Maintenance of Smart Buildings using Fault Trees

NATHALIE CAUCHI, Department of Computer Science, University of Oxford

KHAZA ANUARUL HOQUE, Department of Computer Science, University of Oxford, Oxford, UK; Department of Electrical Engineering & Computer Science, University of Missouri, Columbia, USA

MARIELLE STOELINGA, Formal Methods and Tools Group, University of Twente, The Netherlands; Department of Software Science, Radboud University, The Netherlands

ALESSANDRO ABATE, Department of Computer Science, University of Oxford

Timely maintenance is an important means of increasing system dependability and life span. Fault Maintenance trees (FMTs) are an innovative framework incorporating both maintenance strategies and degradation models and serve as a good planning platform for balancing total costs (operational and maintenance) with dependability of a system. In this work, we apply the FMT formalism to a Smart Building application and propose a framework that efficiently encodes the FMT into Continuous Time Markov Chains. This allows us to obtain system dependability metrics such as system reliability and mean time to failure, as well as costs of maintenance and failures over time, for different maintenance policies. We illustrate the pertinence of our approach by evaluating various dependability metrics and maintenance strategies of a Heating, Ventilation, and Air-Conditioning system.[1]

CCS Concepts: • **Computer systems organization** → *Maintainability and maintenance*;

Additional Key Words and Phrases: Fault maintenance trees, formal modelling, probabilistic model checking, reliability, building automation systems, PRISM

28

[1]Parts of this article have been published in the 4th ACM International Conference on Systems of Energy-Efficient Build Environments (BuildSys 2017) [6].

1 INTRODUCTION

The Internet-of-things has enabled a new type of building, termed *Smart Buildings*, which aim to deliver useful building services that are cost effective, reliable, ubiquitous, and ensure occupant comfort and productivity (thermal quality, air comfort). Smart buildings are equipped with many sensors such that a high level of intelligence is achieved: light and heating can be switched on automatically; fire and burglar alarms can be more sophisticated; and cleaning services can be connected to the occupancy rate. Maintenance is a key element to keep smart buildings smart: without proper maintenance (cleaning, replacements, etc.), the benefits of achieving greater efficiency, comfort, increased building lifespan, reliability, and sustainability are quickly lost.

In this article, we consider an important element in smart buildings, namely, the heating, ventilation, and air-conditioning (HVAC) system, responsible for maintaining thermal comfort and ensuring good air-quality in buildings. One way of improving the lifespan and reliability of such systems is by employing methods to detect faults and to perform preventive and predictive maintenance actions. Techniques for fault detection and diagnosis for Smart Building applications have been developed in References [4, 25]. Predictive and preventive maintenance strategies are devised in References [3, 7, 21]. Moreover, a reliability-centered predictive maintenance policy is proposed in Reference [28]. This policy is for a continuously monitored system, which is subject to degradation due to imperfect maintenance. However, these techniques neglect reliability measurements and focus only on synthesis of maintenance policies in the presence of degradation and faults. The current industrial standard for measuring a system's reliability is the use of Fault trees, where the focus is on finding the root causes of a system failure using a top-down approach and do not incorporate degradation of system components and maintenance action [1, 20, 23]. Reference [22] presents the Fault Maintenance Tree (FMT) as a framework that allows us to perform planning strategies for balancing total costs and reliability and availability of the system. FMTs are an extension of FT encompassing both degradation and maintenance models. The degradation models represent the different levels of component degradation and are known as Extended Basic Events (EBE). The maintenance models incorporate the undertaken maintenance policy, which includes both inspections and repairs. These are modelled using Repair and Inspection modules in the FMT framework.

In literature, analysis of FMTs is performed using Statistical Model checking (SMC) [22], which generates sample executions of a stochastic system according to the distribution defined by the system and computes statistical guarantees based on the executions [19]. In contrast, Probabilistic Model Checking (PMC) provides formal guarantees with higher accuracy when compared with SMC [27], at a cost of being more memory intensive and may result in a state space explosion. PMC is an automatic procedure for establishing if a desired property holds in a probabilistic system model, which encodes the probability of making a transition between states. This allows for making quantitative statements about the system's behaviour, which are expressed as probabilities or expectations [18]. Probabilistic model checking has been successfully applied in different domains, so far including aerospace and avionics [13], optical communication [24], systems biology [9], and robotics [10]. In this article, we tackle the FMT analysis using PMC. Our contributions can be summarised as follows:

(1) We formalise the FMT using Continuous Time Markov Chain (CTMCs) and the dependability metrics of a Heating, Ventilation, and Air-Conditioning (HVAC) system, using the Continuous Stochastic Logic (CSL) formalism, such that they can be computed using the PRISM model checker [17].

(2) To tackle the state space explosion problem, we present an FMT abstraction technique that decomposes a large FMT into an equivalent abstract FMT based on a graph decomposition algorithm. This involves an intermediate step where the large FMT is transformed into
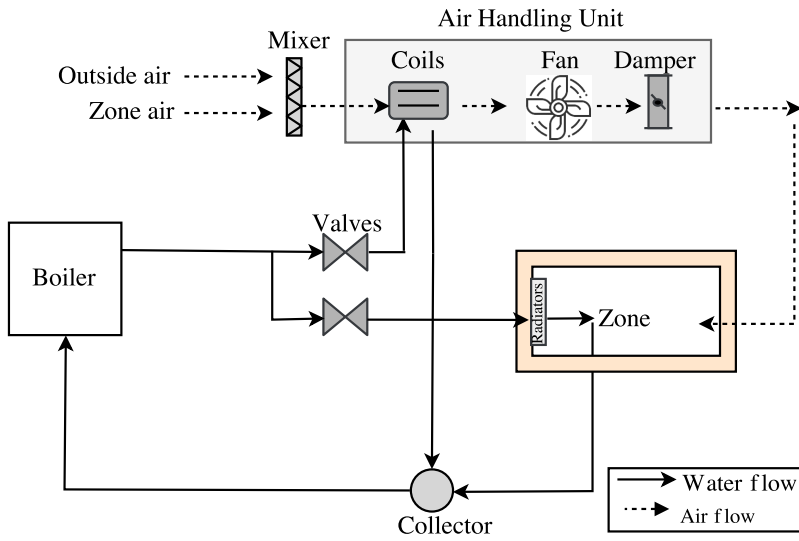
Fig. 1. High level schematic of an HVAC system.

an equivalent direct acyclic graph and decomposed into a set of small sub-graphs. Each of these small sub-graphs are converted to an equivalent smaller CTMC and analysed separately to compute the required metric, while maintaining the original FMT hierarchy. Using our framework, we are able to achieve a 67% reduction in the state space size.

(3) Finally, we construct a FMT that identifies failure of an HVAC, and we illustrate the use of the developed framework to construct and analyse the FMT. We also evaluate relevant performance metrics using the PRISM model checker, compare different maintenance strategies, and highlight the importance of performing maintenance actions.

This article has the following structure: Section 2 introduces the heating, ventilation, and air-conditioning (HVAC) set-up under consideration together with the maintenance question we are addressing. This is followed by Section 3, which presents the fault maintenance trees and probabilistic model checking frameworks. Next, we present the developed methodology for modelling FMT using CTMCs and perform model checking in Section 4. The framework is then applied to the HVAC system in Section 5.

2 PROBLEM FORMULATION

We consider the heating, ventilation, and air-conditioning (HVAC) system setup found within the Department of Computer Science, at the University of Oxford. A graphical description is shown in Figure 1. It is composed of two circuits—the air flow circuitry and the water circuit. The gas boiler heats up the supply water and transfers the supply water into two sections—the supply air heating coils and the radiators. The rate of water flowing in the heating coil is controlled using a heating coil valve, while the rate of water flow in the radiator is controlled using a separate valve. The outside air is mixed with the air extracted from the zone via the mixer. This is fed into the heating coil, which warms up the input air to the desired supply air temperature. This air is supplied back, at a rate controlled by the Air Handling unit (AHU) dampers, into the zone via the supply fan. The radiators are directly connected to the water circuitry and transfer the heat from the water into the zone. The return water, from both the heating coils and the radiators, is then passed through the collector and is returned back to the boiler.
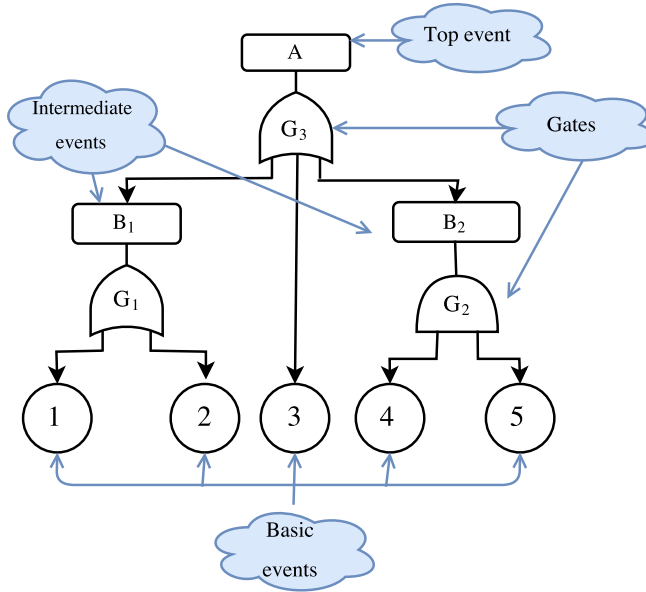
Fig. 2. Example of a FT with five basic events (1–5), two intermediate events (B_1, B_2), and top event A; failures are propagated by the gates (G_1–G_3).

The correct maintenance of this system is essential to ensure that the building operates with optimum efficiency while user comfort is maintained. The choice of the type of maintenance depends on several factors, including the different costs of maintenance and failures and the practical feasibility of performing maintenance. To this end, we aim to address the following maintenance questions: (1) What is the optimal maintenance strategy that minimises system failures? (2) What is the best trade-off between cost of inspections, operation, and maintenance vs. the system's number of expected failures? (3) How frequently should the different maintenance actions such as performing a cleaning or a replacement be performed? (4) What is the effect of employing maintenance over a specific time horizon vs. not performing maintenance?

3 PRELIMINARIES

3.1 Fault Trees

Fault trees (FT) are directed acyclic graphs (DAG) describing the combinations of component failures that lead to system failures. It consists of two types of nodes: events and gates.

Definition 3.1 (Event). An event is an occurrence within the system, typically the failure of a subsystem down to an individual component. Events can be divided into *basic events* (BEs) and intermediate events. BE occur spontaneously and denote the component/system failures while intermediate events are caused by one of or more other events. The event at the top of the tree, called the *top event* (TE), is the event being analysed, modeling the failure of the (sub)system under consideration (both type of events are highlighted in Figure 2).

Definition 3.2 (Gates). The internal nodes of the graph are called *gates* and describe the different ways that failures can interact to cause other components to fail, i.e., how failures in subsystems can combine to cause a system failure. Each gate has one output and one or more inputs. The gates in a FT can be of several types and these include the AND gate, OR gate, k/N-gate [22]. The output of a gate triggers an event that shows how failures propagate through the fault tree.
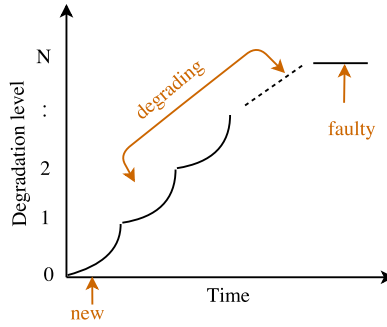
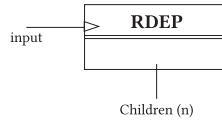Fig. 3. Timing diagram of degradation within an EBE.

Fig. 4. RDEP gate with 1 input and dependent components also known as children.

Figure 2 depicts a fault tree were the basic events are shown using circles, top and intermediate events are depicted by a rectangle.

3.2 Fault Maintenance Trees

Fault maintenance trees (FMT) extend fault trees by including maintenance (all the standard FT gates are also employed by the FMTs). This is achieved by making use of:

(1) *Extended Basic Events (EBE)*—The basic events are modified to incorporate degradation models of the component the EBE represents. The degradation models represent different discrete levels of degradations the components can be in and are a function of time. The timing diagram showing the progression of degradation within an EBE is shown in Figure 3. The presented EBE had N discrete degradation levels, initially the EBE is its new state and it gradually moves from one degradation levels, based on the underlying distribution describing the degradation, to the next until the faulty level N is reached.

(2) *Rate Dependency Events*—A new gate, introduced in Reference [22] and labelled as *RDEP*, accelerates the degradation rates of n dependent child nodes and is depicted in Figure 4. When the component connected to the input of the RDEP fails, the degradation rate of the dependent components is accelerated with an acceleration factor γ. The corresponding timing diagram is shown in Figure 5. When the input signal is enabled (*input* = 1), the child EBE moves to the next degradation levels at a faster rate.

(3) *Repair and Inspection modules*— The repair module (RM) performs cleaning or replacements actions. These actions can be either carried out using fixed time schedules or when enabled by the inspection module (IM). The RM module performs periodic maintenance actions (clean or replace), independently of the IM. The IM performs periodic inspections and when components fall below a certain degradation threshold a maintenance action is initiated by the IM and performed by the RM (outside of the RM's periodic maintenance cycle). The IM and RM modules are depicted in Figure 6. The effect of performing a maintenance cleaning or replacement action on the EBE is illustrated in Figure 7. When a
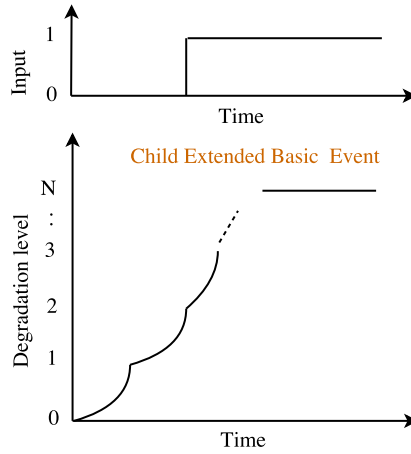
Fig. 5. Degradation level evolution of child EBE showing effect of RDEP on degradation rate. Note, when the input is equal to 1 the curve representing the degradation rate to go from one degradation level to the next (e.g., going from degradation level 2 to 3) is steeper vs. previous degradation level transitions (e.g., going from 0 to 1 or 1 to 2).
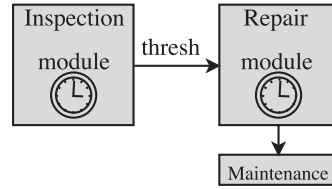
Fig. 6. High-level description of the inspection and repair modules. The repair module performs maintenance actions periodically (clean or replace). The inspection module performs inspections periodically and when the degradation level of an EBE reaches *thresh* level, it triggers the repair module to perform a maintenance action immediately.
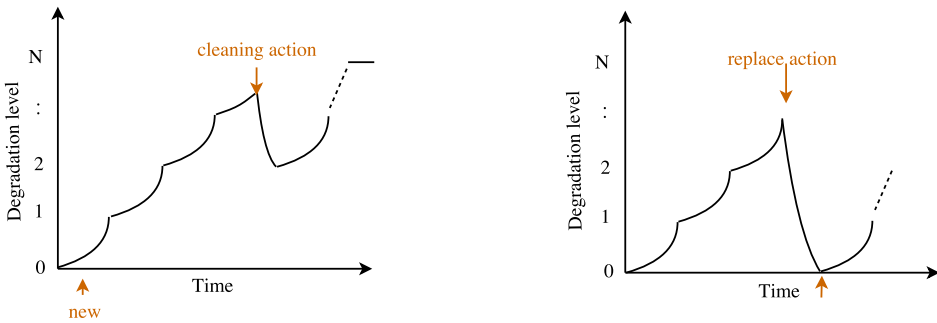
Fig. 7. Degradation level progression of EBE for different maintenance actions.

cleaning action is performed, the EBE moves back to its previous degradation level, while when a replacement is performed, the EBE moves back to the initial level.

A visual rendering of an FMT is given in Figure 8. It is composed of five EBEs located at the bottom of the tree, one RDEP with one dependent child, three gates, one repair and inspection module, and three events that show the different fault stages.
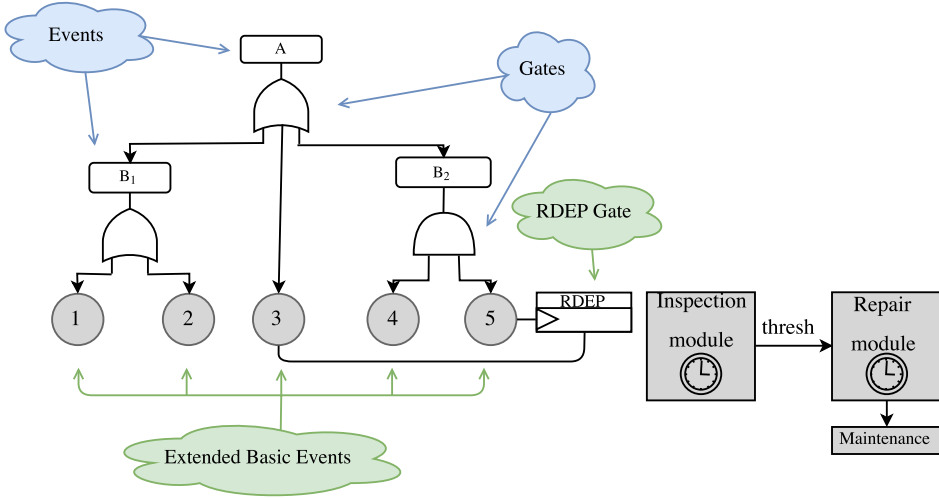
Fig. 8. Example of a fault maintenance tree.

3.3 Probabilistic Model Checking

Model checking [8] is a well-established formal verification technique used to verify the correctness of finite-state systems. Given a formal model of the system to be verified in terms of labelled state transitions and the properties to be verified in terms of temporal logic, the model checking algorithm exhaustively and automatically explores all the possible states in a system to verify if the property is satisfiable or not. *Probabilistic model checking* (PMC) deals with systems that exhibit stochastic behaviour and is based on the construction and analysis of a probabilistic model of the system. We make use of CTMCs, having both transition and state labels, to perform stochastic modelling. Properties are expressed in the form of Continuous Stochastic Logic (CSL) [16], a stochastic variant of the well-known Computational Tree Logic (CTL) [8], which includes reward formulae. Note, a system can be modelled using multiple CTMCs, which represent different sub-components within the whole. Transition labels are then used to synchronise the individual CTMCs representing different parts of a system and in turn obtain the full CTMC representing the whole system.

*Definition 3.3.* The tuple $C = (S, s_0, \text{TL}, \text{AP}, L, \mathbf{R})$ defines a CTMC that is composed of a set of states S, the initial state s_0, a finite set of transition labels TL, a finite set of atomic propositions AP, a labelling function $L : S \to 2^{AP}$ and the transition rate matrix $\mathbf{R} : S \times S \to \mathbf{R}_{\geq 0}$. The rate $\mathbf{R}(s, s')$ defines the delay before which a transition between states s and s' takes place. If $\mathbf{R}(s, s') \neq 0$, then the probability that a transition between the states s and s' is defined as $1 - e^{-\mathbf{R}(s, s')t}$ where t is time. No transitions will trigger if $\mathbf{R}(s, s') = 0$.

The logic of CSL specifies state-based properties for CTMCs, built out of propositional logic (with atoms $a \in \text{AP}$), a steady-state operator (S) that refers to the stationary probabilities, and a probabilistic operator (P) for reasoning about transient state probabilities. The state formulas are interpreted over states of a CTMC, whereas the path formulas are interpreted over paths in a CTMC. The syntax of CSL is

$$\Phi ::= \text{true} \mid a \mid \Phi \wedge \Phi \mid \neg\Phi \mid S_{\sim p}[\Phi] \mid P_{\sim p}[\phi],$$
$$\psi ::= X\,\phi \mid \Phi\,U^{\geq T}\,\Phi,$$

where $\sim\in\{<,\leq,=,\geq,>\}$, $p\in[0,1]$, $T\in\mathbb{R}_{\geq0}$ is the time horizon, X is the next operator, and U is the until operator. The semantics of CSL formulas is given in Reference [16]. $S_{\sim p}[\Phi]$ asserts that the steady-state probability for a Φ-state meets the bound $\sim p$, whereas $P_{\sim p}[\Phi \text{ U}^{\leq t} \Phi]$ asserts that with probability $\sim p$, by the time t a state satisfying Φ will be reached such that all preceding states satisfy Φ. Additional properties can be specified by adding the notion of rewards. The extended CSL logic adds reward operators, a subset of which are [16]

$$\text{R}_{\sim r}[\text{C}^{\leq T}] \mid \text{R}_{\sim r}[\text{F } \Phi],$$

where $r, t\in\mathbb{R}_{\leq0}$ and Φ is a CSL formula. A state s satisfies $\text{R}_{\sim r}[\text{C}^{\leq T}]$ if, from state s, the expected reward cumulated up until T time units have elapsed satisfies $\sim r$ and $\text{R}_{\sim r}[\text{F } \Phi]$ is true if, from state s, the expected reward cumulated before a state satisfying Φ is reached meets the bound $\sim r$.

Examples of a CSL property with its natural language translation are: (i) $P_{\geq0.95}[\text{F } complete]$—"The probability of the system eventually completing its execution successfully is at least 0.95." Each state (and/or transition) of the model is assigned a real-valued reward, allowing queries such as: (ii) $R_{=?}[\text{F } success]$—"What is the expected reward accumulated before the system successfully terminates?" Rewards can be used to specify a wide range of measures of interest, for example, the total operational costs and the total percentage of time during which the system is available.

4 FORMALIZING FMTS USING CTMCS

4.1 FMT Syntax

To formalise the syntax of FMTs using CTMCs, we first define the set $\mathcal{F}$, characterizing each FMT element by type, inputs, and rates. We introduce a new element called DELAY, which will be used to model the deterministic time delays required by the extended basic events (EBE), repair module (RM) and inspection module (IM). We restrict the set $\mathcal{F}$ to contain the EBE, RDEP gate, OR gate, DELAY, RM and IM modules since these will be the components used in the case study presented in Section 5.

Definition 4.1. The set $\mathcal{F} = \{EBE, RDEP, OR, DELAY, RM, IM\}$ of FMT elements consists of the following tuples. Here, $n, N\in\mathbb{N}$ are natural numbers, *thresh*, *in*, *trig* $\in\{0,1\}$ take binary values, $T_{cln}, T_{rplc}, T_{rep}, T_{oh}, T_{insp}\in\mathbb{R}_{\geq0}$ are deterministic delays, $T_{deg}\in\mathbb{R}_{\geq0}$ is a rate and $\gamma\in\mathbb{R}_{\geq0}$ is a factor.

- $(EBE, T_{deg}, T_{cln}, T_{rplc}, N)$ represent the extended basic events with N discrete degradation levels, each of which degrade with a time delay equal to T_{deg}. It also takes as inputs the time taken to restore the EBE to the previous degradation level T_{cln} when cleaning is performed and the time taken to restore the EBE to its initial state T_{rplc} following a replacement action.
- $(RDEP, n, \gamma, in, T_{deg})$ represents the RDEP gate with n dependent children, acceleration factor γ, the input in which activates the gate and T_{deg} the degradation rate of the dependent children.
- (OR, n) represents the OR gate with n inputs. When either one of the inputs reaches the state labelled with $failed$, the OR gate returns a true signal.
- $(RM, n, T_{rep}, T_{oh}, T_{insp}, T_{cln}, T_{rplc}, thresh, trig)$ represents the RM module, which acts on n EBEs (in our case, this corresponds to all the EBEs in the FMT). The RM can either be triggered periodically to perform a cleaning action, every T_{rep} delay, or a replacement action, every T_{oh} delay, or by the IM when the delay T_{insp} has elapsed and the $thresh$ condition is met. The time to perform a cleaning action is T_{cln}, while the time taken to perform a replacement is T_{rplc}. The $trig$ signal ensures that when the component is not in the degraded states, no unnecessary maintenance actions are carried out.
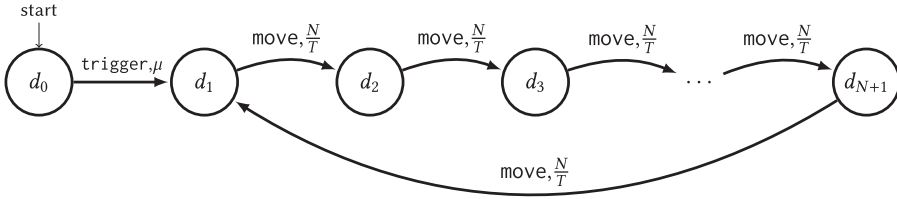
Fig. 9. CTMC representing DELAY with N states used to approximate a delay equal to T approximated using $Erlang(N, \frac{N}{T})$. The transition labels TL = {trigger, move} are shown on each of the transitions. The state labels are not shown and the initial state of the CTMC is pointed to using an arrow labelled with start.

- $(IM, n, T_{insp}, T_{cln}, T_{rplc}, thresh)$ represents the IM module, which acts on n EBEs (in our case, this corresponds to all the EBEs in the FMT). The IM initiates a repair depending on the current state of the EBE. Inspections are performed in a periodic manner, every T_{insp}. If during an inspection the current state of the EBE does not correspond to the *new* or *failed* state (i.e., the degradation level of the inspected EBE is below a certain threshold), then the *thresh* signal is activated and is sent to the RM. Once a cleaning action is performed the IM moves back to the initial state with a delay equal to T_{cln} or T_{rplc} depending on the maintenance action performed.
- $(DELAY, T, N)$ represents the DELAY module, which takes two inputs representing the deterministic delay $T \in \{T_{deg}, T_{cln}, T_{rplc}, T_{rep}, T_{oh}, T_{insp}\}$ to be approximated using an Erlang distribution with N states. This DELAY module can be extended by inclusion of a reset transition label, which when triggered restarts the approximation of the deterministic delay before it has elapsed. The extended DELAY module is referred to as $(DELAY, T, N)_{ext}$.

The FMT is defined as a special type of directed acyclic graph $G = (V, E)$, where the vertices V represent the gates and the events, which represent an occurrence within the system, typically the failure of a subsystem down to an individual component level, and the edges E, which represent the connections between vertices. The vertices V are labelled instances of elements in $\mathcal{F}$, i.e., V may contain multiple elements of the same component obtained from the set $\mathcal{F}$, which are identified by their common element label. Events can either represent the EBEs or *intermediate* events, which are caused by one or more other events. The event at the top of the FMT is the top event (TE) and corresponds to the event being analysed—modelling the failure of the (sub)system under consideration. The EBE are the leaves of the DAG. For G to be a well-formed FMT, we take the following assumptions (i) vertices are composed of the OR, RDEP gates, (ii) there is only one top event, (iii) RDEP can only be triggered by EBEs and (iv) RM and IM are not part of the DAG tree but are modelled separately This DAG formulation allows us to propose a framework in Section 4.5, such that we can efficiently perform probabilistic model checking.

Definition 4.2. A fault maintenance tree is a directed acyclic graph $G = (V, E)$ composed of vertices V and edges E.

4.2 Semantics of FMT Elements

Next, we provide the semantics for each FMT element, which are composed using the syntax of CTMC (cf. Definition 3.3). These elements are then instantiated based on the underlying FMT structure to form the semantics of the whole FMT. We obtain the semantics of the whole FMT via synchronisation of transition labels between the different CTMCs representing the individual FMT elements. This is further explained in Section 4.3.
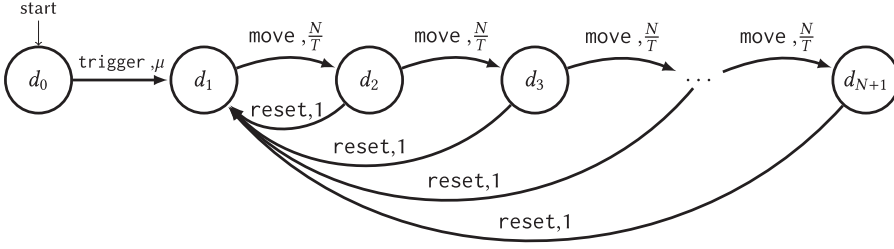
Fig. 10. CTMC representing the extended DELAY with N states used to approximate a delay equal to T. Delay approximated using $Erlang(N, \frac{N}{T})$. The transition labels TL = {trigger, move, reset} are shown on each of the state transitions, while the state labels are not shown.

DELAY. We define the semantics for the $(DELAY, T, N)$ element using Figure 9 and describe the corresponding CTMC using the set of states given by $D = \{d_0, d_1, \ldots, d_{N+1}\}$, the initial state d_0, the set of transitions labels TL = {trigger, move}, the set of atomic propositions AP = $\{T\}$ with $L(d_0) = \cdots = L(d_N) = \emptyset$, and $L(d_{N+1}) = \{T\}$. The rate matrix $\mathbf{R}$ becomes clear from Figure 9 and

$$\mathbf{R}_{ij} = \begin{cases} \mu & i = 0 \wedge j = 1, \\ \frac{N}{T} & ((i \geq 1 \vee i < N+1) \wedge j = i+1) \vee (i = N+1 \wedge j = 1), \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

with i representing the current state, j is the next state and μ is a fixed large value corresponding to introducing a negligible delay, which is used to trigger all the DELAY modules at the same time (cf. Definition 3.3). In Figure 10, we define the semantics of $(DELAY, T, N)_{ext}$. This results in the CTMC described using the state space $D = \{d_0, d_1, \ldots, d_{N+1}\}$, the initial state d_0, the set of transition labels TL = {trigger, move, reset}, the set of atomic propositions AP = $\{T\}$, the labelling function $L(d_0) = L(d_1) = \cdots = L(d_N) = \emptyset$, and $L(d_{N+1}) = \{T\}$ and the rate matrix $\mathbf{R}$, where

$$\mathbf{R}_{ij} = \begin{cases} \mu & i = 0 \wedge j = 1, \\ 1 & (i \geq 2 \vee i < N+1) \wedge j = 1, \\ \frac{N}{T} & ((i \geq 1 \vee i < N+1) \wedge j = i+1) \vee (i = N+1 \wedge j = 1), \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

with i representing the current state and j is the next state. In both instances, the deterministic delays is approximated using an Erlang distribution [12] and all DELAY modules are synchronised to start together using the trigger transition label. The extended DELAY module have the transition labels reset, which restarts the Erlang distribution approximation whenever the guard condition is met at a rate of $1 \times \mathbf{R}_{sync}$ where $\mathbf{R}_{sync}$ is the rate coming from the use of synchronisation with other modules causing the reset to occur (as explained in Section 4.3). This is required when a maintenance action is performed, which restores the EBE's state back to the original state and thus restart the degradation process, before the degradation time has elapsed.

Remark 1. A random variable $Z \in \mathbb{R}_+$ has an Erlang distribution with $k \in \mathbb{N}$ stages and a rate $\lambda \in \mathbb{R}_+, Z \sim Erlang(k, \lambda)$, if $Z = Y_1 + Y_2 + \cdots Y_k$, where each Y_i is exponentially distributed with rate λ. The cumulative density function of the Erlang distribution is characterised using

$$f(t; k, \lambda) = 1 - \sum_{n=0}^{k-1} \frac{1}{n!} \exp(-\lambda t)(\lambda t)^n \quad \text{for } t, \lambda \geq 0, \tag{3}$$

and for $k = 1$, the Erlang distribution simplifies to the exponential distribution. In particular, the sequence $Z_k \sim Erlang(k, \lambda k)$ converges to the deterministic value $\frac{1}{\lambda}$ for large k. Thus, we can approximate a deterministic delay T with a random variable $Z_k \sim Erlang(k, \frac{k}{T})$ [5, 14]. Note, there
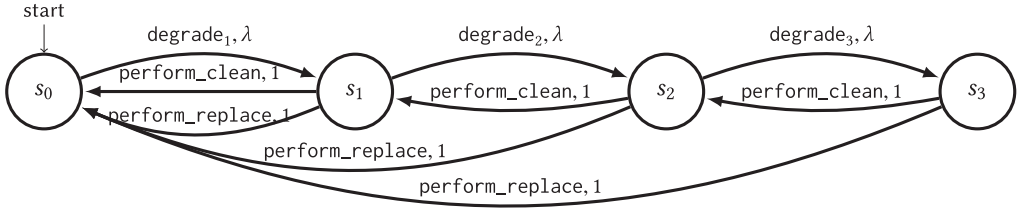
Fig. 11. CTMC representing the EBE with $N = 3$ with the transition labels $\text{TL}_{EBE} = \{\text{degrade}_{i \in \{1,2,3\}},$ perform_clean, perform_replace$\}$ on each of the state transitions. For clarity, the state labels are not shown. The deterministic delays contained represent the transition label that is triggered when the delay generated by the corresponding DELAY module has elapsed. The degradation rate is equal to $\lambda = \frac{N}{MTTF}$ where MTTF is the components mean time to failure.

is a trade-off between the accuracy and the resulting blow-up in size of the CTMC model for larger values of k (a factor of k increase in the model size) [12]. In this work, the Erlang distribution will be used to model the fixed degradation rates, the maintenance and inspection signals. This is a similar approach taken in [22] where degradation phases are approximated by an (k,λ)-Erlang distribution.

Extended Basic Events (EBE). The EBE are the leaves of the FMT and incorporate the component's degradation model. EBE are a function of the total number of degradation steps N considered. Figure 11 shows the semantics of the $(EBE, T_{deg}, T_{cln}, T_{rep}, N = 3)$. The corresponding CTMC is described by the tuple $(\{s_0, s_1, s_2, s_3\}, s_0, \text{TL}_{EBE}, \text{AP}_{EBE}, L_{EBE}, \mathbf{R}_{EBE})$ where s_0 is the initial state,

$$\text{TL}_{EBE} = \{\text{degrade}_{i \in \{0, \dots, N\}}, \text{perform\_clean}, \text{perform\_replace}\},$$

the atomic propositions $\text{AP}_{EBE} = \{new, thresh, failed\}$, the labelling function $L(s_0) = \{new\}$, $L(s_1) = L(s_2) = \{thresh\}, L(s_3) = \{failed\}$ and

$$\mathbf{R}_{EBE} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

The deterministic time delays taken as inputs are modelled using three different DELAY modules:

(1) an extended DELAY module approximating T_{deg} with the transition label move replaced with degrade_N such that synchronisation between the two CTMCs is performed (explained in Section 4.3). When T_{deg} has elapsed the transition labelled with degrade_N is triggered and the EBE moves to the next state at a rate[2] equal to $\frac{N}{T_{deg}} \times 1$. The reset transition label and the corresponding transitions are replicated in extended DELAY module and replaced with perform_clean and perform_replace. When the the previous state (if cleaning action is carried out) or to the initial state (if replace action is performed).

(2) a DELAY module approximating T_{cln} with the transition label move replaced with perform_clean. When T_{cln} has elapsed the transition with transition label perform_clean is triggered and the EBE moves to the previous state at a rate equal to $\frac{N}{T_{cln}}$.

---

[2]This is a direct consequence of synchronisation and corresponds to $\mathbf{R} \times \mathbf{R}_{EBE}$. Refer to Section 4.3.

Fig. 12. CTMC representing the RM with TL_{RM} = {inspect, check_maintenance, perform_maintenance} shown on the state transitions. The guard condition $trig = 0/1$ or $thresh = 0/1$ must be satisfied for the corresponding transition to trigger when it is activated via synchronisation with the transition label.

(3) a DELAY module approximating T_{rplc} with the transition label move replaced with perform_replace. When T_{rplc} has elapsed the transition label perform_replace is triggered and the EBE moves to the initial state at a rate equal to $\frac{N}{T_{rplc}}$.

The transition labels perform_clean and perform_replace cannot be triggered at the same time and it is assumed that $T_{cln} \neq T_{rplc}$. This is a realistic assumption as only one maintenance action is performed at the same time.

RDEP Gate. The RDEP gate has static semantics and is used in combination with the semantics of its n dependent EBEs. When triggered ($input = 1$), the associated EBE reaches the state labelled *failed*, the degradation rate of the n dependent children is accelerated by a factor γ. We model the *input* signal using

$$input = \begin{cases} 1 & L(s) = \text{failed}, \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

where $L(s)$ is the label of the current state of the associated EBE (cf. Figure 5). Similarly, we map the RDEP gate function using

$$RA = \begin{cases} \gamma T_{deg_1}, \ldots, \gamma T_{deg_n} & input = 1, \\ T_{deg_1}, \ldots, T_{deg_n} & \text{otherwise}, \end{cases} \tag{5}$$

where $T_{deg_i}, i \in 1, \ldots n$ corresponds to the degradation rate of the n dependent children.[3]

OR Gate. The OR gate indicates a failure when either of its input nodes have failed and also does not have semantics itself but is used in combination with the semantics of its n dependent input events (EBEs or intermediate events). We use

$$FAIL = \begin{cases} 0 & E_1 = 1 \wedge \cdots \wedge E_n = 1, \\ 1 & \text{otherwise}, \end{cases} \tag{6}$$

where $E_i = 1, i \in 1 \ldots n$ corresponds to when the n events (cf. Definition 3.1), connected to the OR gate, represent a failure in the system. In the case of EBEs, $E_1 = 1$ occurs when the EBE reaches the *failed* state.

Repair Module (RM). Figure 12 shows the semantics of $(RM, n, T_{rep}, T_{oh}, T_{insp}, T_{cln}, T_{rplc}, thresh, trig)$. The CTMC is described using the state space $\{rm_0, rm_1\}$, the initial state rm_0, the

---

[3]Note, this effectively results in changing the deterministic delay being modelled by the DELAY module to a new value if $input = 1$.

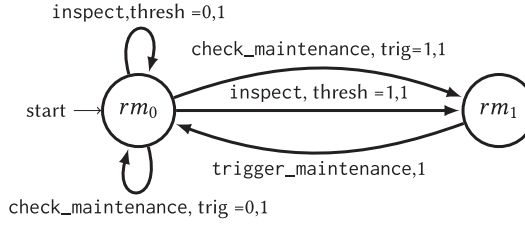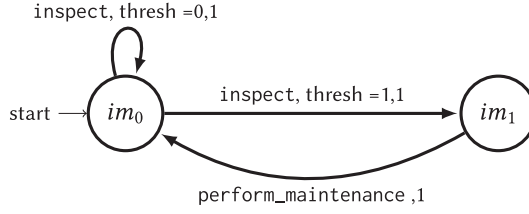Fig. 13. CTMC representing the IM with $\text{TL}_{IM} = \{\texttt{inspect}, \texttt{perform\_maintenance}\}$ shown on the state transitions. The guard condition $trig = 0$ and $thresh = 1$ must be satisfied for the corresponding transition to trigger when it is activated via synchronisation with the transition label.

transition labels

$$\text{TL}_{RM} = \{\texttt{inspect}, \texttt{check\_clean}, \texttt{check\_replace}, \texttt{trigger\_clean}, \texttt{trigger\_replace}\},$$

the atomic propositions $\text{AP} = \{maintenance\}$, the labelling function $L(rm_0) = \{\emptyset\}$, $L(rm_1) = \{maintenance\}$, and with

$$\mathbf{R}_{IM} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

For brevity in Figure 12, we used the transition labels $\texttt{check\_maintenance}$ and $\texttt{trigger\_maintenance}$. The transition label $\texttt{check\_maintenance}$ and corresponding transitions are replicated and the transition labels replaced by $\texttt{check\_clean}$ or $\texttt{check\_replace}$ to allow for both type of maintenance checks. Similarly, the transition label $\texttt{trigger\_maintenance}$ and corresponding transitions are duplicated and the transition labels replaced by $\texttt{trigger\_clean}$ or $\texttt{trigger\_replace}$ to allow the initiation of both type of maintenance actions to be performed. Due to synchronisation, only one of the transitions may trigger at any time instance (as explained in Section 4.3). The transition labels $\texttt{trigger\_clean}$ or $\texttt{trigger\_replace}$ correspond to the transition label $\texttt{trigger}$ within the DELAY module approximating the deterministic delays T_{cln} and T_{rplc}, respectively. The deterministic delays, which trigger $\texttt{inspect}$, $\texttt{check\_clean}$, or $\texttt{check\_replace}$, correspond to when the time delays T_{insp}, T_{rep}, and T_{oh}, respectively, have elapsed. All these signals are generated using individual DELAY modules with the move transition label for each module replaced using $\texttt{inspect}$, $\texttt{check\_clean}$, or $\texttt{check\_replace}$, respectively. The $thresh$ signal is modelled using

$$thresh = \begin{cases} 1 & L(s_{j,1}) = thresh \vee \cdots \vee L(s_{j,n}) = thresh, \\ 0 & \text{otherwise}, \end{cases} \tag{7}$$

where $L(s_{j,i}), j \in 0 \ldots N, i \in 1 \ldots n$ correspond to the label of the current state j of each of the n EBE. Similarly, we model the $trig$ signal using

$$trig = \begin{cases} 1 & L(s_{j,1}) \neq new \vee \cdots \vee L(s_{j,n}) \neq new, \\ 0 & \text{otherwise}. \end{cases} \tag{8}$$

Both signals act as guards which when triggered determine which transition to perform (cf. Figure 12).

Inspection Module (IM). The semantics of the $(IM, n, T_{insp}, T_{cln}, T_{rplc}, thresh)$ is depicted in Figure 13. The CTMC is defined using the tuple $(\{im_0, im_1\}, im_0, \text{TL}_{IM}, \text{AP}_{IM}, L_{IM}, \mathbf{R}_{IM})$. Here,

$$\text{TL}_{IM} = \{\texttt{inspect}, \texttt{perform\_clean}, \texttt{perform\_replace}\},$$

$AP_{IM} = \{\emptyset\}$, with $L(s_0) = L(s_1) = \emptyset$ and

$$\mathbf{R}_{IM} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

The *thresh* signal corresponds to same signal used by the RM, given using Equation (7). In Figure 13, for clarity, we use the transition label perform_maintenance. This transition label and corresponding transitions are duplicated and the transition labels are replaced by either perform_clean or perform_replace to allow for both type of maintenance actions to be performed when one of them is triggered using synchronisation. The same DELAY modules used in the RM and EBE to represent the deterministic delays are used by the IM. The DELAY module used to represent the deterministic delays T_{cln} and T_{rplc} triggers the transition labels perform_clean or perform_replace. This represents that the maintenance action has completed.

4.3 Semantics of Composed FMT

Next, we show how to obtain the semantics of a FMT from the semantics of its elements using the FMT syntax introduced in Section 4.1. We define the DAG G by defining the vertices V and the corresponding events E. The leaves of the DAG are the events corresponding to the EBE. The events E are connected to the vertices V, which trigger the corresponding auxiliary function used to represent the semantics of the gates. The *Events* connected to the RM and IM are initiated by triggering the auxiliary functions *thresh* and *trig* given using Equations (7) and (8), respectively. Based on the structure of G, we compute the corresponding CTMC by applying parallel composition of the individual CTMCs representing the elements of the FMT. The parallel composition formulae are derived from Reference [11] and defined as follows.

Definition 4.3 (Interleaving Synchronization). The interleaving synchronous product of $C_1 = (S_1, s_{01}, TL_1, AP_1, L_1, \mathbf{R}_1)$ and $C_2 = (S_2, s_{02}, TL_2, AP_2, L_2, \mathbf{R}_2)$ is $C_1 || C_2 = (S_1 \times S_2, (s_{01}, s_{02}), TL_1 \cup TL_2, AP_1 \cup AP_2, L_1 \cup L_2, \mathbf{R})$ where $\mathbf{R}$ is given by

$$\frac{s_1 \xrightarrow{\alpha_1, \lambda_1} s_1'}{(s_1, s_2) \xrightarrow{\alpha_1, \lambda_1} (s_1', s_2)}, \text{ and } \frac{s_2 \xrightarrow{\alpha_2, \lambda_2} s_2'}{(s_1, s_2) \xrightarrow{\alpha_2, \lambda_2} (s_1, s_2')},$$

and $s_1, s_1' \in S_1, \alpha_1 \in TL_1, \mathbf{R}_1(s_1, s_1') = \lambda_1, s_2, s_2' \in S_2, \alpha_2 \in TL_2, \mathbf{R}_2(s_2, s_2') = \lambda_2$.

Definition 4.4 (Full Synchronization). The full synchronous product of $C_1 = (S_1, s_{01}, TL_1, AP_1, L_1, \mathbf{R}_1)$ and $C_2 = (S_2, s_{02}, TL_2, AP_2, L_2, \mathbf{R}_2)$ is $C_1 || C_2 = (S_1 \times S_2, (s_{01}, s_{02}), TL_1 \cup TL_2, AP_1 \cup AP_2, L_1 \cup L_2, \mathbf{R})$ where $\mathbf{R}$ is given by

$$\frac{s_1 \xrightarrow{\alpha, \lambda_1} s_1' \text{ and } s_2 \xrightarrow{\alpha, \lambda_2} s_2'}{(s_1, s_2) \xrightarrow{\alpha, \lambda_1 \times \lambda_2} (s_1', s_2')},$$

and $s_1, s_1' \in S_1, \alpha \in TL_1 \wedge TL_2, \mathbf{R}_1(s_1, s_1') = \lambda_1, s_2, s_2' \in S_2, \alpha_2 \in TL_2, \mathbf{R}_2(s_2, s_2') = \lambda_2$.

For any pair of states, synchronisation is performed either using interleaving or full synchronisation. For full synchronisation, as in Definition 4.3, the rate of a synchronous transition is defined as the product of the rates for each transition. The intended rate is specified in one transition and the rate of other transition(s) is specified as one. For instance, the RM synchronises using full synchronisation with the DELAY modules representing T_{insp}, T_{rep} and T_{rplc} and therefore, to perform synchronisation between the RM and the DELAY modules, the rates of all the transitions of RM should have a value of one (cf. Figure 12), while the rate of the DELAY modules represent the actual rates (cf. Figures 9 and 10). The same principle holds for the EBEs and

Table 1. Performing Synchronisation Between the Different FMT Components
and the Synchronisation Method Used

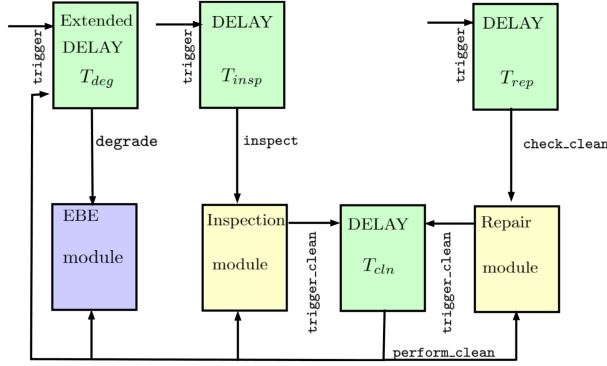| Component	Synchronised with component	Transition label	Synchronisation method
DELAY representing T_{deg}	DELAY modules representing $T_{cln}, T_{rplc}, T_{insp}$	`trigger`	Full synchronisation
RM	DELAY module representing T_{rep}	`trigger_clean`	Full synchronisation
RM	DELAY module representing T_{oh}	`trigger_replace`	Full synchronisation
EBE	DELAY representing T_{deg}	`degrade`$_N$	Full synchronisation
DELAY representing T_{cln}	RM, EBE	`check_clean`	Full synchronisation
DELAY representing T_{rplc}	RM, EBE	`check_replace`	Full synchronisation
DELAY representing T_{insp}	RM, IM	`inspect`	Full synchronisation
DELAY representing T_{rep}	RM, IM, EBE	`perform_clean`	Full synchronisation
DELAY representing T_{oh}	RM, IM, EBE	`perform_replace`	Full synchronisation
EBE	RM, IM, all DELAY modules, other EBEs	-	Interleaving synchronisation

Fig. 14. Block diagram showing the synchronisation connections between one component and the other, together with the corresponding transition label, which triggers synchronisation.

the IM. We refer the reader to Table 1 to further understand the synchronisation between the FMT components and the method employed for parallel composition. Consider a simple example showing the time signals and synchronisations required for modelling an EBE and the RM and IM. The EBE has a degradation rate equal to T_{deg} and we limit the functionality of the RM and IM by allowing only the maintenance action to perform cleaning. We also need the corresponding DELAY modules generating the degradation rates, T_{deg} and the maintenance rates $T_{cln}, T_{insp}, T_{rep}$. The resulting CTMC is obtained by performing a parallel composition of the components $C_{all} = C_{EBE} \| C_{T_{deg}} \| C_{RM} \| C_{IM} \| C_{T_{cln}} \| C_{T_{insp}} \| C_{T_{rep}}$. The resulting state space is then $S_{all} = S_{EBE} \times S_{T_{deg}} \times S_{RM} \times S_{IM} \times S_{T_{cln}} \times S_{T_{insp}} \times S_{T_{rep}}$. The synchronisation between the different components is shown in Figure 14 and proceeds as follows:

(1) All the DELAY modules (except T_{cln}) start at the same time using the `trigger` transition label.
(2) When the extended DELAY module generating the T_{deg} time delay elapses, the corresponding EBE moves to the next state through synchronisation with the transition label `degrade`$_N$.

(3) The clock signals T_{rep}, T_{insp} represent periodic maintenance and inspection actions and when the deterministic delay is reached, through synchronisation with the transition label `check_clean` or the `inspect`, the RM or IM modules are triggered (cf. Figures 12 and 13). If RM triggers a maintenance action, then the DELAY representing T_{cln} is triggered using the synchronisation labels `trigger_clean`. Once the deterministic delay T_{cln} elapses, the EBE, the extended DELAY module representing T_{deg} (where the `reset` transition label within the extended DELAY module is replaced with `perform_clean`) and the IM are reset using the transition label `perform_clean`.

Remark 2. One should note that performing synchronisation results in a large state space, which is a function of the number of states used to approximate the deterministic delays. To counteract this effect, we propose an abstraction framework in Section 4.5.

4.4 Metrics

We use PRISM to compute the metrics of the model described in Section 3.2. The metrics can be expressed using the extended Continuous Stochastic Logic (CSL) as follows:

(1) *Reliability*: This can be expressed as the complement of the probability of failure over the time T, $1 - \text{P}_{=?}[\text{F}^{\leq T} \, failed]$.
(2) *Availability*: This can be expressed as $\text{R}_{=?}[C^{\leq T}]/T$, which corresponds to the cumulative reward of the total time spent in states labelled with *okay* and *thresh* during the time T.
(3) *Expected cost*: This can be expressed using $\text{R}_{=?}[C^{\leq T}]$, which corresponds to the cumulative reward of the total costs (operational, maintenance and failure) within the time T.
(4) *Expected number of failure*: This can be expressed using $\text{R}_{=?}[C^{\leq T}]$, which corresponds to the cumulative transition reward that counts the number of times the top event enters the *failed state* within the time T.

4.5 Decomposition of FMTs

The use of CTMC and deterministic time delays results in a large state space for modelling the whole FMT (cf. Remark 2). We therefore propose an approach that decomposes the large FMT into an equivalent abstract CTMC that can be analysed using PRISM. The process involves two transformation steps. First, we convert the FMT into an equivalent directed acyclic graph (DAG) and split this graph into a set of smaller sub-graphs. Second, we transform each sub-graph into an equivalent CTMC by making use of the developed FMT components semantics (cf. Section 4.2), and performing parallel composition of the individual FMT components based on the underlying structure of the sub-graph. The smaller sub-graphs are then sequentially composed to generate the higher level abstract FMT. Figure 15 depicts a high-level diagram of the decomposition procedure.

Conversion of the Original FMT to the Equivalent Graph. The FMT is a DAG (cf. Section 4), and in this framework we need to apply a transformation to the DAG in the presence of an RDEP gate, such that we can perform the decomposition. The RDEP causes an acceleration of events on dependent children nodes when the input node fails. To capture this feature in a DAG, we need to duplicate the input node such that it is connected directly to the RDEP vertex. This allows us to capture when the failure of the input occurs and the corresponding acceleration of the the children. This is reasonable as the same RM and IM are used irrespective of the underlying FMT structure.

Graph Decomposition. We define modules within the DAG as sub-trees composed of at least two events, which have no inputs from the rest of the tree and no outputs to the rest except from its output event [20]. We can divide the graph into multiple partitions based on the number of
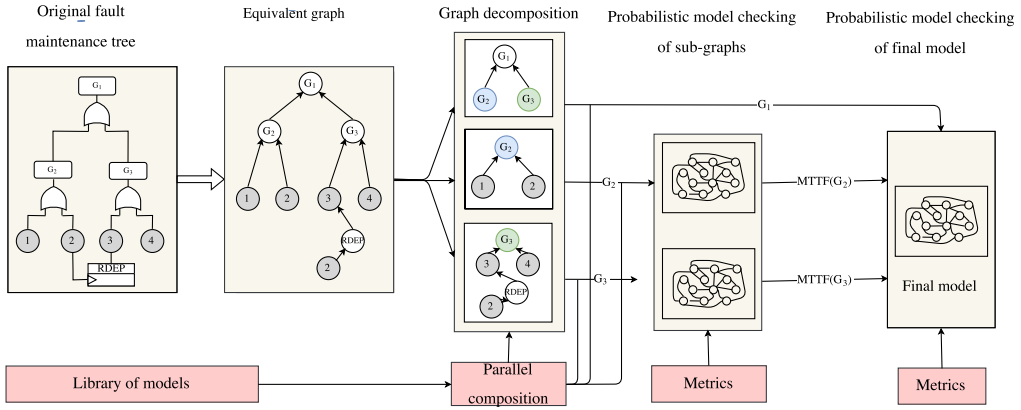
Fig. 15. Overall developed framework for decomposition of FMTs into the equivalent abstract CTMCs.

modules making up the DAG. We define the following notations to ease the description of the algorithm:

- V_o indicates whether the node is the top node of the DAG.
- V_g indicates the node where the graph split is performed.
- Modules correspond to sub-graphs in DAG.

We set V_o when we construct the DAG from the FMT and then proceed with executing Algorithm 1. We first identify all the sub-graphs within the whole DAG and label all the top nodes of each sub-graph i as V_{Ti}. We loop through each sub-graph and its immediate child (the sub-graph at the immediate lower level) and at the point where the sub-graph and child are connected, the two graphs are split and a new node V_g is introduced. Thus, executing Algorithm 1 results in a set of sub-graphs linked together by the labelled nodes V_g. For each of the lower-level sub-graphs, we now proceed to compute the mean time to failure (MTTF). This will serve as an input to the higher-level sub-graphs, such that metrics for the abstract equivalent CTMC can be computed.

ALGORITHM 1: DAG decomposition algorithm

    Input: DAG $G = (V, E)$
    Output: Set of sub-graphs with one of the end nodes labelled as V_g.

1 Identify sub-graphs using 'depth-first' traversal
2 Label all top nodes of each sub-graph i as V_{T_i}
3 **forall the** *select the top node of every sub-graph and the child defined at the immediate lower level* **do**
4   **if** *label V_T already found in one of the leaf nodes of the sub-graph* **then**
5   Split sub-graph
6   Insert new node V_g, which will be used as input from connected sub-graph
7   **end**
8 **end**

PMC of Sub-graphs. We start from the bottom level sub-graphs and perform the conversion to CTMC using the formal models presented in Section 4.2. The formal models have been built into a library of PRISM modules and based on the underlying components and structure making up the sub-graph, the corresponding individual formal models are converted into the sub-graph's equivalent CTMC by performing parallel composition (cf. Section 4.3). For each sub-graph, we
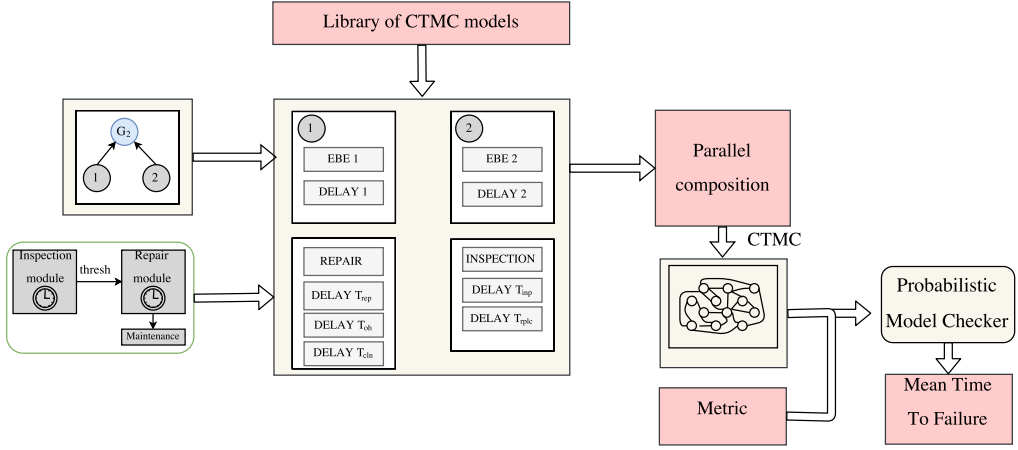
Fig. 16. PMC of sub-graphs.

compute the probability of failure $D_e(T)$ at time T, from which we calculate the MTTF [23] using

$$MTTF = \frac{\ln(1 - D_e(T))}{-T}.$$

The MTTF serves as the input to the higher level sub-graph at time T. The new node in the higher-level sub-graph, now degrades with the new time delay $T_{deg} = MTTF$, which is fed into the corresponding DELAY component. This process is repeated for all the different sub-graphs until the top level node V_o is reached. Figure 16 depicts the steps needed to perform PMC for one of the sub-graphs.

PMC of Final Equivalent Abstract CTMC. On reaching the top level node V_o, we compute the metrics for the equivalent abstract CTMC for a specific time horizon T. For different horizons, the previous step of computing the MTTF for the underlying lower level sub-graphs needs to be repeated. Using this technique, we can formally verify larger FMTs, while using less memory and computational time due to the significantly smaller state space of the underlying CTMCs. Next, we proceed with an illustrative example comparing the process of directly modelling the large FMT using CTMCs versus the de-compositional modelling procedure. Figure 17 presents the FMT composed of two modules and the corresponding abstracted FMT. The abstract FMT is a pictorial representation of the model represented by the equivalent abstract CTMC obtained using the developed decomposition framework (cf. Figure 15). For both the large FMT and the equivalent abstract FMT a comparison between the total number of states for the resulting CTMC models, the total time to compute the reliability metric and the resulting reliability metric is performed. All computations are run on an 2.3GHz Intel Core i5 processor with 8GB of RAM and the resulting statistics are listed in Table 2. The original FMT has a state space with 193,543 states, while the equivalent abstract CTMC has a state space with 63,937 states. This corresponds to a 67% reduction in the state space size. The total time to compute the reliability metric is a function of the final time horizon and a maximal 73% reduction in computation time is achieved. Accuracy in the reliability metric of the abstract model is a function of the time horizon and the number of states used to approximate the deterministic delay representing the computed MTTF. The larger the number of states the more accurate the representation of the MTTF, but this comes at a cost on the size of the underlying CTMC model. In our case, $N = 4$ is chosen. The accuracy of the reliability metric computed by the abstract FMT results in a maximal reduction of 0.61%.
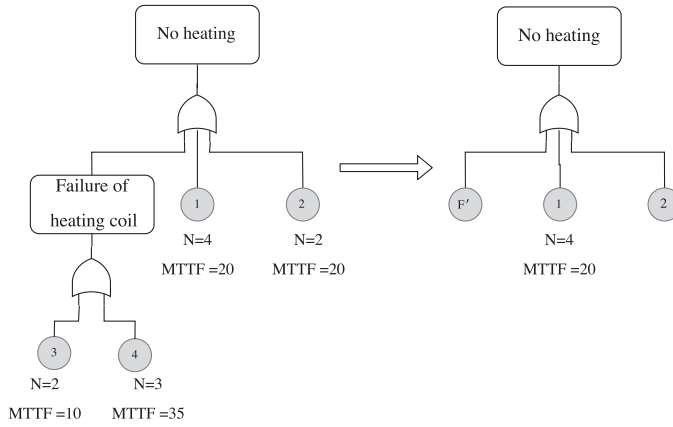
Fig. 17. The original FMT and the abstract FMT corresponding to the equivalent abstract CTMC generated by the developed framework. The MTTF for the F′ is computed based on the probability of failure of the heating coil.

Table 2. Comparison Between the Original Large FMT and the Abstracted FMT

Time Horizon	Original FMT		Abstracted FMT			
	Time to compute metric	Reliability	Time to compute MTTF	metric	Total Time	Reliability
(years)	(mins)		(mins)	(mins)	(mins)	
5	**0.727**	0.9842	0.142	0.181	**0.223**	0.9842
10	**1.406**	0.8761	0.219	0.309	**0.528**	0.8769
15	**2.489**	0.3290	0.292	0.622	**0.914**	0.3270

5 CASE STUDY

We apply the FMT framework to a Heating, Ventilation, and Air-conditioning (HVAC) system used to regulate a building's internal environment (cf. Section 2). Based on this HVAC system, we construct the corresponding FMT shown in Figure 18. The FMT structure follows the structure of the underlying HVAC system, as can be seen from the colour shading used in Figure 18. The leaves of the tree are EBE with discrete degradation rates computed using Table 3, approximated by the Erlang distribution where N is the number of degradation phases ($k = N$ for the Erlang distribution) and MTTF is the expected time to failure with $MTTF = 1/\lambda$ (cf. Remark 1). We choose an acceleration factor $\gamma = 2$ for the RDEP gate. The system is periodically cleaned every T_{rep} months and a major overhaul with a complete replacement of all components is carried out once every T_{oh} years. Inspections are performed every T_{insp} months and return the components back to the previous state, corresponding to a cleaning action. The total time to perform a cleaning action is 1 day ($T_{cln} = 1\ day$), while performing a total replacement of components takes 7 days ($T_{rplc} = 7\ days$). The time timing signals $\{T_{rep}, T_{oh}, T_{insp}, T_{cln}, T_{rplc}\}$ are all approximated using the Erlang distribution with $N = 3$. All maintenance actions are performed simultaneously on all components.

5.1 Quantitative Results

In the following sections, we employ the developed framework (cf. Section 4.5) to the FMT representing the failure of the HVAC system (cf. Figure 18) and perform three different experiments.

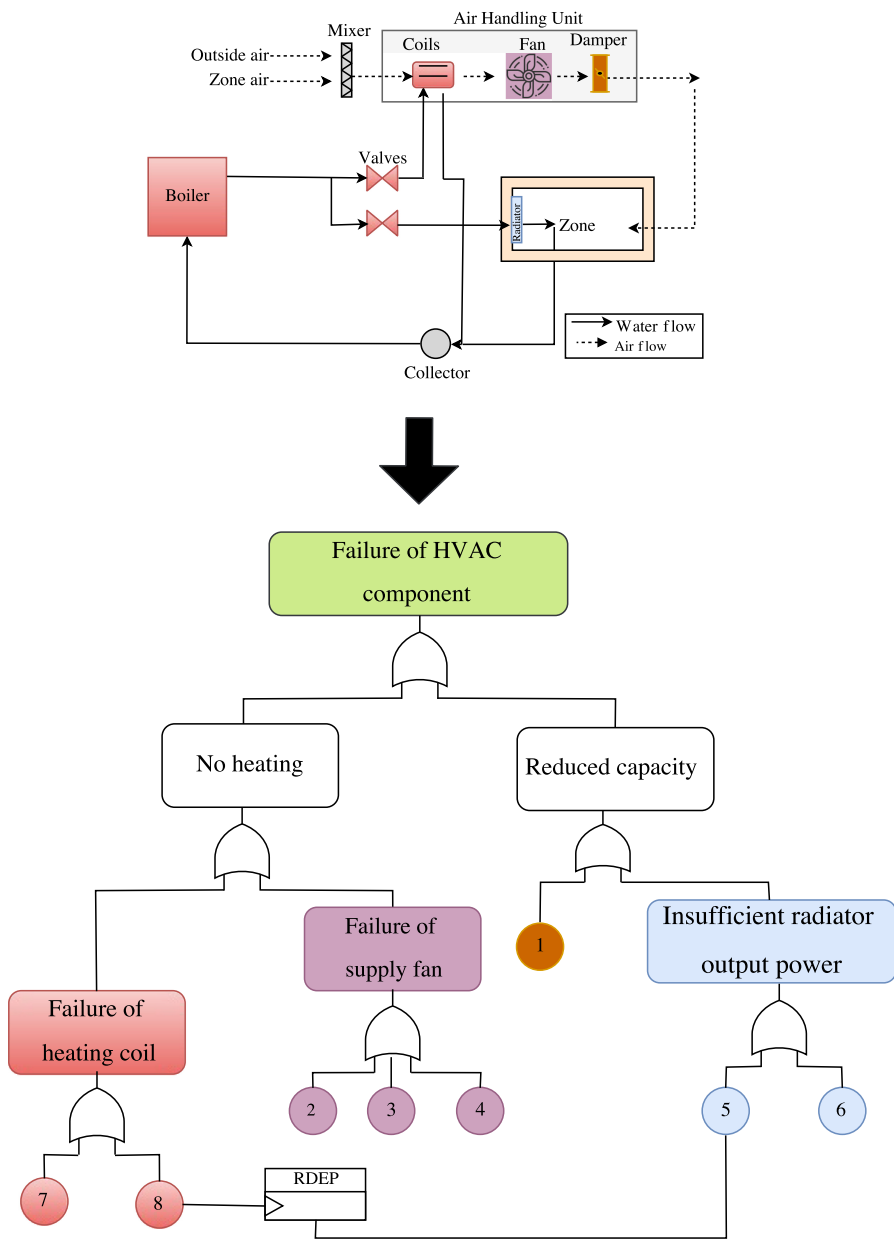Fig. 18. FMT for failure in HVAC system with leaves represented using EBE (associated RM and IM not shown in figure). The EBE are labelled to correspond to the component failure they represent using the fault index presented in Table 3. The EBE and intermediate events are colour coded such that they correspond to the different HVAC components thus showing how the propagation of faults in the HVAC is reflected within the FMT.

Table 3. Extended Basic Events in FMT
with Associated Degradation Rates (N, MTTF)
Obtained from References [2, 15]

Fault Index	Failure Mode	N	MTTF (years)
1	Broken AHU Damper	4	20
2	Fan motor failure	3	35
3	Obstructed supply fan	4	31
4	Fan bearing failure	6	17
5	Radiator failure	4	25
6	Radiator stuck valve	2	10
7	Heater stuck valve	2	10
8	Failure in heat pump	4	20

We first demonstrate the use of the developed framework by converting the FMT for the HVAC set-up into an abstract CTMC. For this abstract CTMC, we compute the metrics (cf. Section 4.4) using probabilistic model checking to show the type of analysis that can be performed using the set-up. Next, we perform a comparison between different maintenance strategies applied to the same FMT. This allows the user to deduce the optimal strategy for the set-up. Last, we construct a FMT, which does not employ the repair and inspection module and compare it with the original FMT (includes the maintenance modules) to further highlight the advantage of incorporating maintenance.

Applying the Framework to HVAC Set-up. We convert the FMT representing the failure of the HVAC system into the equivalent abstract CTMC and perform probabilistic model checking over six time horizons $N_r = \{0, 5, 10, 15, 20, 25\}$ years with the maintenance policy consisting of periodic cleaning every $T_{rep} = 2$ years and inspections every $T_{insp} = 1$ year. No replacement actions are considered. For this set-up, all the metrics corresponding to the reliability, availability, total costs (maintenance, inspection, and operational costs) and the total expected number of failures of the HVAC systems over the time horizon are computed and are shown in Figure 19. The total maintenance cost to perform a clean is 100[GBP], while an inspection cost is 50[GBP]. The maximal time taken to compute a metric using the abstract FMT is 1.47min. It is deduced that the reliability reduces over time. The availability is seen to be nearly constant, while the expected number of failures increases until it reaches a steady-state value. This shows that there is a saturation in the number of maintenance actions that one can perform before the system no longer achieves higher performance in reliability and availability. One can further note that, as expected, the maintenance costs increases linearly with time.

Comparison between Different Maintenance Strategies. In this second experiment, we compare all the metrics (reliability, availability, total costs, and expected number of failures) over the time horizon $N_r = \{0, 5, 10, 15, 20, 25\}$ years when considering different maintenance strategies, such that we can identify the optimal maintenance strategy that minimises cost and achieves the best trade-off in HVAC performance (i.e., with minimal expected number of failures and high reliability and availability). We consider five different maintenance strategies, which are listed in Table 4.

We select strategies that have a different combination of repair, inspection, and replacement strategies to highlight the effect the different maintenance actions have on the HVAC system's performance. Figure 20 depicts the resulting metrics for the employed strategies.
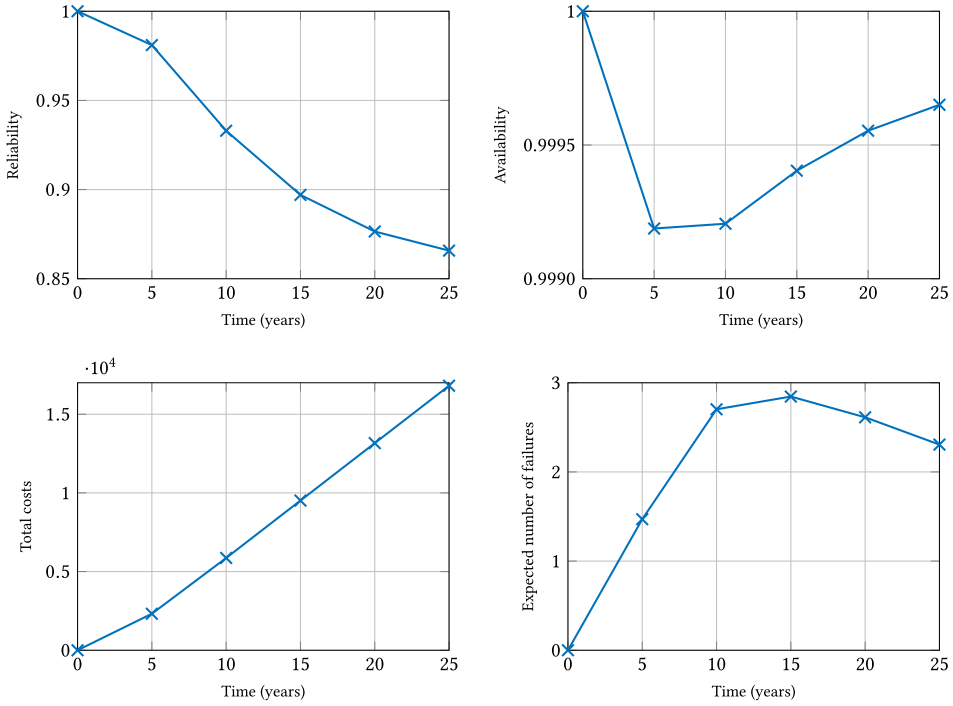
Fig. 19. Reliability, availability, total costs, and expected number of failures of HVAC over time horizon $N_r = \{0, 5, 10, 15, 20, 25\}$.

Table 4. Implemented Maintenance Strategies

Strategy index	T_{rep}	T_{oh}	T_{insp}
M_0	2 years	—	1 year
M_1	5 years	—	2 years
M_2	2 years	5 years	—
M_3	2 years	10 years	1 year
M_4	2 years	20 years	6 months

We can deduce that the worst performing strategy is when cleaning actions are carried out every 5 years with inspection carried out bi-annually and no replacements (corresponding to strategy M_1). Strategies M_2 and M_3 have comparable high performance but with a significant increase in the total costs due to the replacement action. We witness the highest costs using strategy M_2 due to the frequent replacement of the HVAC system. Comparing strategies M_3 and M_4, we can note that M_3 has fewer number of failures over the whole time horizon but this comes with higher total costs due to the replacements. Strategies M_0 and M_4 have similar performance with M_0 having a slightly lower availability and higher expected number of failures but with comparable maintenance costs. From this analysis, we can deduce that the optimal strategy, which gives the best trade-off between costs and HVAC system's performance, is strategy M_0 (i.e., with annual inspections, bi-annual cleaning, and no replacements).

Comparison between Performing Maintenance and No Maintenance. Last, we compare the performance of the HVAC system without performing any maintenance actions vs. the HVAC
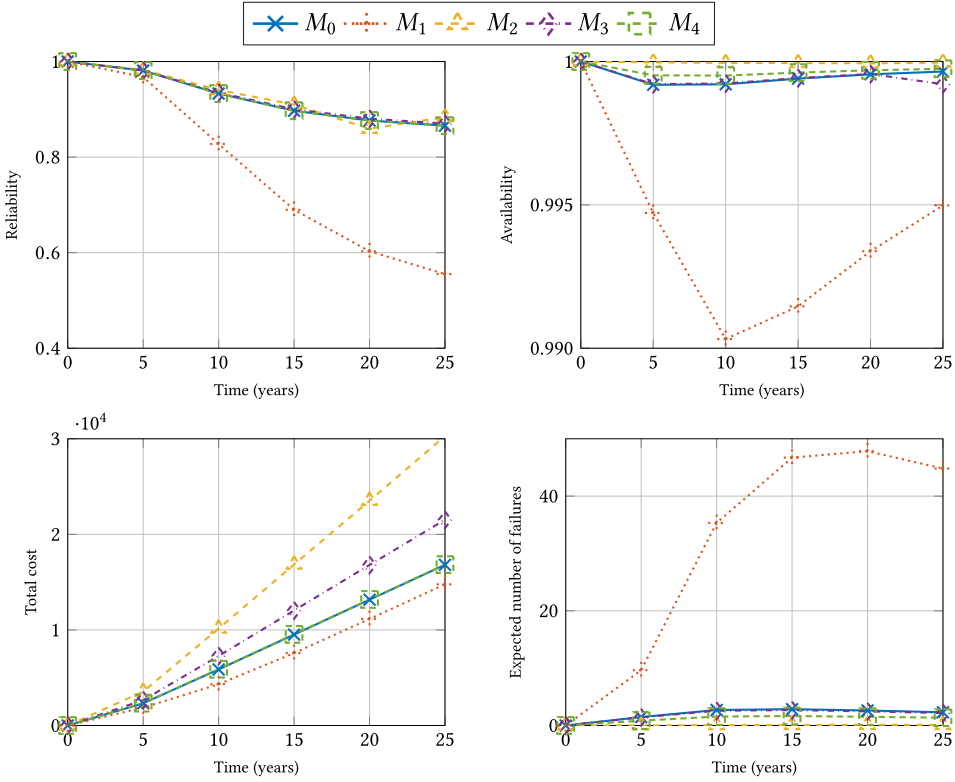
Fig. 20. Comparison between different number of maintenance strategies for an HVAC systems.

system with annual inspections, bi-annual cleaning, and a major overhaul after 10 years. We employ the developed framework to represent the FMT of the HVAC system, first without incorporating the repair and inspection modules and then incorporating the repair and inspection modules with $T_{insp} = 1$ year, $T_{rep} = 2$ years, and $T_{oh} = 10$ years. The obtained results, depicted in Figure 21, highlight the importance of maintenance and how appropriate maintenance strategies are required to maintain a reliable and available HVAC. When no maintenance is performed, both the reliability and availability of the HVAC system are gradually reduced, while the expected number of failures increases, as the components are degrading with time. This is in contrast to when maintenance is performed where high performance values of reliability and availability are achieved and the expected number of failures are low, throughout the whole time horizon. One should note that this comes at a price, where the total costs increase when maintenance is applied. Consequently, this further highlights the need to perform an analysis to deduce the optimal maintenance strategy that gives the best trade-off between costs, reliability, availability, and the expected number of failures.

6 CONCLUSION AND FUTURE WORKS

The article presents a methodology for applying probabilistic model checking to FMTs. We model FMTs using CTMCs, which simplify the transformation of FMT into formal models that can be analysed using PRISM. We further present a novel technique for abstracting the equivalent CTMC model. The novel decomposition procedure tackles the issue of state space explosion and results in a significant reduction in both the state space size and the total time required to compute metrics.
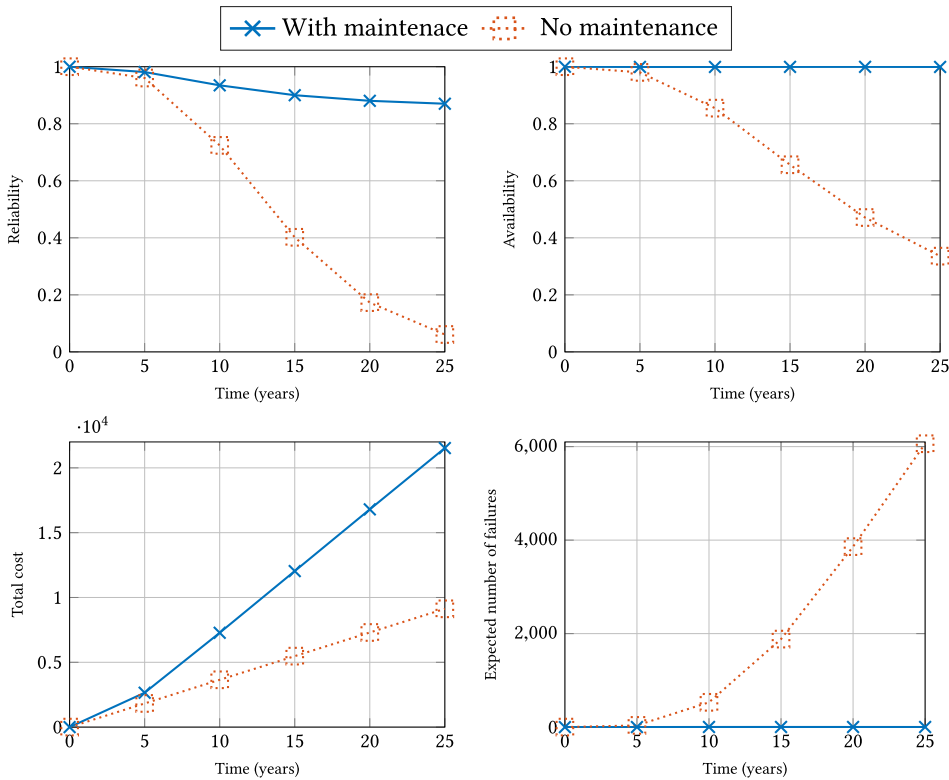
Fig. 21. Comparison between incorporating the maintenance modules vs. performing no maintenance.

The framework is applied to an HVAC system and a set of different experiments to demonstrate the use of the developed framework and to highlight (i) the importance of performing maintenance and (ii) the effect of applying different maintenance strategies has been presented. The presented framework can be further enhanced by adding more gates to the PRISM modules library, which include the Priority-AND, INHIBIT, and k/N gates, and to incorporate lumping of states as in Reference [26].

ACKNOWLEDGMENTS

REFERENCES

[1] Marwan Ammar, Khaza Anuarul Hoque, and Otmane Ait Mohamed. 2016. Formal analysis of fault tree using probabilistic model checking: A solar array case study. In *Proceedings of the Annual IEEE Systems Conference (SysCon'16)*. IEEE, 1–6.

[2] Handbook ASHRAE. 1996. HVAC systems and equipment. American Society of Heating, Refrigerating, and Air Conditioning Engineers, Atlanta, GA, 1–10.

[3] Vladimir Babishin and Sharareh Taghipour. 2016. Optimal maintenance policy for multicomponent systems with periodic and opportunistic inspections and preventive replacements. *Appl. Math. Model.* 40, 24 (2016), 10480–10505.

[4] Francesca Boem, Riccardo M. G. Ferrari, Christodoulos Keliris, Thomas Parisini, and Marios M. Polycarpou. 2017. A distributed networked approach for fault detection of large-scale systems. *IEEE Trans. Automat. Control* 62, 1 (2017), 18–33.

[5] Luca Bortolussi and Jane Hillston. 2012. Fluid approximation of CTMC with deterministic delays. In *Proceedings of the 9th International Conference on Quantitative Evaluation of Systems (QEST'12)*. IEEE, 53–62.

[6] Nathalie Cauchi, Khaza Anuarul Hoque, Alessandro Abate, and Mariëlle Stoelinga. 2017. Efficient probabilistic model checking of smart building maintenance using fault maintenance trees. In *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*. ACM, 24.

[7] Nathalie Cauchi, Karel Macek, and Alessandro Abate. 2017. Model-based predictive maintenance in building automation systems with user discomfort. *Energy* 138 (2017), 306–315.

[8] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* 8 (1986), 244–263.

[9] Frits Dannenberg, Marta Kwiatkowska, Chris Thachuk, and Andrew J. Turberfield. 2013. DNA walker circuits: Computational potential, design, and verification. In *Proceedings of the International Workshop on DNA-Based Computers*. Springer, 31–45.

[10] Lu Feng, Clemens Wiltsche, Laura Humphrey, and Ufuk Topcu. 2015. Controller synthesis for autonomous systems interacting with human operators. In *Proceedings of the ACM/IEEE 6th International Conference on Cyber-Physical Systems*. ACM, 70–79.

[11] Holger Hermanns and Lijun Zhang. 2011. From concurrency models to numbers. In *Nato Science for Peace and Security Series*. IOS Press.

[12] Khaza Anuarul Hoque, Otmane Ait Mohamed, and Yvon Savaria. 2015. Towards an accurate reliability, availability and maintainability analysis approach for satellite systems based on probabilistic model checking. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 1635–1640.

[13] Khaza Anuarul Hoque, Otmane Ait Mohamed, and Yvon Savaria. 2017. Formal analysis of SEU mitigation for early dependability and performability analysis of FPGA-based space applications. *J. Appl. Logic* 25 (2017), 47–68.

[14] Khaza Anuarul Hoque, O. Ait Mohamed, Yvon Savaria, and Claude Thibeault. 2014. Probabilistic model checking based DAL analysis to optimize a combined TMR-blind-scrubbing mitigation technique for FPGA-based aerospace applications. In *Proceedings of the 12th ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'14)*. IEEE, 175–184.

[15] Faisal I. Khan and Mahmoud M. Haddara. 2003. Risk-based maintenance (RBM): A quantitative approach for maintenance/inspection scheduling and planning. *J. Loss Prevent. Process Industr.* 16, 6 (2003), 561–573.

[16] Marta Kwiatkowska, Gethin Norman, and David Parker. 2007. Stochastic model checking. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. Springer, 220–270.

[17] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11) (LNCS)*, G. Gopalakrishnan and S. Qadeer (Eds.), Vol. 6806. Springer, 585–591.

[18] Marta Kwiatkowska, Gethin Norman, and David Parker. Advances and challenges of probabilistic model checking. In *Proceedings of the 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton'10)*. IEEE.

[19] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical model checking: An overview. *RV* 10 (2010), 122–135.

[20] Z. F. Li, Yi Ren, L. L. Liu, and Z. L. Wang. 2015. Parallel algorithm for finding modules of large-scale coherent fault trees. *Microelectronics Reliability* 55, 10 (2015), 1400–1403. In *Proceedings of the 26th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF'15)*.

[21] Karel Macek, Petr Endel, Nathalie Cauchi, and Alessandro Abate. 2017. Long-term predictive maintenance: A study of optimal cleaning of biomass boilers. *Energy Build.* 150 (2017), 111–117.

[22] Enno Ruijters, Dennis Guck, Peter Drolenga, and Mariëlle Stoelinga. 2016. Fault maintenance trees: Reliability centered maintenance via statistical model checking. In *Proceedings of the Annual Reliability and Maintainability Symposium (RAMS'16)*. IEEE, 1–6.

[23] Enno Ruijters and Mariëlle Stoelinga. 2015. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Comput. Sci. Rev.* 15 (2015), 29–62.

[24] Umair Siddique, Khaza Anuarul Hoque, and Taylor T. Johnson. 2017. Formal specification and dependability analysis of optical communication networks. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'17)*. IEEE, 1564–1569.

[25] Ying Yan, Peter B. Luh, and Krishna R. Pattipati. 2017. Fault diagnosis of HVAC air-handling systems considering fault propagation impacts among components. *IEEE Trans. Auto. Sci. Eng.* 14, 2 (Apr. 2017), 705–717.

[26] Olexandr Yevkin. 2016. An efficient approximate Markov chain method in dynamic fault tree analysis. *Qual. Reliabil. Eng. Int.* 32, 4 (2016), 1509–1520.

[27] Håkan L. S. Younes, Marta Kwiatkowska, Gethin Norman, and David Parker. 2006. Numerical vs. statistical probabilistic model checking. *Int. J. Softw. Tools Technol. Transfer* 8, 3 (2006), 216–228.

[28] Xiaojun Zhou, Lifeng Xi, and Jay Lee. 2007. Reliability-centered predictive maintenance scheduling for a continuously monitored system subject to degradation. *Reliabil. Eng. Syst. Safe.* 92, 4 (2007), 530–534.

RIGHTSLINK()