P^{-1}_{\Delta F} L_m(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L_m(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L_m(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L_m(\mathbf{G_{F,m}})$. (T7.1)

Part A) Show ($\Rightarrow$)

Assume $\mathbf{S}$ and $\mathbf{G}$ are non-repeatable N-fault tolerant nonblocking. (T7.2)

Must show implies: $(\forall s \in L(\mathbf{G'}))(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{G'})$

Let $s \in L(\mathbf{G'})$.

$\Rightarrow s \in L(\mathbf{S}) \cap L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L(\mathbf{G_{F,m}})$ (T7.3)

$\Rightarrow s \in L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L(\mathbf{G_{F,m}})$

$\Rightarrow s \in L(\mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{G_{F,1}}||\ldots||\mathbf{G_{F,m}})$

We can thus apply Proposition 3 and conclude that: $s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF}$.

As we have $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ from (T7.3), we can apply (T7.2) and conclude that:

$(\exists s' \in \Sigma^*)\, ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \wedge ss' \notin L_{\Delta F} \cup L_{NRF} \wedge ss' \in L_{NF}$ (T7.4)

We now need to show that $ss' \in L_m(\mathbf{G'})$.

Sufficient to show:

$ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P^{-1}_{\Delta F} L_m(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L_m(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L_m(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L_m(\mathbf{G_{F,m}})$.

From (T7.4), we have $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})$, so only need to show:

$ss' \in P^{-1}_{\Delta F} L_m(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L_m(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L_m(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L_m(\mathbf{G_{F,m}})$

We note from Algorithm 1 that as all states in $\mathbf{G_{\Delta F}}$ are marked, we have $L(\mathbf{G_{\Delta F}}) = L_m(\mathbf{G_{\Delta F}})$. From Algorithm 2, we have that all states in $\mathbf{G_{NF}}$ are marked, thus $L(\mathbf{G_{NF}}) = L_m(\mathbf{G_{NF}})$. From Algorithm 3, we have that all states in $\mathbf{G_{F,i}}$ are marked, thus $L(\mathbf{G_{F,i}}) = L_m(\mathbf{G_{F,i}}), i = 1, \ldots, m$.

It is thus sufficient to show:

$ss' \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L(\mathbf{G_{F,1}}) \cap \cdots \cap P^{-1}_{F_m} L(\mathbf{G_{F,m}})$

As $ss' \in L_m(\mathbf{G})$ by (T7.4), we have $ss' \in L(\mathbf{G})$, since $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$.

From (T7.4), we have: $ss' \notin L_{\Delta F} \cup L_{NRF} \wedge ss' \in L_{NF}$

Applying Proposition 3, we can conclude that: $ss' \in L(\mathbf{G}||\mathbf{G_{\Delta F}}||\mathbf{G_{NF}}||\mathbf{G_{F,1}}||\ldots||\mathbf{G_{F,m}})$

$\Rightarrow ss' \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_F L(\mathbf{G_{NF}}) \cap P^{-1}_{F_1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P^{-1}_{F_m} L(\mathbf{G_{F,m}})$

We thus have that $\mathbf{G'}$ is nonblocking, as required.

Part B) Show ($\Leftarrow$)

Assume $\mathbf{G}'$ is nonblocking. (T7.5)

Must show implies $\mathbf{S}$ and $\mathbf{G}$ are FT consistent (follows from initial assumptions) and that:

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G}))\, s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF} \Rightarrow$

$(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \wedge ss' \notin L_{\Delta F} \cup L_{NRF} \wedge ss' \in L_{NF}$

Let $s \in L(\mathbf{S}) \cap L(\mathbf{G})$. (T7.6)

Assume $s \notin L_{\Delta F} \cup L_{NRF} \wedge s \in L_{NF}$. (T7.7)

To apply (T7.5), we need to show:

$s \in L(\mathbf{G}') = L(\mathbf{S}) \cap L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}}) \cap P_{F_1}^{-1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P_{F_m}^{-1} L(\mathbf{G_{F,m}})$

As we have $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ from (T7.6), we only still need to show:

$s \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}}) \cap P_{F_1}^{-1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P_{F_m}^{-1} L(\mathbf{G_{F,m}})$.

By (T7.6) and (T7.7), we can apply Proposition 3 and conclude:

$s \in L(\mathbf{G} || \mathbf{G_{\Delta F}} || \mathbf{G_{NF}} || \mathbf{G_{F,1}} || \ldots || \mathbf{G_{F,m}})$

$\Rightarrow s \in L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}}) \cap P_{F_1}^{-1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P_{F_m}^{-1} L(\mathbf{G_{F,m}})$

We thus have $s \in L(\mathbf{G}')$. As $\mathbf{G}'$ is nonblocking, we can conclude: $(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{G}')$

$\Rightarrow ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}}) \cap P_{F_1}^{-1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P_{F_m}^{-1} L(\mathbf{G_{F,m}})$, by (T7.1)

We thus have $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})$ and only need to show that $ss' \notin L_{\Delta F} \cup L_{NRF}$ and $ss' \in L_{NF}$.

We first note that we have $ss' \in L(\mathbf{G})$, as $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$.

We next note that $ss' \in P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}})$ implies $ss' \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$ as every state is marked in $\mathbf{G_{\Delta F}}$, by Algorithm 1.

We note that $ss' \in P_F^{-1} L_m(\mathbf{G_{NF}})$ implies $ss' \in P^{-1} L(\mathbf{G_{NF}})$ as every state is marked in $\mathbf{G_{NF}}$, by Algorithm 2.

Also, we note that $ss' \in P_{F_i}^{-1} L_m(\mathbf{G_{F,i}})$ implies $ss' \in P_{F_i}^{-1} L(\mathbf{G_{F,i}})$ as every state is marked in $\mathbf{G_{F,i}}$, $i = 1, \ldots, m$, by Algorithm 3.

$\Rightarrow ss' \in L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_F^{-1} L(\mathbf{G_{NF}}) \cap P_{F_1}^{-1} L(\mathbf{G_{F,1}}) \cap \ldots \cap P_{F_m}^{-1} L(\mathbf{G_{F,m}})$

$\Rightarrow ss' \in L(\mathbf{G} || \mathbf{G_{\Delta F}} || \mathbf{G_{NF}} || \mathbf{G_{F,1}} || \ldots || \mathbf{G_{F,m}})$

We can now conclude by Proposition 3 that: $ss' \notin L_{\Delta F} \cup L_{NRF}$, and $ss' \in L_{NF}$

We thus conclude that $\mathbf{S}$ and $\mathbf{G}$ are non-repeatable N-fault tolerant nonblocking.

We can thus conclude by parts (A) and (B), that $\mathbf{S}$ and $\mathbf{G}$ are non-repeatable N-fault tolerant nonblocking iff $\mathbf{G}'$ is nonblocking. □

Theorem 8:

Proof. Assume initial conditions for theorem.

We first note that if $m = 0$, we have $\Sigma_F = \emptyset$ and the proof is identical to the proof of Theorem 5. We can thus assume $m \geq 1$ for the rest of the proof without any loss of generality.

Must show $\mathbf{S}$ and $\mathbf{G}$ are resettable fault tolerant nonblocking $\iff$ $\mathbf{G}'$ is nonblocking.

From Algorithm 12, we have: $\mathbf{G}' = \mathbf{G} || \mathbf{G_{\Delta F}} || \mathbf{G_{TF,1}} || \ldots || \mathbf{G_{TF,m}} || \mathbf{S}$

From Algorithm 1, we know that $\mathbf{G_{\Delta F}}$ is defined over $\Sigma_{\Delta F}$. From Algorithm 4, we know that $\mathbf{G_{TF,i}}$ is defined over $\Sigma_{F_i} \cup \Sigma_{T_i}$, $i = 1, \ldots, m$.

Let $P_{\Delta F} : \Sigma^* \to \Sigma_{\Delta F}^*$ and $P_{TF_i} : \Sigma^* \to (\Sigma_{F_i} \cup \Sigma_{T_i})^*, i = 1, \ldots, m$, be natural projections.

As $\mathbf{G}$ is defined over Σ, we have that $L(\mathbf{G}') = L(\mathbf{S}) \cap L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_{TF_1}^{-1} L(\mathbf{G_{TF,1}})$

$\cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$ and $L_m(\mathbf{G}') = L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P^{-1}_{\Delta F} L_m(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L_m(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L_m(\mathbf{G_{TF,m}})$. (T8.1)

Part A) Show ($\Rightarrow$)

Assume $\mathbf{S}$ and $\mathbf{G}$ are resettable fault tolerant nonblocking. (T8.2)

Must show implies: $(\forall s \in L(\mathbf{G}'))(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{G}')$

Let $s \in L(\mathbf{G}')$.

$\Rightarrow s \in L(\mathbf{S}) \cap L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$ (T8.3)

$\Rightarrow s \in L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap$
$P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$

$\Rightarrow s \in L(\mathbf{G} || \mathbf{G_{\Delta F}} || \mathbf{G_{TF,1}} || \ldots || \mathbf{G_{TF,m}})$

We can thus apply Proposition 4 and conclude:

$s \notin L_{\Delta F} \cup L_{TF}$

As we have $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ from (T8.3), we can apply (T8.2) and conclude:

$(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \wedge ss' \notin L_{\Delta F} \cup L_{TF}$ (T8.4)

We now need to show that $ss' \in L_m(\mathbf{G}')$.

Sufficient to show:

$ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P^{-1}_{\Delta F} L_m(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L_m(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L_m(\mathbf{G_{TF,m}})$

From (T8.4), we have $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})$, so only need to show $ss' \in P^{-1}_{\Delta F} L_m(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L_m(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L_m(\mathbf{G_{TF,m}})$.

We note from Algorithm 1 that as all states in $\mathbf{G_{\Delta F}}$ are marked, we have $L(\mathbf{G_{\Delta F}}) = L_m(\mathbf{G_{\Delta F}})$. From Algorithm 4, we have that all states in $\mathbf{G_{TF,i}}$ are marked, $i = 1, \ldots, m$, thus $L(\mathbf{G_{TF,i}}) = L_m(\mathbf{G_{TF,i}})$.

It is thus sufficient to show:

$ss' \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$

As $ss' \in L_m(\mathbf{G})$ by (T8.4), we have $ss' \in L(\mathbf{G})$, since $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$.

Also from (T8.4), we have: $ss' \notin L_{\Delta F} \cup L_{TF}$

Applying Proposition 4, we can conclude that: $ss' \in L(\mathbf{G} || \mathbf{G_{\Delta F}} || \mathbf{G_{TF,1}} || \ldots || \mathbf{G_{TF,m}})$

$\Rightarrow ss' \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$

We thus have that $\mathbf{G}'$ is nonblocking, as required.

Part B) Show ($\Leftarrow$)

Assume $\mathbf{G}'$ is nonblocking. (T8.5)

Must show implies $\mathbf{S}$ and $\mathbf{G}$ are FT consistent (follows from initial assumptions) and that:

$(\forall s \in L(\mathbf{S}) \cap L(\mathbf{G})) s \notin L_{\Delta F} \cup L_{TF} \Rightarrow (\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \wedge ss' \notin L_{\Delta F} \cup L_{TF}$

Let $s \in L(\mathbf{S}) \cap L(\mathbf{G})$. (T8.6)

Assume $s \notin L_{\Delta F} \cup L_{TF}$. (T8.7)

To apply (T8.5), we need to show:

$s \in L(\mathbf{G}') = L(\mathbf{S}) \cap L(\mathbf{G}) \cap P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$

As we have $s \in L(\mathbf{S}) \cap L(\mathbf{G})$ from (T8.6), we only still need to show:

$s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap P^{-1}_{TF_1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P^{-1}_{TF_m} L(\mathbf{G_{TF,m}})$

By (T8.6) and (T8.7), we can conclude by Proposition 4: $s \in L(\mathbf{G} || \mathbf{G_{\Delta F}} || \mathbf{G_{TF,1}} || \ldots || \mathbf{G_{TF,m}})$

$\Rightarrow s \in P^{-1}_{\Delta F} L(\mathbf{G_{\Delta F}}) \cap \mathbf{P^{-1}_{TF_1} L(G_{TF,1})} \cap \ldots \cap \mathbf{P^{-1}_{TF_m} L(G_{TF,m})}$

We thus have $s \in L(\mathbf{G}')$. As $\mathbf{G}'$ is nonblocking, we can conclude: $(\exists s' \in \Sigma^*) ss' \in L_m(\mathbf{G}')$
$\Rightarrow ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G}) \cap P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}}) \cap P_{TF_1}^{-1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P_{TF_m}^{-1} L(\mathbf{G_{TF,m}})$, by (T8.1)
We thus have $ss' \in L_m(\mathbf{S}) \cap L_m(\mathbf{G})$ and only need to show that $ss' \notin L_{\Delta F} \cup L_{TF}$.
We first note that we have $ss' \in L(\mathbf{G})$, as $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$.
We next note that $ss' \in P_{\Delta F}^{-1} L_m(\mathbf{G_{\Delta F}})$ implies $ss' \in P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}})$ as every state is marked in $\mathbf{G_{\Delta F}}$, by Algorithm 1.
Also, we note that $ss' \in P_{TF_i}^{-1} L_m(\mathbf{G_{TF,i}})$ implies $ss' \in P_{TF_i}^{-1} L(\mathbf{G_{TF,i}})$ as every state is marked in $\mathbf{G_{TF,i}}$, by Algorithm 4, for $i = 1, \ldots, m$.
$\Rightarrow ss' \in L(\mathbf{G}) \cap P_{\Delta F}^{-1} L(\mathbf{G_{\Delta F}}) \cap P_{TF_1}^{-1} L(\mathbf{G_{TF,1}}) \cap \ldots \cap P_{TF_m}^{-1} L(\mathbf{G_{TF,m}})$
$\Rightarrow ss' \in L(\mathbf{G} || \mathbf{G_{\Delta F}} || \mathbf{G_{TF,1}} || \ldots || \mathbf{G_{TF,m}})$
We can now conclude by Proposition 4 that: $ss' \notin L_{\Delta F} \cup L_{TF}$
We thus conclude that $\mathbf{S}$ and $\mathbf{G}$ are resettable fault tolerant nonblocking.

We can thus conclude by parts (A) and (B), that $\mathbf{S}$ and $\mathbf{G}$ are resettable fault tolerant nonblocking iff $\mathbf{G}'$ is nonblocking. □