Timing Fault Models for Systems with Multiple Timers

M. Ümit Uyar[1], Yu Wang[1], Samrat S. Batth[1], Adriana Wise[2], M. A. Fecko[3]

[1] The City College of the City University of New York, New York, USA
[2] Department of Computer Science, Graduate Center, CUNY, New York, USA
[3] Applied Research Area, Telcordia Technologies Inc., New Jersey, USA

Abstract. Multiple timing faults, although detectable individually, can hide each other's faulty behavior making the faulty system indistinguishable from a non-faulty one. A set of graph augmentations are introduced for single timing faults. The fault detection capability of the augmentations is analyzed in the presence of multiple timing faults and shown that multiple occurrences of a class of timing faults can be detected.
Key words: Conformance Testing; Timer Constraints; Multiple Faults; Fault Modeling; Timed Automata

1 Introduction

This paper analyzes the fault detection capability of the timed FSM model introduced in Ref. [7] in the presence of multiple timing faults. It is shown here that multiple timing faults, although detectable individually, can hide each other's faulty behavior thereby making the faulty system indistinguishable from a non-faulty one. A set of graph augmentations are introduced for single timing faults. It is shown that the augmentations for single faults can also detect the presence of multiple faults occurring simultaneously.

Fault coverage has been studied mostly with respect to transfer/output faults for FSMs [1, 9, 11, 15]. Petrenko et al. [9] investigate fundamental underlying concepts of fault coverage analysis, whose primary focus is protocol conformance testing. The detection of such faults, which is not part of the timing-fault analysis, depends on the adopted conformance relation, the underlying fault models, and the state verification method [4, 8, 10, 13]. If a timing fault results in a transfer/output fault, we assume that it is detected with high probability under the widely accepted assumption that the faults do not increase the number of states in an implementation under test (IUT).

The related work on testing systems with timing dependencies focuses on testing Timed Automata (TA) [2], with a theoretical framework in Ref. [12] achieving a provably complete test coverage at the expense of a prohibitively large number of test cases. Dssouli et al. [5, 6] introduce a method based on the state characterization technique using a timed extension of the Wp-method [8]. The technique formulates fault models for timed systems by considering time specific one-clock and multi-clock timing faults in addition to FSM-like transfer/output faults. The aim of a complete test coverage is relaxed—by choosing a proper granularity, a “good” fault coverage is achieved with reasonably long test sequences. Dssouli et al. are the first to present a classification of timing faults [6], and formally prove that their technique detects all single faults of a given type [5]. None of the above techniques are shown to have the ability to detect multiple simultaneous timing faults. A major contribution of this paper is a formal analysis of such a fault detection capability for the testing methodology introduced in Ref. [7].

Section 2 of this paper gives the basic definitions. The simplified version of the timed FSM model of [7] is given in Section 3. Single and multiple timing faults are discussed in Sections 4 and 5, respectively.

2 Definitions

A communicating protocol can be modeled as a Finite State Machine (FSM) represented by a directed graph $G(V, E)$. Vertex set V and edge set E represent the states and transitions triggered by events of a system, respectively. For time-related FSM, FSM can be extended to consider of a set of timers $\mathcal{T}$ that may be arbitrarily started or stopped.

Timed FSM is a tuple $M = (V, A, O, \mathcal{T}, E, v_0)$ where V is a finite set of states, $v_0 \in V$ is the initial state, A is a finite set of inputs, O is a finite set of outputs, $\mathcal{T}$ is a finite set of timers, and $E \subseteq V \times (A \times \mathcal{T} \times O) \times V$ is a set of transitions $V \times A \times \mathcal{T} \longrightarrow O \times V$.

In the presence of timers, an FSM becomes an Extended Finite State Machine (EFSM). Timer-related variables will appear in addition to the variables from the tuple above, in the form of conditions $\langle \mathbf{t}_j \rangle$ on the timer variables and of actions $\{\mathbf{t}_j\}$on variable values. A tuple $e_i = (v_p, v_q, a_i, o_i, \langle \mathbf{t}_j \rangle, \{\mathbf{t}_j\})$ is a transition $e_i \in E$, where v_p is a current state, v_q is a next state, a_i is the input defined in current state v_p or in current transition $v_p \xrightarrow{e_i} v_q$, o_i is the output from current transition $v_p \xrightarrow{e_i} v_q$, $\mathbf{t}_j$ is a vector of timer variables, $\langle \mathbf{t}_j \rangle$ are the conditions on time-related variables, and $\{\mathbf{t}_j\}$ are the actions which update time-related variables.

A timer $T_j \in \mathcal{T}$ can be defined with a timer vector $\mathbf{t}_j = (T_j, D_j, f_j, L_p)$ where $T_j \in \{0, 1\}$ is a timer running status variable denoted by a boolean variable, $D_j \in R^{\circ+}$ is a time-characteristic variable that indicates the length of timer T_j, $f_j \in R^{\infty}$ is a time-keeping variable that indicates the time elapsed since timer T_j started, and $L_p \in \{0, 1\}$ is a flow enforcing variable that forces the test sequence to traverse the augmented graph according to model specific rules. Timer $T_j \in \mathcal{T}$ is expired iff $\langle (T_j == 1) \wedge (f_j \geqslant D_j) \rangle$ and is running iff $\langle (T_j == 1) \wedge (f_j < D_j) \rangle$.

$T_j == 1$ (depicted as T_j henceforth) denotes a timer is running and $T_j == 0$ (depicted as $\neg T_j$ henceforth) denotes a timer is not running (i.e., stopped, expired or not started yet). D_j is the length for T_j and $\forall\ f_j \in \mathbf{Z}^{\infty}$ is the time elapsed since its start. When T_j has just started, $f_j := 0$, and $f_j := -\infty$ if T_j is not running. Over an edge e_i the value of f_j is increased by the cost c_i of e_i as $f_j := f_j + c_i$. Once f_j becomes $(f_j \geqslant D_j)$, T_j is said to be expired or timed-out. The difference of $(D_j - f_j)$ represents the remaining time until T_j's expiry. L_p is a flow enforcing variable where $L_p = 0$ implies that no transition can leave the current state v_p and $L_p = 1$ means that all transitions are allowed to leave v_p.

For $h_k = (v_k, v_{k+1}, a_k, o_k, \langle \mathbf{t}_j \rangle, \{\mathbf{t}_j\})$ $(\forall h_k \in E, \forall v_k \in V, \forall k \in \mathbf{Z}^+)$, a finite transition sequence is represented as $\rho = h_1, \cdots h_k, h_{k+1}, \cdots h_n$ in the graph G associated with M. For any $\forall k \in [1, n-1]$, h_k was progressed before h_{k+1}.

Assume that there are K running timers: $\{T_1, T_2, \cdots T_j, \cdots T_K\} \subset \mathcal{T}$. Then edge cost $c_i \in R^{\circ+}$ is the amount of time required to completely traverse the current edge e_i. Timeout transition $e_i = (v_p, v_q, a_i, o_i, \langle \mathbf{t}_j \rangle, \{\mathbf{t}_j\})$ is triggered by T_j expiry and it becomes feasible if at least one of the running timers T_j expires, $\forall T_k \neq T_j$, which can be described as follows:

$$\langle \mathbf{t}_j \rangle : \langle T_j \wedge (f_j \geqslant D_j) \wedge T_k \wedge (f_k < D_k) \wedge (D_j - f_j < D_k - f_k) \rangle$$
$$\{\mathbf{t}_j\} : \{T_j := 0; \quad f_j := -\infty; \quad T_k := T_k; \quad f_k := f_k + c_i\} \quad k \in \{1, 2, \cdots K, \forall k \neq j\}$$

A transition in which timer T_j, $\forall j \in [1, K]$, does not expire is defined as a non-timeout transition. A timer can be started in an action as follows:

$$\langle \mathbf{t}_j \rangle : \langle \neg T_j \wedge T_k \wedge (f_k < D_k) \rangle$$
$$\{\mathbf{t}_j\} : \{T_j := 1; \quad f_j := 0; \quad T_k := T_k; \quad f_k := f_k + c_i\} \quad k \in \{1, 2, \cdots K\}, \forall k \neq j.$$

A timer can be stopped as follows:

$$\langle \mathbf{t}_j \rangle : \langle T_j \wedge (f_j < D_j) \wedge T_k \wedge (f_k < D_k) \rangle$$
$$\{\mathbf{t}_j\} : \{T_j := 0; \quad f_j := -\infty; \quad T_k := T_k; \quad f_k := f_k + c_i\} \quad k \in \{1, 2, \cdots K\}, \forall k \neq j.$$

3 Modeling Timed FSM

To simplify the test generation from timed FSM models (which are essentially EFSMs due to the timing variables as described in Section 2), we introduce a graph augmentation for conversion of G to G' as follows:

Step (i): All the self loops in G are represented as ordinary (i.e., state-to-state) edges in G';

Step (ii): For every state v_p in G, an additional state called v'_p is introduced in G', which becomes the ending state for all of self-loops defined in v_p;

Step (iii): For self-loops of v_p in G, the return from v'_p to v_p is ensured by the introduction of an additional edge called return edge e_p^{ret} in G':

$$e_{p,k} = (v_p, v'_p) \text{ (self-loop converted as state-to-state edge)}$$
$$e_p^{ret} = (v'_p, v_p) \text{ (return edge from replica state } v'_p\text{)}$$

Step (iv): A new *observer state* is appended to v_p in G', namely v''_p. This state can be reached from and to v_p via additional edges $e_{p,obs}, e_{p,wait}$ and $e_{p,obs}^{ret}$, respectively:

$$e_{p,obs} = (v_p, v''_p) \text{ (observer edge)}$$
$$e_{p,wait} = (v_p, v''_p) \text{ (wait edge)}$$
$$e_{p,obs}^{ret} = (v''_p, v_p) \text{ (return edge from observer state)}$$

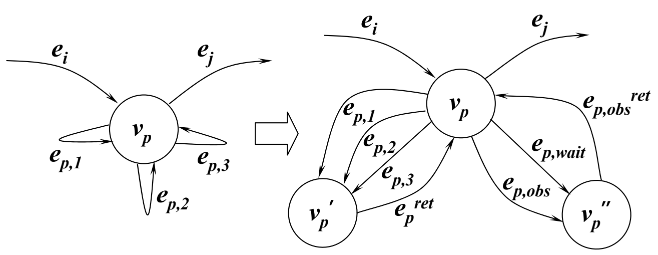

Fig. 1. Self-loops and traversal enforcement rules for v_p in G into v_p, v_p' and v_p'' in G'

In this model, the new states in G' are introduced to convert the self-loop transitions as state-to-state transitions. The role of the observer state is to "consume" pending timeouts and enable outgoing edges by setting L_p to 1. Figure 1 shows, for state v_p, an example conversion of self-loops to state-to-state transitions and the introduction of observer states/edges by our model. Augmented graph G' will contain two types of transitions, as defined below:

Type 1 *Timeout transition e_i^j defined as the transition triggered by the expiry of timer T_j. (Note that in the original graph G, e_i^j corresponds to either a state-to-state edge or a self-loop).*

Type 2 *Non-timeout transition e_i, which may start/stop a timer, may be a regular, non-timeout, state-to-state transition or may have been converted from a non-timeout self-loop transition.*

3.1 Edge Conditions and Actions for New Model

The original edge conditions and the actions of G are modified by appending timer-related conditions and actions, as described below.

Edge conditions that need to be satisfied before an edge can be traversed are formulated using the three variable types described in Section (2): *timer status variables* (T_j—on or off), *time-keeping variables* (D_j—timer length, f_j—time elapsed) and *flow-enforcing variables* (L_p—edge traversal control). Below are the edge conditions used by our model:

A *Type 1* (timeout) transition is feasible if all of the following conditions are true during the traversal:

- at least one of the running timers expires (for any two running timers, T_j and T_k, either T_j or T_k expires, and thus enables the timeout edge): $\big((T_j \wedge \neg T_k) \vee (\neg T_j \wedge T_k)\big) \;\; \forall T_j \neq T_k$
- the timer that expired was the timer with the least remaining time (i.e., if some T_k was also running, and if T_j's remaining time was less, then it was T_j that expired): $T_j \wedge \big(\neg T_k \vee \big(T_k \wedge (D_j - f_j < D_k - f_k)\big)\big) \;\; \forall T_j \neq T_k$
- the flow-enforcing variable is set as follows:

$$L_p == \begin{cases} 0, \text{if the edge was a timeout self loop edge in } G \\ 1, \text{if the edge was a timeout state-to-state edge in } G \end{cases}$$

These three components of a *Type 1* edge condition can be, therefore, combined and formalized as:

- for a converted edge in G' (i.e., a self-loop edge in G): $\big\langle T_j \wedge (f_j \geqslant D_j) \wedge T_k \wedge (f_k < D_k) \wedge (D_j - f_j < D_k - f_k) \wedge (L_p == 0)\big\rangle$
- for an original edge in G' (i.e., a state-to-state edge in G): $\big\langle T_j \wedge (f_j \geqslant D_j) \wedge T_k \wedge (f_k < D_k) \wedge (D_j - f_j < D_k - f_k) \wedge (L_p == 1)\big\rangle$

The above equations imply that before a timeout transition, T_j should be still running, remaining time should be the least among all other running timers and the flow-enforcing variable is appropriately set for either a converted or an original edge in G'. Any nondeterminism due to multiple timeouts can be detected during test-sequence generation, e.g., if tm_j and tm_k are to expire simultaneously, then $(D_j - f_j = D_k - f_k)$ and their conditions cannot be satisfied.

Similarly, during the traversal, a *Type 2* (non-timeout) transition becomes feasible if both of the following conditions are true:

- either there is no running timer started in a previous transition (there may be a timer started on the current transition): $\langle \neg T_j \rangle$; or, if there is, it did not expire over a previous transition (time variable f_j of running timer T_j is less than timer's length D_j): $\langle T_j \wedge (f_j < D_j)\rangle$
- the flow-enforcing variable is set as:

$$L_p == \begin{cases} 0, \text{if the edge was a non-timeout self-loop in } G \\ 1, \text{if the edge was a non-timeout state-to-state edge in } G \end{cases}$$

Therefore, the time conditions for *Type 2* edges can be formalized as follows:

- for a converted edge in G' (i.e., a self-loop edge in G): $\big\langle\big(\neg T_j \vee (f_j < D_j)\big) \wedge (L_p == 0)\big\rangle$
- for an original edge in G' (i.e., a state-to-state edge in G): $\big\langle\big(\neg T_j \vee (f_j < D_j)\big) \wedge (L_p == 1)\big\rangle$

The time condition for the wait edge $e_{p,wait}$ and observer edge $e_{p,obs}$, from the original state v_p to the observer state v''_p is formulated as: $\big\langle L_p == 0\big\rangle$.

The return edges (i.e., e^{ret}_p and $e^{ret}_{p,obs}$) added by the graph augmentation to G' are no-cost edges with time condition as true: $\big\langle 1\big\rangle$.

Action list can be executed by an edge whose traversal was determined by its time condition being satisfied. Such an edge may proceed and update all variables that changed during the current transition accordingly:

- If a timer expires, the timeout edge will reset the status variable T_j to 0 and the time-keeping variable to $-\infty$: $\big\{T_j := 0; \quad f_j := -\infty\big\}$
- If a timer started on a previous transition is still running, the current edge e_i will update its value with its cost c_i (which may bring $f_j \geqslant D_j$, and thus timeout T_j and trigger a timeout transition): $\big\{f_j := f_j + c_i\big\}$
- If a timer is started on the current transition, the current action list will initialize the timer state T_j to 1 and the time-keeping variable f_j to 0: $\big\{T_j := 1; \quad f_j := 0\big\}$
- The flow-enforcing variable L_p is also set by every edge according to its type:

$$L_p := \begin{cases} 1, \text{set by observer edge to allow traversal of state-to-state edges} \\ 0, \text{set by either } \textit{Type 1} \text{ or } \textit{Type 2} \text{ edges} \end{cases}$$

Each edge type will perform a subset of the above listed actions, according to its specifics, as follows:

- *Type 1* (timeout) edge: $\{T_j := 0;\ \ f_j := -\infty;\ \ T_k := T_k;\ \ f_k := f_k + c_i;\ \ L_p := 0\}$
- *Type 2* (non-timeout) edge: $\{f_k := f_k + c_i;\ \ L_p := 0\}$ if the edge starts no timers; $\{T_j := 1;\ \ f_j := 0;\ \ T_k := T_k;\ \ f_k := f_k + c_i;\ \ L_p := 0\}$ if the edge starts timer T_j
- *Wait* (artificial) edge: $\{f_j := f_j + 1\}$ or $\{f_j := f_j + (D_j - f_j)\}$ where $D_j - f_j$ is the remaining time of timer T_j to timeout
- *Observer* (artificial) edge: $\{L_p := 1\}$
- *Return* (artificial) edge: $\{\ \}$ (i.e., there is no actions for this edge)

Since both edge types, namely *Type 1* and *Type 2*, disable outgoing transitions by setting $L_p := 0$ the only edges whose actions will enable these transitions are the artificially-created observer edges.

4 Modeling Timing Faults

In general, timing faults in an IUT can be classified into: (*i*) 1-clock interval faults, (*ii*) n-clock interval faults (introduced by Dssouli et al. [5, 6]), and (*iii*) incorrect settings of timer lengths. The goal is to detect such faults during testing through special-purpose timers and graph augmentations that force a test sequence to take a different path for a faulty IUT than for the conformant one.

In our model during the testing of transition $e_i = (v_p, v_q, a_i, o_i, \langle \mathbf{t}_j \rangle, \{\mathbf{t}_j\})$, after input a_i is applied, the expected output o_i should be generated no later than θ time units, $\theta \in \mathbf{R}^+$. If there is no output observed in θ time units (represented as $\neg o_i$) or output o_i is observed after θ time units, a fault occurs. The θ time units is part of a test harness rather than the IUT.

4.1 1-Clock Interval Faults

1-Clock Interval Faults are related to timing conflicts due to one clock/timer regardless of other concurrent clocks/timers. Unacceptable input timing (i.e., an input may be 'rushed' or 'delayed') results either in an unacceptable output value for a transition or unexpected output timing (i.e., an output may be 'rushed' or 'delayed'). 1-clock interval faults occur either when at least one input interval boundary is violated in the IUT or no interval boundary is modified but no output is observed.

Timing Requirement: Transition $e_i = (v_p, v_q, a_i, o_i, \langle \mathbf{t}_j \rangle, \{\mathbf{t}_j\})$ can correctly trigger only if applied input a_i is within the required time interval $[\alpha, \beta]$ measured from the traversal of h_k—an edge prior to e_i in a test sequence.

Based on this requirement, two faults, namely *Timing Faults I* and *II*, can be defined as follows:

Timing Fault I: Input a_i is applied either too early ($\delta' < \alpha$) or too late ($\delta' > \beta$), but output o_i may still be observed and state v_q be verified in no later than θ time units from the instance input a_i is applied.

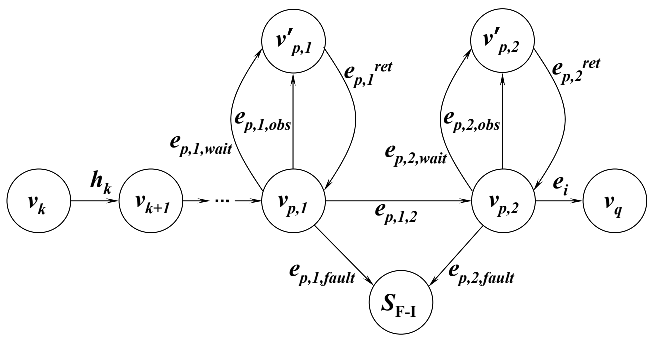

Fig. 2. Augmenting state v_p for Timing Fault I detection

Timing Fault II: Input a_i is applied within the required time interval $[\alpha, \beta]$, but either the output is not observed (i.e., $\neg o_i$) or state v_q cannot be verified in less than $\delta + \theta$ time units. The detection of Fault II has not been included in the analysis presented in this paper since it has been handled by transfer fault detection models reported in literature [9].

Graph Augmentation to Detect Timing Fault I: The modeling of 1-clock timing requirement for an edge $e_i = (v_p, v_q, a_i, o_i, \langle \mathbf{t}_j \rangle, \{\mathbf{t}_j\})$, is accomplished by using two *special purpose timers* and creating the so-called *observer states/edges*. The special purpose timers are called T_α and T_β with lengths $D_\alpha = \alpha$ and $D_\beta = \beta$ time units, respectively, where $\alpha < \beta$. Note that timers T_α and T_β are not the part of the IUT, but maintained by the test harness run by the tester.

The edge e_i triggers only after input a_i is applied within time interval $[\alpha, \beta]$ (i.e., after timer T_α but before timer T_β expires), and in its action stops timer T_β. Therefore, in our augmentation, the modified timing conditions for h_k (which starts T_α and T_β timers) and e_i are as follows:

$$\begin{aligned} h_k &: \big\langle \neg T_\alpha \wedge \neg T_\beta \big\rangle & &\big\{ T_\alpha := 1; \quad f_\alpha := 0; \quad T_\beta := 1; \\ & & &\quad f_\beta := 0 \big\} \\ e_i &: \big\langle \neg T_\alpha \wedge T_\beta \wedge (f_\beta \in [\alpha, \beta]) \wedge (L_p == 1) \big\rangle & &\big\{ T_\beta := 0; \quad f_\beta := -\infty; \quad L_p := 0 \big\} \end{aligned}$$

Additionally, v_p (starting state of e_i) is replaced by two new states, $v_{p,1}$ and $v_{p,2}$, connected by a new edge $e_{p,1,2}$ from $v_{p,1}$ to $v_{p,2}$; the original incoming and outgoing edges of v_p are connected to $v_{p,1}$ and $v_{p,2}$, respectively. The time condition for $e_{p,1,2}$ is the expiry of T_α with the cost of zero:

$$e_{p,1,2} : \big\langle T_\alpha \wedge (f_\alpha \geqslant \alpha) \wedge T_\beta \wedge (L_p == 1) \big\rangle \quad \big\{ T_\alpha := 0; \quad f_\alpha := -\infty; \quad L_p := 0 \big\}$$

Two new observer states, namely $v'_{p,1}$ and $v'_{p,2}$, with their associated observer edges, $e_{p,1,obs}$ and $e_{p,2,obs}$, are appended to $v_{p,1}$ and $v_{p,2}$, respectively. The new wait edges $e_{p,1,wait}$ from $v_{p,1}$ to $v'_{p,1}$ (with cost $c_{p,1,wait} = 1$ time unit) and $e_{p,2,wait}$ from $v_{p,2}$ to $v'_{p,2}$ (with cost $c_{p,2,wait} = 1$ time unit), and their return edges, namely $e^{ret}_{p,1}$ and $e^{ret}_{p,2}$ (both with zero cost), are created:

$$\begin{aligned}
e_{p,1,wait} &: \big\langle T_\alpha \wedge (f_\alpha < \alpha) \wedge T_\beta \wedge (f_\beta < \alpha) \wedge (L_p == 0)\big\rangle && \big\{f_\alpha := f_\alpha + c_{p,1,wait};\\
& && \quad f_\beta := f_\beta + c_{p,1,wait}\big\}\\
e_{p,1,obs} &: \big\langle T_\alpha \wedge (f_\alpha \geqslant \alpha) \wedge T_\beta \wedge (L_p == 0)\big\rangle && \big\{L_p := 1\big\}\\
e_{p,2,wait} &: \big\langle \neg T_\alpha \wedge T_\beta \wedge (f_\beta < \beta) \wedge (L_p == 0)\big\rangle && \big\{f_\beta := f_\beta + c_{p,2,wait}\big\}\\
e_{p,2,obs} &: \big\langle \neg T_\alpha \wedge T_\beta \wedge (f_\beta \in [\alpha, \beta]) \wedge (L_p == 0)\big\rangle && \big\{L_p := 1\big\}
\end{aligned}$$

Finally, two new *fault edges*, named $e_{p,1,fault}$ and $e_{p,2,fault}$, from $v_{p,1}$ and $v_{p,2}$ to a new *fault state* called S_{F-I}, respectively, are introduced. The edge conditions and actions of $e_{p,1,fault}$ and $e_{p,2,fault}$ are formulated such that if the input is applied before T_α's expiry (i.e., the lower boundary of $[\alpha, \beta]$) and after T_β's expiry (i.e., the upper boundary of $[\alpha, \beta]$), respectively, the sequence is forced to move into state S_{F-I}. In other words, when input a_i is applied, if the following timing conditions are true, the IUT will be assumed to be in state S_{F-I}, where the test will be declared as failed:

$$\begin{aligned}
e_{p,1,fault} &: \big\langle T_\alpha \wedge (f_\alpha < \alpha) \wedge T_\beta \wedge (f_\beta < \alpha) \wedge (L_p == 0)\big\rangle && \big\{T_\alpha := 0;\ \ f_\alpha := -\infty;\\
& && \quad T_\beta := 0;\ \ f_\beta := -\infty\big\}\\
e_{p,2,fault} &: \big\langle \neg T_\alpha \wedge T_\beta \wedge (f_\beta > \beta) \wedge (L_p == 0)\big\rangle && \big\{T_\beta := 0;\ \ f_\beta := -\infty\big\}
\end{aligned}$$

Therefore, e_i triggers only when a_i is applied after T_α's and before T_β's expiry. But if the input interval condition is not satisfied, G' forces the traversal of either $e_{p,1,fault}$, or $e_{p,2,fault}$, making the tester declare the IUT in the fault state of S_{F-I} (Figure 2).

4.2 n-Clock Interval Fault

Timing conflicts due to n-clock interval faults are concerned with n clocks/timers running concurrently. In a faulty IUT, this fault may result in an altered traversal sequence which can go unnoticed during testing. n-clock interval fault occurs when at least one edge is traversed out of the required testing sequence.

Timing Requirement: Edge $e_i = (v_p, v_q, a_i, o_i, \langle \mathbf{t}_j \rangle, \{\mathbf{t}_j\})$, can be only traversed after a sequence of transitions $\rho = h_1, h_k, h_{k+1} \cdots h_n$, such that h_k was executed before h_{k+1} ($\forall k \in [2, n] \subset \mathbf{Z}^+$).

Timing Fault III: The required order of edges is not respected and the relation between them does not hold true (i.e., for at least one edge $\exists k \in [2, n]$, h_{k+1} was executed before h_k). As a result, for a test sequence, the final state $v'_q \neq v_q$ is verified and the final output $o'_i \neq o_i$ is observed.

The graph augmentation for this case has been skipped due to space constraints, but an extensive study can be found in Refs. [3, 14].

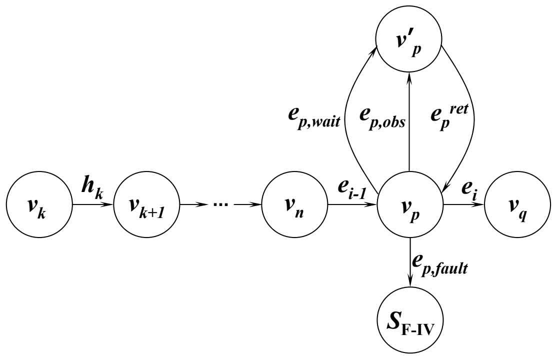

Fig. 3. Graph augmentation of state v_p for detecting Timing Fault IV (a similar augmentation is also applicable to Timing Fault V)

4.3 Incorrect Timer Setting Faults

Timing conflicts which arise due to faulty timer length settings in an IUT are called incorrect timer setting faults where the timer length is incorrectly set either too short or too long (i.e., the timer expires too early or too late).

Timing Requirement: In a test sequence, edge h_k starts timer T_j and is traversed before e_i. Timeout transition $e_i = (v_p,\ v_q, a_i, o_i,\ \langle \mathbf{t}_j \rangle,\ \{\mathbf{t}_j\})$ triggers exactly in D_j time units, where D_j is the timer length.

Timing Fault IV: Timeout transition e_i triggers in D'_j time units and output o_i is observed and state v_q is verified in shorter than the expected time (i.e., $D'_j < D_j$).

Timing Fault V: Timeout transition e_i triggers in D'_j time units and output o_i is observed and state v_q is verified in longer than the expected time (i.e., $D'_j > D_j$).

Graph Augmentation to Detect Timing Fault IV: Let us consider timer T_j with length D_j defined by the specification to be started by the actions of edge h_k and to be expired at edge e_i (reachable from h_k). To detect if the length for T_j is set to D'_j which is shorter than D_j, we introduce a special purpose timer T_s where D_s is the correct timer length as defined by the specification. Timer T_s will be started by edge h_k, which also starts T_j. Therefore, after the augmentation, the time-related conditions and actions for h_k are modeled as:

$$h_k : \langle 1 \rangle \qquad\qquad \{T_j := 1;\quad f_j := 0;\quad T_s := 1;\quad f_s := 0\}$$

An observer state v'_p is appended to state v_p via a new *observer edge* $e_{p,obs}$, *wait edge* $e_{p,wait}$ and *return edge* e_p^{ret} (with cost $c_{p,wait} := 1$ time unit and $c_p^{ret} := 0$, respectively).

$$\begin{aligned} e_{p,obs} &: \langle T_s \wedge (f_s \geqslant D_s) \wedge (T_j \text{ timeout}) \wedge (L_p == 0) \rangle \qquad \{L_p := 1\} \\ e_{p,wait} &: \langle T_s \wedge (f_s < D_s) \wedge (\neg T_j \text{ timeout}) \wedge (L_p == 0) \rangle \qquad \{f_s := f_s + 1\} \end{aligned}$$

Finally, a new *fault state* S_{F-IV} is created which is connected to v_p via $e_{p,fault}$. The edge condition of $e_{p,fault}$ is modified such that if timer T_j expires

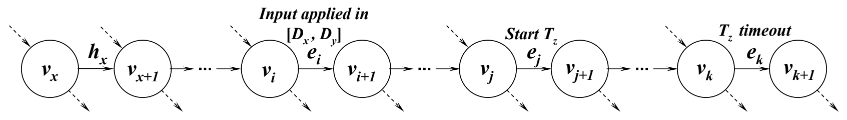

Fig. 4. Generalization of timer specification where Faults I and V hide each other

earlier than expected, the sequence is forced to move to state S_{F-IV} where the tester declares the verdict of the test as failure:

$$e_{p,fault} : \big\langle T_s \wedge (f_s < D_s) \wedge (T_j \text{ timeout}) \wedge (L_p == 0)\big\rangle \quad \big\{T_s := 0; \;\; f_s := -\infty\big\}$$

The edge condition of e_i is also modified such that it traverses only when $f_s \geqslant D_s$ and T_j expires as shown in Figure 3:

$$e_i : \big\langle T_s \wedge (f_s \geqslant D_s) \wedge (T_j \text{ timeout}) \wedge (L_p == 1)\big\rangle \quad \big\{T_s := 0; \;\; f_s := -\infty; \;\; L_p := 0\big\}$$

Graph Augmentation to Detect Timing Fault V: Graph augmentation for Fault V is similar to that of Fault IV (Figure 3), except that the edge conditions are formulated differently:

$$\begin{aligned}
h_k &: \big\langle 1 \big\rangle && \big\{T_j := 1; \;\; f_j := 0; \;\; T_s := 1; \;\; f_s := 0\big\} \\
e_{p,obs} &: \big\langle T_s \wedge (f_s \geqslant D_s) \wedge (T_j \text{ timeout}) \wedge (L_p == 0)\big\rangle && \big\{L_p := 1\big\} \\
e_{p,wait} &: \big\langle T_s \wedge (\neg T_j \text{ timeout}) \wedge (L_p == 0)\big\rangle && \big\{f_s := f_s + 1\big\} \\
e_{p,fault} &: \big\langle T_s \wedge (f_s \geqslant D_s) \wedge (\neg T_j \text{ timeout}) \wedge (L_p == 0)\big\rangle && \big\{T_s := 0; \;\; f_s := -\infty\big\} \\
e_i &: \big\langle T_s \wedge (f_s \geqslant D_s) \wedge (T_j \text{ timeout}) \wedge (L_p == 1)\big\rangle && \big\{T_s := 0; \;\; f_s := -\infty; \;\; L_p := 0\big\}
\end{aligned}$$

5 Multiple Faults

It is possible that, for a given test sequence, a single timing fault, occurring simultaneously with a fault of different type, can exhibit a behavior indistinguishable from an IUT without any faults. We prove in this section that the graph augmentations introduced for single timing faults in Section 4 are capable of detecting such multiple faults. Due to space constraints, only the pairwise combinations of Timing Faults I, IV and V are presented in detail. The other combinations with Timing Fault III are available in [3, 14].

5.1 Multiple Faults of I and V

It is possible that a single Fault I and a single Fault V can hide each other such that the observable behavior of a faulty system is not distinguishable from a non-faulty system.

Lemma 1: Graph augmentation for Fault I (Section 4.1) and Fault V (Section 4.3) can detect simultaneous presence of a single Fault I and a single Fault V in an IUT, irrespective of the order they occur in an edge sequence.

Proof: It is possible to construct an edge sequence such that an input applied too early violating a timing interval requirement of a specification (i.e., Fault I) followed by a timer expiring too late (i.e., Fault V) can generate an output as if the IUT is non-faulty. For the general case, consider a test sequence containing $\cdots, h_x, \cdots, e_i, \cdots, e_j, \cdots, e_k, \cdots$ (Figure 4) where:

- Edge e_i has a timing interval requirement that input a_i be applied within the interval of $[\alpha, \beta]$ (i.e. $\delta \in [\alpha, \beta]$, where δ is the instant at which input is applied, measured from edge h_x).
- Edge e_j from state v_j to state v_{j+1} starts timer T_z with length D_z. e_j : $\langle \neg T_z \rangle \{T_z := 1; \quad f_z = 0\}$
- T_z timeout triggers edge e_k which generates an observable output o_k in $\delta + c_i + c_{(i+1 \longrightarrow j+1)} + D_z + c_k$ time units from h_x, where $c_{(i+1 \longrightarrow j+1)}$ is the total cost of all edges used in the sequence between states v_{i+1} and v_{j+1}.

If input a_i is applied too early (i.e., Fault I where $\delta' < \alpha$) and, at the same time, D_z is incorrectly implemented as too long (i.e., Fault V where $D'_z > D_z$) such that $\delta - \delta' \equiv D'_z - D_z$, the time at which the output o_k is generated remains the same for both the faulty and non-faulty IUTs. The output o_k is generated in $\delta + c_i + c_{(i+1 \longrightarrow j+1)} + D_z + c_k$ time units for non-faulty IUT and in $\delta' + c_i + c_{(i+1 \longrightarrow j+1)} + D'_z + c_k$ time units for faulty IUT after h_x. Since $\delta - \delta' \equiv D'_z - D_z$, Faults I and V can hide each other.

To detect the simultaneous existence of a single Fault I and a single Fault V, the original graph (Figure 4) is augmented (Figure 5) to include new wait and fault states with their associated edges (as in Sections 4.1 and 4.3, respectively). For the above generalized sequence, our graph augmentation introduces special timers T_x and T_y in the test harness with lengths D_x and D_y, respectively, to test the requirement of applying input a_i in the interval $[\alpha, \beta]$, where $\alpha = D_x$ and $\beta = D_y$. Augmentation in Section 4.1 state that both special timers to be started at edge h_x:

$$h_x : \langle \neg T_x \wedge \neg T_y \rangle \qquad \{T_x := 1; \quad f_x := 0; \quad T_y := 1; \quad f_y := 0\}$$

Edge e_i triggers after applying input a_i within time interval $\delta \in [D_x, D_y]$ and stops T_y in its actions:

$$e_i : \langle \neg T_x \wedge T_y \wedge (f_y \in [D_x, D_y]) \wedge (L_p == 1) \rangle \quad \{T_y := 0; \quad f_y := -\infty; \quad L_p := 0\}$$

Similarly, a special purpose timer T_s at the test harness with length D_s is introduced to define the correct timer length for T_z. Therefore, edge e_j starts both T_z and T_s:

$$e_j : \langle \neg T_z \wedge \neg T_s \rangle \qquad \{T_z := 1; \quad f_z := 0; \quad T_s := 1; \quad f_s := 0\}$$

For Fault I augmentation, v_i (starting state of e_i) is replaced by two new states, $v_{i,1}$ and $v_{i,2}$, connected via $e_{i,1,2}$. S_{F-I} and its incoming edges ($e_{i,1,fault}$ and $e_{i,2,fault}$) are created for states $v_{i,1}$ and $v_{i,2}$, respectively. Then, an observer state $v'_{i,1}$ with its associated edges $e_{i,1,wait}$, $e_{i,1,obs}$ and $e^{ret}_{i,1}$ to the $v_{i,1}$ are introduced. Similarly, $v'_{i,2}$, $e_{i,2,wait}$, $e_{i,2,obs}$ and $e^{ret}_{i,2}$ are created for T_β (Section 4.1).

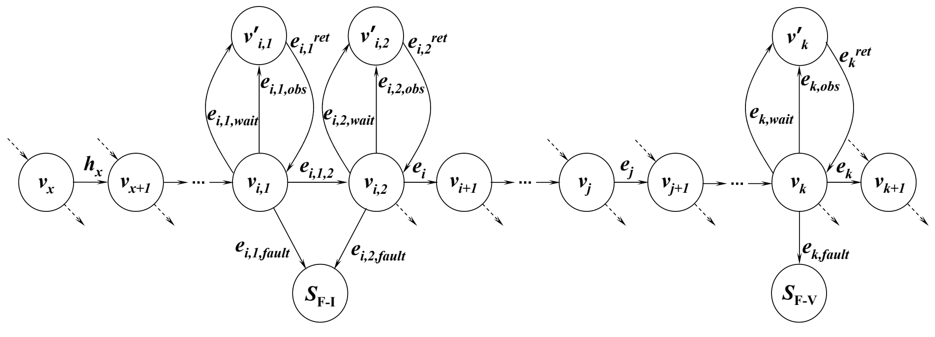

Fig. 5. Graph augmentation for a single occurrence of Faults I and V

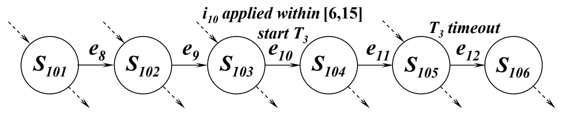

Fig. 6. Timed FSM: T_3 is started by applying i_{10} within time interval $[6, 15]$

For Fault V, state v'_k with edges $e_{k,wait}$, $e_{k,obs}$ and e_k^{ret} are attached to v_k whose outgoing edge is the T_j timeout edge e_k. also state S_{F-V} and edge $e_{k,fault}$ are added to the graph:

$$\begin{aligned}
e_{k,obs} &: \big\langle T_s \wedge (f_s \geqslant D_s) \wedge (T_z \text{ timeout}) \wedge (L_p == 0)\big\rangle & &\big\{L_p := 1\big\} \\
e_{k,wait} &: \big\langle T_s \wedge (\neg T_z \text{ timeout}) \wedge (L_p == 0)\big\rangle & &\big\{f_s := f_s + 1\big\} \\
e_{k,fault} &: \big\langle T_s \wedge (f_s \geqslant D_s) \wedge (\neg T_z \text{ timeout}) \wedge (L_p == 0)\big\rangle & &\big\{T_z := 0;\quad f_z := -\infty; \\
& & &\quad T_s := 0;\quad f_s := -\infty\big\} \\
e_k &: \big\langle T_s \wedge (f_s \geqslant D_s) \wedge (T_z \text{ timeout}) \wedge (L_p == 1)\big\rangle & &\big\{T_z := 0;\quad f_z := -\infty; \\
& & &\quad T_s := 0;\quad f_s := -\infty; \\
& & &\quad L_p := 0\big\}
\end{aligned}$$

After augmentation for both Faults I and V, a correct edge traversal sequence for a non-faulty IUT can be given as: $\cdots$, h_x, $\cdots$, $e_{i,1,wait}$, $e_{i,1}^{ret}$, $e_{i,1,obs}$, $e_{i,1}^{ret}$, $e_{i,1,2}$, $e_{i,2,wait}$, $e_{i,2}^{ret}$, $e_{i,2,obs}$, $e_{i,2}^{ret}$, e_i, $\cdots$, e_j, $\cdots$, $e_{k,wait}$, e_k^{ret}, $e_{k,obs}$, e_k^{ret}, e_k, $\cdots$. A faulty IUT with Faults I and V, where Fault I is traversed before Fault V, will not follow this traversal, but, instead, will end up at state S_{F-I}. Similarly, it can be shown that a sequence where a single Fault V is traversed before a single Fault I will end up at state S_{F-V}. Therefore, a single Fault I and a single Fault V, irrespective of the order of their occurrences, can be detected by our augmentations as indicated in Sections 4.1 and 4.3. □

An example test sequence of containing $\cdots$, e_8, e_9, e_{10}, e_{11}, e_{12}, $\cdots$ is given for the FSM of Figure 6. Suppose the FSM specification defines that, for e_{10}, the input i_{10} should be applied within time interval of $[6, 15]$ seconds (measured from e_8) and that e_{10} starts T_3 with length $D_3 = 4$ seconds. Edge e_{12} is a timeout transition for T_3, and for edges e_9, e_{10}, e_{11}, and e_{12} the costs are $c_9 = 4$, $c_{10} = 1$, $c_{11} = 4$, and $c_{12} = 2$ seconds, respectively. In a correct implementation, i_{10} is

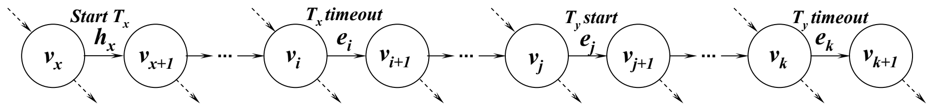

Fig. 7. Generalization of timer specification where Faults IV and V hide each other

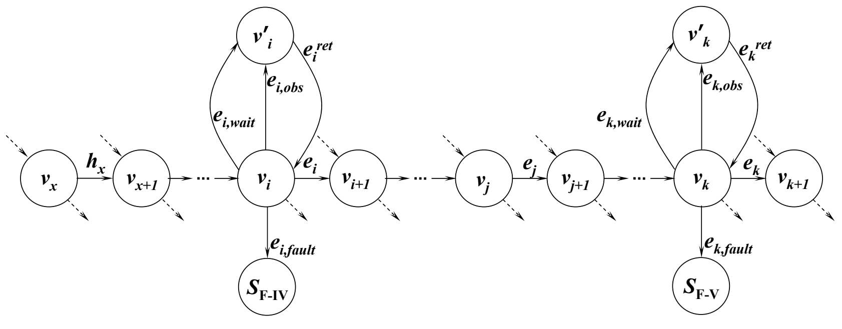

Fig. 8. Graph augmentation for a single occurrence of Faults IV and V

applied 6 seconds after e_8 and timer T_3 expires in 4 seconds (i.e. D_3 = c_{11}= 4 seconds). Hence, the output o_{12} generated by e_{12} is observed in 13 seconds after e_8 traversal (i.e., $\delta + c_{10} + D_3 + c_{12}$= 6 + 1 + 4 + 2 seconds). Now suppose input i_{10} is applied too early at 5 seconds after e_8, and T_3 is incorrectly implemented too long as D'_3 = 5 seconds. In this scenario, output o_{12} is also observed in 13 seconds (i.e., $\delta' + c_{10} + D'_3 + c_{12}$ = 5 + 1 + 5 + 2 seconds). Therefore, without the augmentations, this single occurrences of Faults I and V cannot be detected. However, in the augmented graph G', the sequence will detect single Fault I by forcing the traversal to state S_{F-I} as proven by Lemma 1.

Corollary 1: The multiple occurrences of Faults I and V, irrespective of their occurrence order, are detectable after the graph is augmented for single Faults I and V as in Sections 4.1 and 4.3.

5.2 Multiple Faults of I and IV

Lemma 2: Graph augmentation for Fault I (Section 4.1) and Fault IV (Section 4.3) can detect simultaneous presence of a single Fault I and a single Fault IV, irrespective of the order they occur in an edge sequence (proof analogous to that for Lemma 1).

Corollary 2: Multiple occurrences of Faults I and V, irrespective of their occurrence order, are detectable after the graph is augmented for single Faults I and IV as in Sections 4.1 and 4.3.

5.3 Multiple Faults of IV and V

Lemma 3: Graph augmentations for Fault IV and Fault V (Section 4.3) can detect simultaneous presence of a single Fault IV and a single Fault V, irrespective of the order they occurren an edge sequence.

Proof: Let us first prove that timing faults can hide each other such that the observable behavior for an IUT with Faults IV and V, and a non-faulty IUT are identical. Consider an edge sequence over which two timers, namely T_x and

T_y, are started and expired. For the general case, such a sequence can be defined as $\cdots, h_x, \cdots, e_i, \cdots, e_j, \cdots, e_k, \cdots$ (Figure 7) where:

- Edge h_x from state v_x to v_{x+1} starts timer T_x with length D_x. $h_x : \langle 1 \rangle \{ T_x := 1; f_x := 0 \}$
- Expiry of T_x triggers edge e_i, for which no observable output is generated. $e_i : \langle T_x \wedge (f_x \geqslant D_x) \rangle \{ T_x := 0; \quad f_x := -\infty \}$
- Reachable from e_i, an edge e_j, from state v_j to v_{j+1}, starts timer T_y with length D_y. $e_j : \langle \neg T_x \rangle \{ T_y := 1; \quad f_y := 0 \}$
- Expiry of T_y triggers edge e_k such that output o_k is observed in $(D_x + c_{(i \longrightarrow j+1)} + D_y + c_k)$ time units after h_x is traversed, where $c_{(i \longrightarrow j+1)}$ is the cost of all the edges between states v_i and v_{j+1}. $e_k : \langle T_y \wedge (f_y \geqslant D_y) \rangle \{ T_y := 0; \quad f_y := -\infty \}$
- The inputs for the edges between e_i and e_k do not have input interval requirements (i.e., input timing requirements pertaining to Fault I, which would have been detected by Corollary 2).

Let us suppose now that T_x is implemented too short (i.e., Fault IV with $D'_x < D_x$) and T_y is implemented too long (i.e., Fault V with $D'_y > D_y$) such that $D_x - D'_x \equiv D'_y - D_y$. For a non-faulty IUT, the output o_k will be generated in $(D_x + c_{(i \longrightarrow j+1)} + D_y + c_k)$ time units after the traversal of h_x. For an IUT with Faults IV and V, it will take $(D'_x + c_{(i \longrightarrow j+1)} + D'_y + c_k)$ time units to generate the output o_k. Therefore, since $D_x - D'_x \equiv D'_y - D_y$, it is possible that Timing Faults IV and V can hide each other.

Applying the graph augmentation methods described in Section 4.3, the generalized case of Figure 7 can be modified to include the new wait and fault states with their associated edges. As shown in Figure 8, our graph augmentation introduces special purpose timers T_{sx} and T_{sy} with lengths D_{sx} and D_{sy}, respectively, to define the correct timer lengths for timer T_x and T_y, where $D_{sx} \equiv D_x$ and $D_{sy} \equiv D_y$ time units. In the augmented graph, h_x starts both T_x and T_{sx}, and e_j starts both T_y and T_{sy}:

$$\begin{aligned} h_x &: \langle \neg T_x \wedge \neg T_{sx} \rangle && \{ T_x := 1; \quad f_x := 0; \quad T_{sx} := 1; \quad f_{sx} := 0 \} \\ e_j &: \langle \neg T_y \wedge \neg T_{sy} \rangle && \{ T_y := 1; \quad f_y := 0; \quad T_{sy} := 1; \quad f_{sy} := 0 \} \end{aligned}$$

For Fault IV augmentation, a wait state v'_i with its associated edges $e_{i,wait}$, $e_{i,obs}$ and e_i^{ret} are attached to v_i whose outgoing edge is the timeout edge e_i. A new state S_{F-IV} and its edge $e_{i,fault}$ is added to state v_i:

$$\begin{aligned} e_{i,obs} &: \langle T_{sx} \wedge (f_{sx} \geqslant D_{sx}) \wedge (T_x \text{ timeout }) \wedge (L_p == 0) \rangle && \{ L_p := 1 \} \\ e_{i,wait} &: \langle T_{sx} \wedge (f_{sx} < D_{sx}) \wedge (\neg T_x \text{ timeout}) \wedge (L_p == 0) \rangle && \{ f_{sx} := f_{sx} + 1 \} \\ e_{i,fault} &: \langle T_{sx} \wedge (f_{sx} < D_{sx}) \wedge (T_x \text{ timeout }) \wedge (L_p == 0) \rangle && \{ T_{sx} := 0; \\ & && \quad f_{sx} := -\infty \} \\ e_i &: \langle T_{sx} \wedge (f_{sx} \geqslant D_{sx}) \wedge (T_x \text{ timeout }) \wedge (L_p == 1) \rangle && \{ T_{sx} := 0; \\ & && f_{sx} := -\infty; L_p := 0 \} \end{aligned}$$

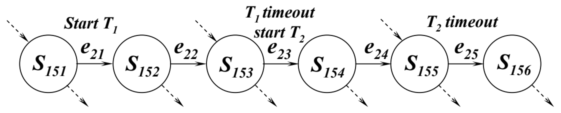

Fig. 9. Timed FSM: T_2 is started by T_1 expiry

Similarly, for Fault V augmentation, an observer state v'_k with its associated edges $e_{k,wait}$, $e_{k,obs}$ and e_k^{ret} are attached to v_k whose outgoing edge is the timeout edge e_k. A new state S_{F-V} and its associated edge $e_{k,fault}$ is added to state v_k:

$$\begin{aligned}
e_{k,obs} &: \big\langle T_{sy} \wedge (f_{sy} \geqslant D_{sy}) \wedge (T_y \text{ timeout }) \wedge (L_p == 0) \big\rangle && \big\{L_p := 1\big\} \\
e_{k,wait} &: \big\langle T_{sy} \wedge (\neg T_y \text{ timeout}) \wedge (L_p == 0) \big\rangle && \big\{f_{sy} := f_{sy} + 1\big\} \\
e_{k,fault} &: \big\langle T_{sy} \wedge (f_{sy} \geqslant D_{sy}) \wedge (\neg T_y \text{ timeout }) \wedge (L_p == 0) \big\rangle && \big\{T_{sy} := 0; \\
& && \quad f_{sy} := -\infty\big\} \\
e_k &: \big\langle T_{sy} \wedge (f_{sy} \geqslant D_{sy}) \wedge (T_y \text{ timeout }) \wedge (L_p == 1) \big\rangle && \big\{T_{sy} := 0; \\
& && f_{sy} := -\infty; L_p := 0\big\}
\end{aligned}$$

After these augmentations, a test sequence for a non-faulty IUT is h_x, $\cdots$, $e_{i,wait}$, e_i^{ret}, $e_{i,obs}$, e_i^{ret}, e_i, $\cdots$, e_j, $\cdots$, $e_{k,wait}$, e_k^{ret}, $e_{k,obs}$, e_k^{ret}, e_k. For a faulty IUT where Fault IV is reached before Fault V, the test sequence will end up in state S_{F-IV} (i.e., the edge $e_{i,fault}$ will be traversed instead of e_i), and hence will detect Fault IV.

Similarly, it can be shown that a test sequence can be constructed such that, if a single Fault V is traversed before a single Fault IV, the test sequence will be forced to state S_{F-V}. Therefore, a single Fault IV and a single Fault V, irrespective of the order of their occurrence, can be detected by augmentations given in Section 4.3. □

Let us illustrate the simultaneous occurrence of Faults IV and V with an example. In Figure 9, the FSM specification defines that edges e_{21} and e_{23} start timers T_1 (expires in e_{23} with $D_1 = 5$ seconds) and T_2 (expires in $e25$ with $D_2 = 4$ seconds), respectively. The costs for the edges e_{22}, e_{23}, e_{24} and e_{25} are given as $c_{22} = 5$, $c_{23} = 2$, $c_{24} = 4$ and $c_{25} = 3$ seconds, respectively.

The test sequence for a non-faulty IUT can be constructed as e_{21}, e_{22}, e_{23}, e_{24}, e_{25} such that timer T_1 expires in 5 seconds and T_2 in 4 seconds. Therefore, using this test sequence, a non-faulty IUT will generate o_{25} by e_{25} 14 seconds after e_{21} traversal (i.e., $D_1 + c_{23} + D_2 + c_{25} = 5 + 2 + 4 + 3$ seconds). Now suppose T_1 is incorrectly implemented as $D'_1 = 4$ seconds and T_2 as $D'_2 = 5$ seconds. This faulty IUT would also generate o_{25} in 14 seconds after e_{21} is traversed (i.e., $D'_1 + c_{23} + D'_2 + c_{25} = 4 + 2 + 5 + 3$ seconds). This example illustrates that, without our augmentations, simultaneous occurrence of single Faults IV and V may be indistinguishable from the non-faulty IUT for certain test cases. However, after graph augmentations, the sequence will detect single occurrences of Fault IV and V by forcing the faulty IUT into state S_{F-IV}.

Corollary 3: The multiple occurrences of Faults IV and V, irrespective of their occurrence order, are detectable after the graph is augmented for single Faults IV and V as in Section 4.3.

6 Concluding Remarks

A number of individually detectable timing faults can hide each other's faulty behavior, making the faulty system indistinguishable from a non-faulty one. A set of augmentations for the timed FSM model introduced in Ref. [7] is presented for single timing faults. The augmentations for the single faults are shown to be capable of detecting multiple occurrences of pairwise combinations of these timing faults. Fault detection capabilities of existing timed automata models and the model studied in this paper will be compared as an extension of this work.

References

[1] J. Alilovic-Curgus and S.T. Vuong. A metric-based theory of test selection and coverage. In *Proc. IFIP Protocol Specif. Test. Verif. (PSTV)*, Liege, Belgium, 1993.

[2] R. Alur and D.L. Dill. A theory of timed automata. *[Elsevier] Theoret. Comput. Sci.* **126**, pp. 183–235, 1994.

[3] S.S. Batth. *Fault Models for Timed EFSMs.* MS thesis, The City College of CUNY, New York, NY, 2004.

[4] W.Y.L. Chan and S.T. Vuong. The UIOv—method for protocol test sequence generation. In *Proc. IFIP Int'l Wksp Protocol Test Syst. (IWPTS)*, Berlin, Germany, 1989.

[5] A. En-Nouaary, R. Dssouli, and F. Khendek. Timed Wp-method: Testing real-time systems. *IEEE Trans. Softw. Eng.* **28**(11), pp. 1023–1038, 2002.

[6] A. En-Nouaary, R. Dssouli, F. Khendek, and A. Elqortobi. Timed test cases generation based on state characterisation technique. In *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, pp. 220–229, Madrid, Spain, 1998.

[7] M.A. Fecko, M.U. Uyar, A.Y. Duale, and P.D. Amer. A technique to generate feasible tests for communications systems with multiple timers. *IEEE/ACM Trans. Netw.* **11**(5), pp. 796–809, 2003.

[8] G. Luo, G.v. Bochmann, and A.F. Petrenko. Test selection based on communicating nondeterministic finite state machines using a generalized Wp-method. *IEEE Trans. Softw. Eng.* **20**(2), pp. 149–162, 1994.

[9] A.F. Petrenko, G.v. Bochmann, and M.Y. Yao. On fault coverage of tests for finite state specifications. *[Elsevier] Comput. Netw. ISDN Syst.* **29**(1), pp. 81–106, 1996.

[10] A. Rezaki and H. Ural. Construction of checking sequences based on characterization sets. *[Elsevier] Comput. Commun.* **18**(12), pp. 911–920, 1995.

[11] D.P. Sidhu and T.K. Leung. Fault coverage of protocol test methods. In *Proc. IEEE INFOCOM*, pp. 80–85, New Orleans, LA, 1988.

[12] J. Springintveld, F. Vaandrager, and P.R. D'Argenio. Testing timed automata. *[Elsevier] Theoret. Comput. Sci.* **254**(1-2), pp. 225–257, 2001.

[13] H. Ural and K. Zhu. Optimal length test sequence generation using distinguishing sequences. *IEEE/ACM Trans. Netw.* **1**(3), pp. 358–371, 1993.

[14] Y. Wang. *Timing Faults in EFSM Models with Multiple Concurrent Timers.* PhD thesis, Graduate Center of CUNY, New York, NY. (in progress).

[15] J. Zhu and S.T. Chanson. Toward evaluating fault coverage of protocol test sequences. In *Proc. IFIP Protocol Specif. Test. Verif. (PSTV)*, pp. 137–151, Vancouver, Canada, 1994.