Efficient Probabilistic Model Checking of Smart Building Maintenance using Fault Maintenance Trees

Nathalie Cauchi*
Department of Computer Science, University of Oxford
Oxford, United Kingdom
nathalie.cauchi@cs.ox.ac.uk

Khaza Anuarul Hoque
Department of Computer Science, University of Oxford
Oxford, United Kingdom
khaza.hoque@cs.ox.ac.uk

Alessandro Abate
Department of Computer Science, University of Oxford
Oxford, United Kingdom
aabate@cs.ox.ac.uk

Mariëlle Stoelinga
FMT Group, University of Twente
Twente, The Netherlands
marielle@cs.utwente.nl

ABSTRACT

Cyber-physical systems, like Smart Buildings and power plants, have to meet high standards, both in terms of reliability and availability. Such metrics are typically evaluated using Fault trees (FTs) and do not consider maintenance strategies which can significantly improve lifespan and reliability. Fault Maintenance trees (FMTs) – an extension of FTs that also incorporate maintenance and degradation models, are a novel technique that serve as a good planning platform for balancing total costs and dependability of a system. In this work, we apply the FMT formalism to a Smart Building application. We propose a framework for modelling FMTs using probabilistic model checking and present an algorithm for performing abstraction of the FMT in order to reduce the size of its equivalent Continuous Time Markov Chain. This allows us to apply the probabilistic model checking more efficiently. We demonstrate the applicability of our proposed approach by evaluating various dependability metrics and maintenance strategies of a Heating, Ventilation and Air-Conditioning system's FMT.

CCS CONCEPTS

•**Computer systems organization** → *Maintainability and maintenance;*

KEYWORDS

Fault Maintenance Trees, Formal modelling, Probabilistic Model checking, Reliability, Building Automation Systems, PRISM

ACM Reference format:
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 International Conference on Systems for Energy-Efficient Built Environments, Delft, The Netherlands, November 6–8 2017 (BuildSys),* 10 pages.
DOI: 00.0001

*The corresponding author

BuildSys, Delft, The Netherlands

DOI: 00.0001

1 INTRODUCTION

Worldwide, buildings account for approximately 40% of the total energy consumption and 20% of the total CO_2 emissions, annually [5]. Efficient Building Automation Systems (BAS) can reduce energy consumption by up to 30% through their optimal operation, continuous commissioning and maintenance [5]. Constructions employing such technologies are termed *Smart Buildings*. High standards have to be adhered by such technologies, both in terms of reliability and availability. One way of achieving this is by employing methods to perform preventative and predictive maintenance actions. Diagnostic and fault detection techniques for Smart Building applications have been developed in [2, 15]. Predictive and preventative maintenance strategies are devised in [1, 4]. However, these techniques preclude availability and reliability measurements and focus only on synthesis of maintenance policies in the presence of degradation and fault finding. Reliability and availability are typically tackled using Fault Trees (FTs), where the focus is on finding the root causes of a system failure using a top-down approach. FTs do not include maintenance strategies in the analysis – a key element in reducing component failures. [14] presents the Fault Maintenance Tree (FMT) as 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, FMTs are analysed using Statistical Model Checking technique (SMC) [14] and provide statistical guarantees. In contrast, Probabilistic Model Checking (PMC), based on numerical analysis, provide formal guarantees with higher accuracy when compared with SMC [17]. However, numerical methods are far more memory intensive and may result in a state space explosion. This limitation of PMC often leaves SMC as the last resort [17]. In this paper we tackle the FMT analysis using PMC. Our contributions can be summarised as follows:

(1) We formalise the FMT framework using Continuous Time Markov Chain (CTMCs).
(2) We formalise the dependability metrics using the extended Continuous Stochastic Logic (CSL) formalism such that they can be computed using the PRISM model checker [12].

(3) To mitigate the state space explosion problem, we present an FMT abstraction technique which decomposes a large FMT into an equivalent abstract FMT based on our proposed graph decomposition algorithm. Using our framework, we are able to achieve a 67% reduction in the state space size.
(4) Finally, we construct a FMT that identifies failure of a Heating, Ventilation and Air-conditioning system (HVAC). We apply the developed framework to the built FMT and evaluate relevant dependability metrics, together with different maintenance strategies using the PRISM model checker.

To the best of our knowledge, this is the first attempt to analyse FMTs using Probabilistic Model Checking and also the first application to Smart Building systems.

This article has the following structure: Section 2 introduces the fault maintenance trees and probabilistic model checking frameworks. This is followed by the developed methodology for modelling FMT using CTMCs and performing model checking in Section 3. The framework is applied to a heating, ventilation and air-conditioning (HVAC) case study which is presented in Section 4.

2 PRELIMINARIES

2.1 Fault maintenance trees framework

Fault trees are directed acyclic graphs (DAG) describing the combinations of component failures that lead to system failures. The leaves in the fault trees are called *basic events* and denote the system failures. The internal nodes of the graph are called *gates* and describe the different ways that failures can interact to cause other components to fail. The gates in a fault tree can be of several types and these include the AND gate, OR gate, k/N-gate [14].

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* - The basic events are modified to incorporate degradation models of the component the leaf represents. The degradation models represent different discrete levels of degradations the components can be in and are a function of time.
(2) *Rate Dependency Events* - A new gate introduced in [14], labelled as *RDEP* that accelerates the degradation rates of dependent child nodes and is depicted in Figure 1. 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.

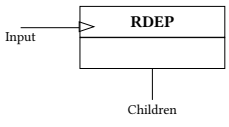


Figure 1: RDEP gate with 1 input and dependent components also known as children.

(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 IM performs periodic inspections and when components fall below a certain degradation threshold a repair or partial replacement is initiated by the IM to be performed by the RM.

2.2 Probabilistic model checking

Model checking 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* 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 extended Continuous Stochastic Logic (CSL) [11].

Definition 2.1. The tuple $C = (S, s_0, \text{TL}, \text{AP}, L, \mathbf{R})$ defines a CTMC which 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 $L : S \rightarrow 2^{AP}$ and the transition rate matrix $\mathbf{R} : S \times S \rightarrow \mathbf{R}_{\geq 0}$. 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, a steady-state operator that refers to the stationary probabilities, and a probabilistic operator 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. For detail about the syntax and semantics of CSL (which also includes reward formulae), we refer the interested readers to [11]. Examples of a CSL property with its natural language translation are: (i) $\text{P}_{\geq 0.95}[\text{F } \textit{complete}]$ - "The probability of the system eventually completing its execution successfully is at least 0.95". (ii) $\text{R}_{=?}[\text{F } \textit{success}]$ - "What is the expected reward accumulated before the system successfully terminates?"

3 FORMALIZING FMTS USING CTMCS

In this section, we first formalise the FMT framework by presenting the formal syntax and semantics for modelling FMTs using CTMCs. Next, we list the set of metrics used to analyse the FMT. Finally, we present the developed framework which allows us to analyse large FMTs using probabilistic model checking (PMC).

3.1 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 4.

Definition 3.1. The set $\mathcal{F}$ 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, $T_{deg}, T_{cln}, T_{rplc}, T_{rep}, T_{oh}, T_{insp} \in \mathbb{R}^{\geq 0}$ are deterministic delays, and $\gamma \in \mathbb{R}^{\geq 0}$ is a rate.

- $(EBE, T_{deg}, T_{cln}, T_{rplc}, N)$ 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 rate 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.
- $(RM, n, T_{rep}, T_{oh}, T_{insp}, T_{cln}, T_{rplc}, thresh, trig)$ 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.
- $(IM, n, T_{insp}, T_{cln}, T_{rplc}, thresh)$ 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), the *thresh* signal is activated and is sent to the RM. Once a repair 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 number of 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. 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 [1]. This DAG formulation allows us to propose a framework in Subsection 3.5 such that we can efficiently perform probabilistic model checking.

Definition 3.2. A fault maintenance tree is a directed acyclic graph $G = (V, E)$ composed of vertices V and edges E.

3.2 Semantics of FMT elements

Next, we provide the CTMC semantics for each FMT element $f \in \mathcal{F}$. These elements are then instantiated based on the underlying FMT structure to form the semantics of the whole FMT in CTMC form.

DELAY. We define the semantics for the $(DELAY, T, N)$ element using Figure 2(a) 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 2(a) 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 2.1). In Figure 2(b) 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

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

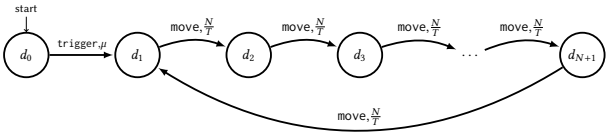
with i representing the current state and j is the next state. In both instances, the deterministic delays is approximated using an Erlang distribution [8] and all DELAY modules are synchronised to start together using the `trigger` transition label. The extended DELAY module have the transition labels `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 Subsection 3.3). 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.

[1] Note, for different FMT structure same RM and IM modules are used, thus RM and IM modules are independent of FMT structure

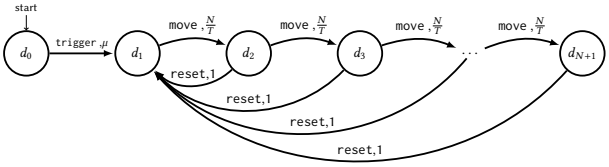
Remark 1. *The basic properties of an Erlang distribution: A random variable $Z \in \mathbb{R}_+$ has an Erlang distribution with $k \in \mathbb{N}$ stages and a rate $\lambda \in \mathbb{R}_+$, $Z \sim Erlang(k, \lambda)$, if $Z = Y_1 + Y_2 + \ldots Y_k$ where each Y_i is exponentially distributed with 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 $Z_k \sim Erlang(k, \frac{k}{T})$ [3]. Note, there 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) [8, 9]. 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 [14] where degradation phases are approximated by an (k,λ)-Erlang distribution.



(a) 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.



(b) 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.

Figure 2: CTMC for (a) DELAY and (b) DELAY with reset guard.

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 3 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 ,

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

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:

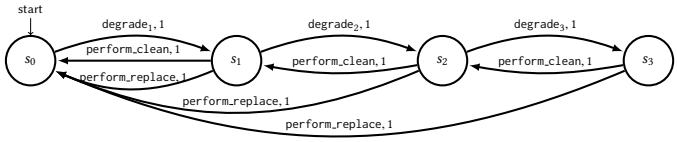


Figure 3: CTMC representing the EBE with $N = 3$ with the transition labels $\text{TL}_{EBE} = \{\texttt{degrade}_{i \in \{1,2,3\}}, \texttt{perform\_clean}, \texttt{perform\_replace}\}$ on each of the state transitions. The state labels are not shown and the initial state is pointed to by the arrow labelled with start.

(1) an extended DELAY module approximating T_{deg} with the transition label `move` replaced with $\texttt{degrade}_N$ such that synchronisation between the two CTMCs is performed (explained in Subsection 3.3). When T_{deg} has elapsed the transition labelled with $\texttt{degrade}_N$ is triggered and the EBE moves to the next state at a rate equal to $\frac{N}{T_{deg}} \times 1$[2]. The `reset` transition label and corresponding transitions are replicated in extended DELAY module and replaced with `perform_clean` and `perform_replace`. When the corresponding maintenance action is performed one of the transition label is triggered and the state of the EBE moves to 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 $\frac{N}{T_{cln}}$.
(3) a DELAY module approximating T_{rplc} with the transition label `move` replaced with `perform_replace`. When T_{rplc} has elapsed the transition having 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 ($in = 1$), 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 *in* signal using,

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

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

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

[2]This is a direct consequence of synchronisation and corresponds to $\mathbf{R} \times \mathbf{R}_{EBE}$. Refer to Subsection 3.3

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, 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 4 (a) shows the semantics of (RM, n, $T_{rep}, T_{oh}, T_{insp}, T_{cln}, T_{rplc}, T_{rplc}$, *thresh*, *trig*). The CTMC is described using the state space $\{rm_0, rm_1\}$, the initial state rm_0, the transition labels TL_{RM} = {`inspect`, `check_clean`, `check_replace`, `trigger_clean`, `trigger_replace`}, the atomic propositions AP = {*maintenance*}, the labelling function $L(rm_0) = \{\emptyset\}, L(rm_1) = \{maintenance\}$ and with $\mathbf{R}_{IM} = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right]$. For the sake of clarity in Figure 4 (a), we used the transition labels `check_maintenance` and `trigger_maintenance`. The transition label `check_maintenance` and corresponding transitions are replicated and the transition labels replaced by `check_clean` or `check_replace` to allow for both type of maintenance checks. Similarly, the transition label `trigger_maintenance` and corresponding transitions are duplicated and the transition labels replaced by `trigger_clean` or `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 Subsection 3.3). The transition labels `trigger_clean` or `trigger_replace` correspond to the transition label `trigger` within the DELAY module approximating the deterministic delays T_{cln} and T_{rplc} respectively. The deterministic delays which trigger `inspect`, `check_clean` or `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 `inspect`, `check_clean` or `check_replace` respectively. The *thresh* signal is modelled using,

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

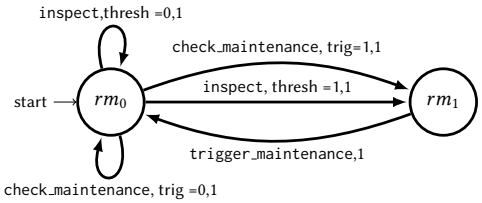
where $L(s_{j,i}), j \in 0 \ldots N, i \in 1 \ldots n$ correspond to the label of the current state j of each of the n EBE. Similarly, we model the *trig* signal using

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

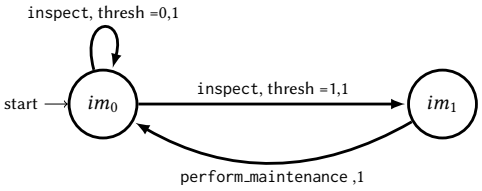
Both signals act as guards which when triggered determine which transition to perform (cf. Fig. 4 (a)).

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

Inspection module (IM) . The semantics of the (IM, n, T_{insp}, T_{cln}, T_{rplc}, *thresh*) is depicted in Figure 4 (b). 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, TL_{IM} = {`inspect`, `perform_clean`, `perform_replace`}, $\text{AP}_{IM} = \{\emptyset\}$, $L(s_0) = L(s_1) = \emptyset$ and $\mathbf{R}_{IM} = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right]$. The *thresh* signal corresponds to same signal used by the RM, given using (7). In Figure 4 (b), 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.



(a) 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.



(b) CTMC representing the IM with TL_{IM} = {`inspect`, `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.

Figure 4: CTMC for (a) RM and (b) IM.

3.3 Semantics of FMT

Next, we show how to obtain the semantics of a FMT from the semantics of its elements using the FMT syntax introduced in Subsection 3.1. 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 (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 [7] and defined as follows,

Definition 3.3 (Interleaving Synchronization). The interleaving synchronous product of $C_1 = (S_1, s_{01}, \mathrm{TL}_1, \mathrm{AP}_1, L_1, \mathbf{R}_1)$ and $C_2 = (S_2, s_{02}, \mathrm{TL}_2, \mathrm{AP}_2, L_2, \mathbf{R}_2)$ is $C_1||C_2 = (S_1 \times S_2, (s_{01}, s_{02}), \mathrm{TL}_1 \cup \mathrm{TL}_2, \mathrm{AP}_1 \cup \mathrm{AP}_2, L_1 \cup L_2, \mathbf{R})$ 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 $s_1, s_1' \in S_1$, $\alpha_1 \in \mathrm{TL}_1$, $\mathbf{R}_1(s_1, s_1') = \lambda_1$, $s_2, s_2' \in S_2$, $\alpha_2 \in \mathrm{TL}_2$, $\mathbf{R}_2(s_2, s_2') = \lambda_2$.

Definition 3.4 (Full Synchronization). The full synchronous product of $C_1 = (S_1, s_{01}, \mathrm{TL}_1, \mathrm{AP}_1, L_1, \mathbf{R}_1)$ and $C_2 = (S_2, s_{02}, \mathrm{TL}_2, \mathrm{AP}_2, L_2, \mathbf{R}_2)$ is $C_1||C_2 = (S_1 \times S_2, (s_{01}, s_{02}), \mathrm{TL}_1 \cup \mathrm{TL}_2, \mathrm{AP}_1 \cup \mathrm{AP}_2, L_1 \cup L_2, \mathbf{R})$ 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 $s_1, s_1' \in S_1$, $\alpha \in \mathrm{TL}_1 \wedge \mathrm{TL}_2$, $\mathbf{R}_1(s_1, s_1') = \lambda_1$, $s_2, s_2' \in S_2$, $\alpha_2 \in \mathrm{TL}_2$, $\mathbf{R}_2(s_2, s_2') = \lambda_2$.

For any pair of states, synchronisation is performed either using interleaving or full synchronisation. For full synchronisation, as in Definitions 3.3, 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 1. 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 1 (cf. Fig. 4 (a)), while the rate of the DELAY modules represent the actual rates (cf. Fig 2). The same principle holds for the EBEs and the IM. We refer the reader to Table 1 to further elucidate the synchronisation between the FMT components and the method employed during the parallel composition.

Example. *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 5 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* $\texttt{degrade}_N$.
(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 is triggered (cf. Fig. 4(a) and 4(b)). If RM triggers a maintenance action, 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`.

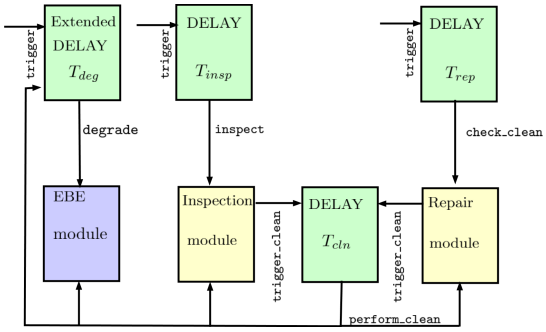


Figure 5: Block diagram showing the synchronisation connections between one component and the other, together with the corresponding transition label which trigger synchronisation.

Remark 2. *One should note that this results in the requirement of a large state space, which is a function of the number of states used to approximate the deterministic delays. Thus, to counteract this effect we propose an abstraction framework in Subsection 3.5.*

3.4 Metrics

We use PRISM to compute the metrics of the model described in Subsection 2.1. 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, $1 - \mathrm{P}_{=?}[\mathrm{F}^{\leq T} failed]$.
(2) *Availability*: This can be expressed as $\mathrm{R}_{=?}[C^{\leq T}]/T$, 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 $\mathrm{R}_{=?}[C^{\leq T}]$, 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 $\mathrm{R}_{=?}[C^{\leq T}]$, which corresponds to the cumulative transition reward that counts the number of times the top event enters the *failed state* within the time T.

3.5 Decomposition of FMTs

The use of CTMC and deterministic time delays results in the requirement of a large state space for modelling the whole FMT (cf. Remark 2). We therefore propose an approach which decomposes the large FMT into an equivalent abstract CTMC which can be analysed using PRISM. The process involves two transformation steps. First we convert the FMT into the equivalent directed acyclic graph (DAG) and split this graph into a set of smaller sub-graphs. Second, we transform the sub-graphs into the equivalent CTMC by making use of the developed FMT components semantics (cf.

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}	$\texttt{degrade}_N$	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	-	Interleave synchronisation

Table 1: Performing synchronisation between the different FMT components and the synchronisation method used.

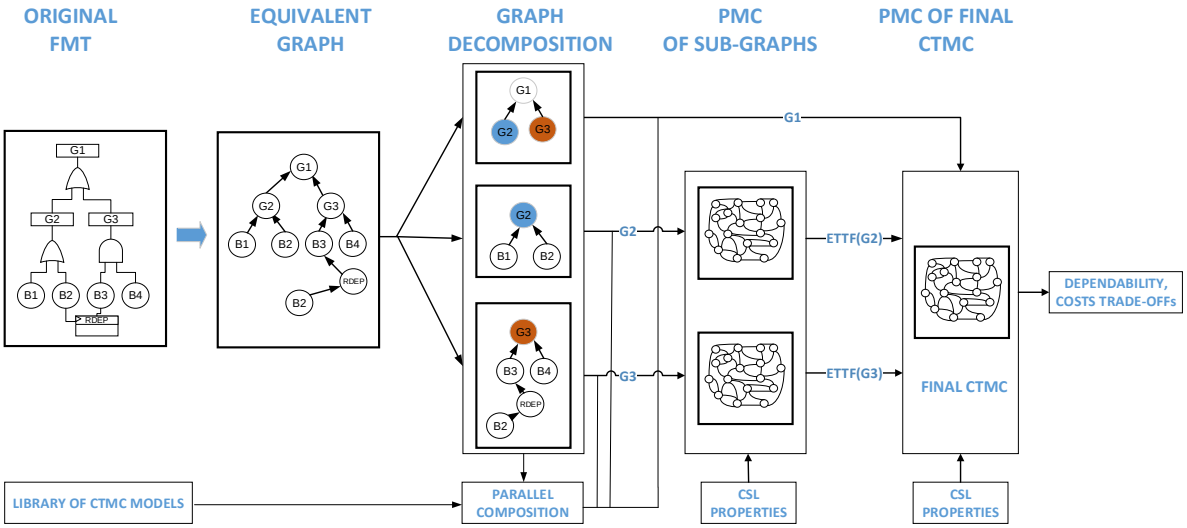


Figure 6: Overall developed framework for decomposition of FMTs into the equivalent abstract CTMCs.

Subsec. 3.2), 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 recomposed to generate the higher level abstract FMT. Figure 6 depicts a high-level diagram of the decomposition procedure.

Conversion of original FMT to the equivalent graph. The FMT is a DAG (cf. Subsection 3) 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 child nodes when the input node fails. In order 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 [13]. We can divide the graph into multiple partitions based on the number of modules making up the DAG. We define the following notations to ease in the description of the algorithm:

- V_o indicates whether the node is the top node of the DAG.
- V_g indicates the node where 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 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 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.

PMC of sub-graphs. We start from the bottom level sub-graphs and perform the conversion to CTMC using the formal models presented in Subsection 3.2. 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. Subsec. 3.3). For each sub-graph, we compute the probability of failure $D_e(T)$ at time

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.

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

T, from which we calculate the MTTF 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 a 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.

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 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 7 presents the FMT composed of two modules and the corresponding abstracted FMT. The abstract FMT is a pictorial representation of the moel represented by the equivalent abstract CTMC obtained using the developed decomposition framework (cf. Fig. 6). For

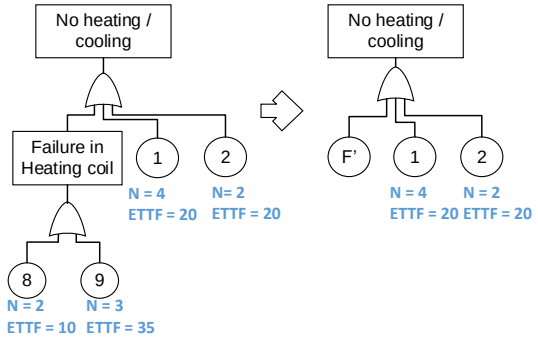


Figure 7: The original FMT and the abstract FMT corresponding to the equivalent abstract CTMC generated by the developed framework. The MTTF for the F' is computed based on the probability of failure of the heating coil.

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 2.3 GHz 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 193543 states, while the equivalent abstract CTMC has a state space with 63937 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. The accuracy of the reliability metric computed by the abstract FMT results in a maximal reduction of 0.61%.

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

Table 2: Comparison between the original large FMT and the abstracted FMT.

4 CASE STUDY

We apply the FMT framework to a Heating, Ventilation and Air-conditioning (HVAC) system used to regulate a building's internal environment. The HVAC system under consideration for the FMT analysis is presented in Figure 8. It is composed of two circuits - the air flow circuitry and the water circuit. The gas boiler heats up the supply water which is fed into the heat pump. The heat pump transfers the supply water into two sections - the supply air heating and cooling coils and the radiators - via the splitter. 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 extracted room air temperature 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 is then passed through the collector and is returned back to the boiler. Based on this HVAC system we construct the corresponding FMT shown in Figure 9. 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 repaired every 6 months ($T_{rep} = 182\ days$) and a major overhaul with a complete replacement of all components is carried out once every 20 years ($T_{oh} = 20 \times 365\ days$). Weekly inspections are performed ($T_{insp} = 7\ days$) which return the components back to the previous state. Only cleaning actions are performed when inspections are carried out. 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.

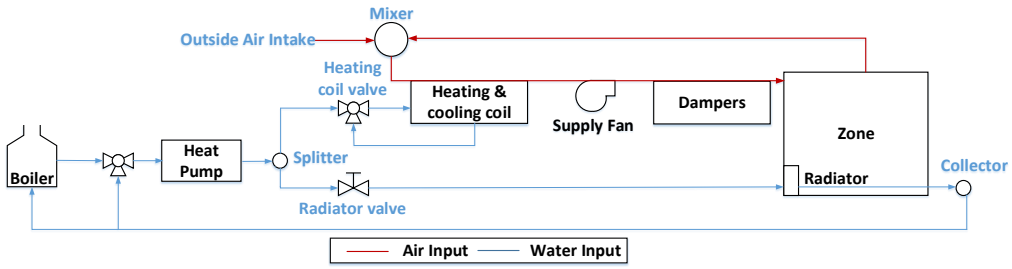


Figure 8: High level schematic ofan HVAC system.

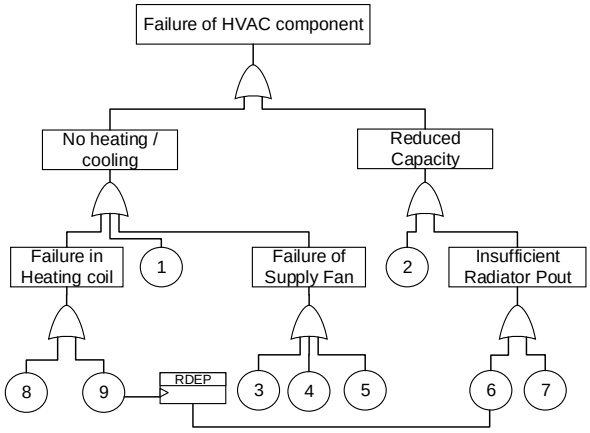


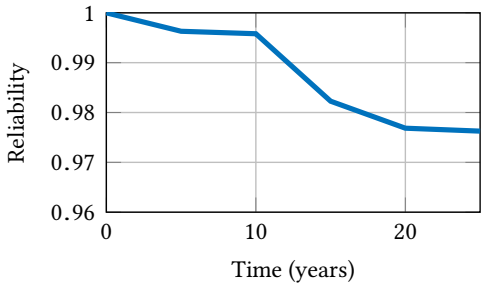
Figure 9: FMT for failure in HVAC system with leaves represented using EBE (associated RM and IM not shown in figure). The EBE are labelled to correspond to the component failure they represent using the fault index presented in Table 3.

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

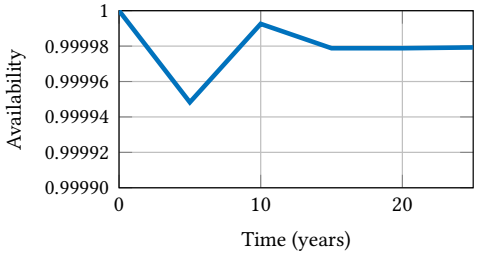
Table 3: Extended Basic events in FMT with associated degradation rates (N, MTTF) obtained from [6, 10].

4.1 Quantitative results

We make use of the developed framework (cf. Subsec. 3.5) and convert the FMT representing the failure of the HVAC system (cf. Fig. 9) into the equivalent abstract CTMC. The abstracted CTMC has a state space of 62779 states. Using our current computing set-up, the complex CTMC representing the whole FMT was not computable as it results in a state space explosion. Highlighting, the advantage of the developed framework. The process is performed over six time horizons $N_r = \{0, 5, 10, 15, 20, 25\}$ years with the maintenance policy consisting of periodic cleaning every 6 months, a major overhaul every 20 years and inspections on a weekly basis. For this set-up, the metrics corresponding to the reliability and availability of the HVAC systems over the time horizon are computed and are shown in Figure 10(b). The maximal time taken to compute a metric using the abstract FMT is 1.47 minutes. It is deduced that both the reliability and availability reduce over time and there is a saturation in the number of maintenance actions which one can perform before the system no longer achieves higher performance in reliability and availability. Next, we compare the total cost of



(a) Reliability of HVAC system.



(b) Availability of HVAC system.

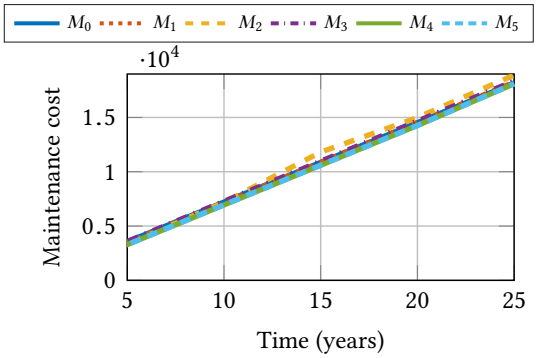
Figure 10: Reliability and availability of HVAC over time horizon N_r.

maintenance and the 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 maintenance strategy that minimises cost and the number of failures over time. We consider six different maintenance strategies which are listed in Table 4. The total maintenance cost to perform a repair is 100 [GBP], while a replacement costs 5000 [GBP]. We now compute the total expected maintenance costs and the total expected number of failures for each strategy. These are shown in Figure 11. The most effective strategy which offers a good trade-off between maintenance costs and the expected number of failures is achieved when repairs are carried out on a yearly basis, replacements are carried out every

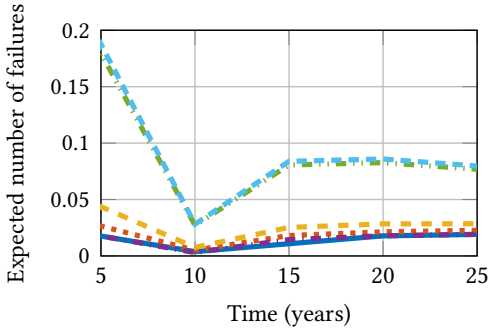
20 years and inspections are carried out weekly (corresponding to strategy M_1). Furthermore, it can be seen that the frequency of inspections has a large effect on the total number of failures. When the frequency of inspection is low (as in M_4 and M_5), the expected number of component failures increases significantly. Note that reducing the periodicity of repairs, as in the case of maintenance strategy M_2 also results in an increase in the expected number of failures.

Strategy index	T_{rep}	T_{oh}	T_{insp}
M_0	6 months	20 years	1 Week
M_1	12 months	20 years	1 Week
M_2	48 months	20 years	1 Week
M_3	6 months	10 years	1 Week
M_4	6 months	20 years	2 years
M_5	6 months	20 years	5 years

Table 4: Implemented maintenance strategies



(a) Maintenance Costs.



(b) Expected number of failures.

Figure 11: Comparison between different number of maintenance strategies for an HVAC systems.

5 CONCLUSION AND FUTURE WORKS

The paper has presented a methodology for applying probabilistic model checking to FMTs. The FMTs are modelled in the form of CTMCs which simplifies the transformation of FMT into formal models that can be analysed using PRISM. A novel technique for abstracting the equivalent CTMC model is also presented. 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. The framework has been applied to an HVAC system and 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, k/N gates and to incorporate lumping of states as in [16], such that the state space can be further reduced.

ACKNOWLEDGMENTS

This work has been funded by the AMBI project under Grant No.: 324432, by the Alan Turing Institute, UK, post-doctoral research grant from Fonds de Recherche du Quebec - Nature et Technologies (FRQNT) and Malta's ENDEAVOUR Scholarships Scheme.

REFERENCES

[1] Vladimir Babishin and Sharareh Taghipour. 2016. Optimal maintenance policy for multicomponent systems with periodic and opportunistic inspections and preventive replacements. *Applied Mathematical Modelling* 40, 24 (2016), 10480–10505.

[2] Francesca Boem, Riccardo MG 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.

[3] Luca Bortolussi and Jane Hillston. 2012. Fluid approximation of CTMC with deterministic delays. In *Quantitative Evaluation of Systems (QEST), 2012 Ninth International Conference on*. IEEE, 53–62.

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

[5] European Parliament and Council of the European Union. 2010. Directive 2010/31/EU. (2010).

[6] ASHRAE Handbook. 1996. HVAC systems and equipment. *American Society of Heating, Refrigerating, and Air Conditioning Engineers, Atlanta, GA* (1996).

[7] Holger Hermanns and Lijun Zhang. 2011. From Concurrency Models to Numbers. In *Nato Science for Peace and Security Series*. IOS Press.

[8] 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 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 1635–1640.

[9] 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 *Formal Methods and Models for Codesign (MEMOCODE), 2014 Twelfth ACM/IEEE International Conference on*. IEEE, 175–184.

[10] Faisal I Khan and Mahmoud M Haddara. 2003. Risk-based maintenance (RBM): a quantitative approach for maintenance/inspection scheduling and planning. *Journal of Loss Prevention in the Process Industries* 16, 6 (2003), 561–573.

[11] 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.

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

[13] ZF Li, Yi Ren, LL Liu, and ZL Wang. 2015. Parallel algorithm for finding modules of large-scale coherent fault trees. *Microelectronics Reliability* 55, 10 (2015), 1400–1403. Proceedings of the 26th European Symposium on Reliability of Electron Devices, Failure Physics and AnalysisSI:Proceedings of {ESREF} 2015.

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

[15] Ying Yan, Peter B Luh, and Krishna R Pattipati. 2017. Fault Diagnosis of HVAC Air-Handling Systems Considering Fault Propagation Impacts Among Components. *IEEE Transactions on Automation Science and Engineering* 14, 2 (April 2017), 705–717.

[16] Olexandr Yevkin. 2015. An efficient approximate Markov chain method in dynamic fault tree analysis. *Quality and Reliability Engineering International* (2015).

[17] Håkan LS Younes, Marta Kwiatkowska, Gethin Norman, and David Parker. 2006. Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer* 8, 3 (2006), 216–228.