Automatic Verification of Stochastic Processes: Certification of Building Automation Systems

Nathalie Margaret Cauchi

Kellogg College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy
Computer Science

Trinity 2019

Abstract

Smart buildings are key to reducing greenhouse gas emissions in the face of the continuous and fast-paced growth of urbanisation. The performance criteria for the optimal operation of such intelligent systems require the coupling of certification goals with the design of both the modelling framework and control algorithms.

In the first part of this thesis we demonstrate the use of Fault Maintenance Trees (FMTs): a qualitative description that embeds component degradation, maintenance policies and the relationship between components leading to a failure. We provide semantics to FMTs using continuous-time Markov chains and employ efficient probabilistic model checking to analyse the performance of heating, ventilation and air-conditioning unit, against certification metrics, under different maintenance schemes. The method is also benchmarked against the application of statistical model checking for analysis.

The second part introduces the modelling framework of discrete-time Stochastic Hybrid Systems (SHS): probabilistic models suitable for describing the dynamics of variables presenting interleaved and interacting continuous and discrete components. We present algorithms and techniques for performing scalable verification and control synthesis of SHS. The kernel of the methods is a novel abstraction procedure that makes use of interval Markov decision processes. The method embeds the exact abstraction error within the abstraction itself. Consequently reducing the number of states required significantly and allowing us to analyse SHS with more than 10 continuous variables. Furthermore, the applicability of the framework is shown via the construction and analysis of a library of models for building automation systems with different certification goals.

Finally, we embed the models and algorithms within StocHy: a new tool aimed at simplifying the analysis of SHS. The tool is written in C++ and allows for simulation, verification and synthesis of stochastic and hybrid systems, in a manner that is accessible to general end-users.

Acknowledgements

This thesis is the culmination of my four-year DPhil journey as part of the Oxford Control and Verification (OxCAV) group, within the Department of Computer Science. It has been greatly shaped and influenced by the people surrounding me. In the following lines, I will do my best to acknowledge their much-appreciated guidance, support and friendship.

First, I would like to express my deep gratitude to Alessandro, for his dedication, support and guidance. I would like to thank him for being a source of inspiration and motivation. Thank you for the many thoughtful exchanges of ideas, opportunities, trust, advice and friendship along the way.

Second, I would like to also express my gratitude to my assessors Marta Kwiatkowska and Alex Rogers, for their fruitful guidance and support. Thanks also to my college advisor, Jeremy Gibbons for the delightful lunch conversations during my stay here in Oxford. I would also like to thank my external examiner Ernst Moritz Hahn for his constructive and thorough feedback.

I am appreciative for the great research and personally enriching experience at the research lab in Honeywell, Prague. I would like to specifically thank Petr Stulka, Ondrej Holub, Petr Endl, Karel Macek, Riccardo Ferrari, Simone Baldi and Raman Samusevich for their hospitality, work ethic and friendship. I am also appreciative of the support and help given, concerning the Smart Buildings laboratory, by the Building Estate services at Oxford.

I would like to thank Sadegh Soudjani, Sofie Haesaert and Kendra Lesser for dispensing their attention and their knowledge of formal abstractions and policy synthesis onto me. I am thankful for their friendship and guidance. I would also like to thank Sofie, for the many hours spent sharing our thoughts. They will always be cherished.

For the work related to fault maintenance trees (FMT), I would like to thank my collaborators. Namely, I would like to thank Mariëlle Stoelinga for introducing me to the FMT framework and your constant support; Khaza Anuarul Hoque for your support and feedback; Carlos Budde for the sharing of ideas, working together and the interesting confabulations.

Luca Laurenti and Morteza Lahijanian have both contributed to this project in first person, dispensing their attention and knowledge of interval Markov decision processes on me. I greatly appreciate the discussions throughout our collaborations and have greatly benefited from your technical rigour and effective feedback.

I would like to thank all the participants of the Stochastic Modelling group within the ARCH competition and to the organisers, Goran Frehse and Matthias Althoff. Thanks for believing in our goal and working together to push forward more tools for analysing stochastic processes.

The Oxford Women in Computer Science (OxWoCS) group has also played a significant role in my journey. My experience as a seminar series coordinator was enriching and participation in the workshops and conferences strengthened my love for my research. I would like to thank all the members of the OxWoCS committee over the last four years, especially Aurore Lyon for her friendship and many wonderful discussions.

I am grateful to the people at Oxford that I have had the privilege to spend my best time with, and which greatly influenced my growth during my stay. In particular, a heartfelt thank you to the people of the OxCAV group (past and present) and foremost, to Andrea Peruffo, Viraj Brian Wijesuriya, Gareth Molyneux, Hosein Hasanbeig, Yuriy Zacchia Lun, Mehran Hosseini, Ada Alevizaki, Elizabeth Polgreen, Muhammad Syifa'ul Mufid and Dario Cattaruzza.

I would like to thank all my friends back in Malta, who even though far away, I know their support is there. Specifically, I would like to thank Adrian Grima who was on a similar journey to my own and with whom I have shared many qualms.

The concluding acknowledgement goes to my family and to Kurt. No words can express the immense gratitude for your never-ending love and support. You have been my strength and motivation.

Grazzi mill-qalb lil kull min kien parti ta' dawn l-aħħar erba' snin.

Contents

List of Figures

List of Tables

List of Abbreviations

ARCH . . . . . Applied Verification for Continuous and Hybrid Systems

AHU . . . . . Air Handling Unit

BAS . . . . . Building Automation System

BE . . . . . Basic Event

BLTL . . . . . Bounded-time Linear Temporal Logic

CSL . . . . . Continuous Stochastic Logic

CSLTL . . . . . Co-Safe Linear Temporal Logic

CTMC . . . . . Continuous Time Markov Chain

DAG . . . . . Direct Acyclic Graph

DFA . . . . . Deterministic Finite Automata

DFT . . . . . Dynamic Fault Tree

DTMC . . . . . Discrete Time Markov Chain

EBE . . . . . Extended Basic Event

ENF . . . . . Expected Number of Failures

FMT . . . . . Fault Maintenance Tree

GD . . . . . Gradient Descent

HVAC . . . . . Heating, Ventilation and Air-Conditioning

IE . . . . . Intermediate Event

iid . . . . . independent identically distributed

IM . . . . . Inspection Module

IMDP . . . . . Interval Markov Decision Process

KKT . . . . . Karush-Kuhn-Tucker

KPI . . . . . Key Performance Indicators

MDP . . . . . Markov Decision Process

MTTF . . . . . Mean Time To Failure

PHA . . . . . Probabilistic Hybrid Automata

PI . . . . . Performance Indices

PMC . . . . . Probabilistic Model Checking

PTA . . . . . Price Timed Automata

RC . . . . . Resistance Capacitance

RM . . . . . Repair Module

SHS . . . . . Stochastic Hybrid Systems

SMC . . . . . Statistical Model Checking

TLE . . . . . Top Level Event

1 Introduction

Contents

The growing ubiquitousness of the internet-of-things has enabled a new type of buildings, termed Smart Buildings, which aim to deliver useful building services that are cost effective, reliable and ensure occupant comfort and productivity (thermal quality, air comfort). Smart buildings combine complex integration of software with multi-physical subsystems, 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.

The overall operation of a smart building is an interconnected web between different subsystems stemming over different domains. These include the building automation system (BAS) which incorporates the communication protocols and control algorithms; the civil engineering building infrastructure; and the thermal, electrical and mechanical components. Unfortunately, different components are typically designed separately and are not synchronised together, adding to the total complexity of buildings [113]. This is further reflected within the research community

where specific goals for sub-components are typically tackled: a multitude of present-day works focus on detecting occupancy in rooms (e.g. [103], [115], [167]), synthesising optimal control policies (e.g. [12], [51], [63], [127]), modelling the internal zone temperature and air quality (e.g. [7], [77], [163]), designing communication protocols connecting smart buildings with the power grid (e.g. [89], [169]).

The operational and maintenance costs of smart buildings, together with the need to reduce the total energy consumption of buildings (buildings as of 2018 consume more than 36% of the global final energy usage [85]), places increasing importance on the need for them to function correctly and efficiently. In order to reach this goal, models and control strategies for buildings need to be (i) standardised following protocols and regulations; and (ii) targeted towards handling the whole building structure. Protocol and regulations come in the form of a set of rules and specifications defined by international agencies. Within the building domain, two prominent certification standards have been documented, namely: LEED [117] targets the generation of net-zero energy consumption and ENERGYSTAR [149] which defines a standard measure of performance compliance for residential buildings.

Real-world requirements and definitions have been shown to contain ambiguous wordings and are under specified [27]. *Formal methods* try to alleviate this situation by requiring specifications to be formulated using logical and mathematical constructs that rid ambiguity. Using formal methods, the behaviour of the system in question, described using rigorous mathematical models is checked against the specifications via formal proofs. Outside the building domain, many safety and embedded systems standards, such as IEC 61499 [164], DO-331 [135] and ISO 26262 [126] strongly recommend the application of formal methods in varying degrees.

Automated and *quantitative verification* form the underlying principles of formal methods. In automated verification *exhaustive* traversal of the underlying model is performed to establish if certain properties hold for a system model. An example of such a property is “does the system remain switched on for the next ten minutes?" and are typically defined using temporal logic. The process of automatically

verifying the correctness of properties of finite-state systems is known as *model checking*. Analogously, quantitative verification extends automated verification, to allow reasoning over quantitative properties such as "what is the probability of the temperature exceeding 25 $°C$?". Quantitative verification is applied in a multitude of domains: aerospace and avionics [128], [148], energy systems [60], [116], and systems biology [46], [99].

Automated and quantitative verification: A technique for establishing if certain properties, typically expressed in temporal logic hold for a system model. This is carried out via a combination of a traversal of the state-transition graph of the model and numerical computation [97].

It comes in two flavours [97]:

- *Probabilistic Model Checking* (PMC, [101]) deals with systems that exhibit probabilistic behaviour and is based on the construction and analysis of a probabilistic model of the system. The system is usually specified as a state transition model and a *probabilistic model checker* calculates the probabilities of reaching different states in a model or computes the expected reward given a time horizon.

- *Statistical Model Checking* (SMC, [168]) generates sample executions of a probabilistic system according to the distribution defined by the system and a *statistical model checker* computes statistical guarantees based on the executions [106].

PMC provides formal guarantees with higher accuracy when compared with SMC [168], at a cost of being more memory intensive and may result in a state-explosion.

More recently within the domain of Control Engineering, quantitative verification has induced a new branch of *correct-by-construction* policy (strategy) synthesis. This encodes performance properties into a logical formula and then makes use of techniques from quantitative verification to determine the optimal control actions

needed to ensure a certain performance level is achieved. A common correct-by-construction approach is based on the computation of a finite-state abstraction of the control system. Given a specification expressed in a formal language such as temporal logic, a controller that enforces this specification on the abstraction is first synthesised and then refined to a controller enforcing the same specification on the original system [156]. This technique yields promising results in the domains, but not limited to, of robotics for the planning of robot motion in a continuous space (e.g. [26], [57]) and of cyber-physical systems for specifying complex tasks over interleaved discrete and continuous variables (e.g. [72], [137]).

Correct-by-construction: A method where control software is synthesised along proof of correctness [156].

This thesis provides the underpinning framework that connects formal methods with the domain of Smart buildings. It sets up a modelling framework for BAS over which certificates can be generated using techniques stemming from quantitative formal verification and control policy synthesis. In Section 1.1, we first identify the key research problems being addressed in this thesis. This is followed by a short overview of the thesis structure together with the novel contributions within each chapter (see Section 1.2). Finally, we list the associated publications in Section 1.3.

1.1 Research questions

From the many open questions related to both the field of BAS and formal methods, we focus on the intersection between constructing models with sufficient detail and reasoning about a system's performance. Determining the level of detail captured by a model to represent a system, is an arduous task and a trade-off between complexity, size and true representation of the dynamics of a system, is a must. A good middle ground is the inclusion of uncertainty within the modelling framework which takes into account unmodelled effects. In BAS, this is highly relevant due to the many sources of uncertainty stemming from, for instance, the

weather, occupancy patterns, or from the sensor measurements themselves [113]. We can capture this uncertainty by incorporating probability within the modelling structure. Furthermore, BAS operates within an embedded system where discrete events directly affect the continuous variables representing the building's internal dynamics. Hybrid models allow us to capture the relationship between the discrete and continuous variables. Analysis of models with both probabilistic and hybrid elements is hard and presents multiple interesting challenges. On this premise, we raise the following questions:

- How do we construct a general modelling framework which: (i) captures uncertainty and the hybrid elements in BAS to a sufficient degree; (ii) can be used for sets of similar buildings and applications and (iii) is both theoretically sound and practical to use?

- How do we couple certification goals with the modelling framework, such that we can: (i) model check the performance of the system against a certain requirement; and (ii) embed the requirement within both the modelling phase and the control strategy design?

We propose the use of stochastic hybrid systems (SHS) as the modelling framework which can compound both uncertainty and the relationship between discrete and continuous variables in the model. The coupling of certification goals with SHS can be achieved by applying formal methods. However, this introduces a number of quandaries:

- general verification goals are bound to be undecidable, and their approximated (and finite) versions are frequently stymied by the state-space explosion problem. This restricts the level of detail and complexity attainable for the verification of stochastic models.

- similar to the previously raised point, the synthesis of control policies on such models is also hindered by the curse of dimensionality. To this end, different techniques aimed at speeding up formal algorithms through abstractions [147],

or by data-driven approaches, such as reinforcement learning [33], [80] have been developed. However, they all require specific types of models and work under their own set of assumptions.

1.2 Overview of the thesis

This thesis sets up a formal modelling, verification and synthesis framework for BAS that is easily extensible and can be adopted by non-technical users. As a deliverable, a software tool called StocHy, which encapsulates the solutions to our main goals, is also presented. This aims to address the research questions identified in Section 1.1. Figure 1.1 depicts a pictorial representation of the relationship between the different chapters within this thesis.

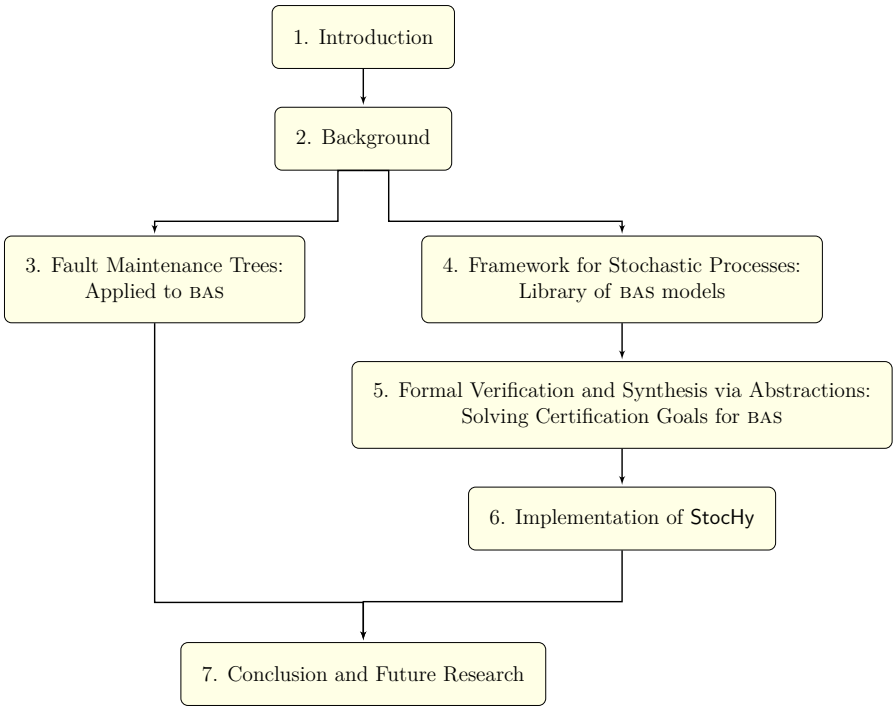


Figure 1.1: Overview of thesis structure.

A breakdown of the structure of this manuscript is as follows:

- **Chapter 2** formally defines the term certification and the different levels of such certification: high-level and low-level. This is followed by definition of the general technical preliminaries: general notation, the different probabilistic models and associated logics.

- **Chapter 3** introduces fault maintenance trees (FMT) and discusses a novel framework to analyse FMTs using PMC. We apply the framework to a BAS case-study and presents a comparison between analysing FMTs using the PMC framework and the current state of the art which employ SMC.

- **Chapter 4** formally defines the term stochastic processes and in particular SHS. We describe how BAS models fit within this modelling framework and present a library of models for BAS. For these models, we explain identify verification and synthesis goals.

- **Chapter 5** builds upon Chapter 4 and sets up a framework for performing verification and synthesis. We introduce two different formal abstraction techniques: (i) for uncountable continuous state spaces we abstract models into discrete-time Markov processes (MDP); and (ii) for continuous state spaces with countable control actions we abstract models into interval Markov decision process (IMDP). More specifically, we present extensions to the policy synthesis frameworks via a MDP, and methods for performing verification and synthesis using IMDP. Finally, we solve the high-level certification goals of the BAS case studies delineated in Chapter 4.

- **Chapter 6** connects Chapter 4 and 5 by integrating the work into a tool called StocHy, which aims to make the analysis of models endowed with stochastic elements easier for general end-users. This chapter describes the implementation and serves as a user guide to StocHy.

- **Chapter 7** concludes the work by recapitulating the main contributions of this manuscript and outlines directions for future research.

All the methods and algorithms developed are based on data obtained from the Smart Buildings Laboratory within the Department of Computer Science, at the University of Oxford. The Smart Building Laboratory consists of two highly sensorised teaching rooms (see Figure 1.2) which are used daily for the MSc Degree in Software Engineering offered by the Department.

Figure 1.2: Smart Building laboratory within the Department of Computer Science, Oxford (Room 478 - 479).

The teaching rooms are connected back to back and have three of the walls of the two rooms exposed to the outside, while the rest connects to the interior hall. The dimensions of the floor plan areas, for each respective room, are $94m^2$ and $96m^2$. Both rooms have two external windows and form part of a larger building management system that controls the rest of the building. Indoor heating control is primarily managed with thermostat controlled radiators and air handling units, which form part of a variable-air ventilation system. The rooms are further equipped with temperature and CO_2 sensors. Measurements consisting of five-minute sampled values are recorded and stored on a daily basis.

Using the sensor measurements for temperature, CO_2 levels, radiator valve positions and boiler status (ON or OFF) over the period of one year, we constructed the library of model for BAS that is formally defined in Chapter 4. All the models have been further validated against and tested on an internal simulator provided by Honeywell Labs, Prague. Further note that the controllers built using these models, which are described within Chapter 5 have not been implemented as part of a real test-bed due to safety and timing constraints.

1.3 Publications by the author

The material presented in this doctoral thesis has appeared in top-tier international conference proceedings, or has been published in peer-reviewed journals. The

connection between each chapter and the publications is as follows:

- Chapter 3
  - N. Cauchi, K. A. Hoque, A. Abate, and M. Stoelinga, "Efficient Probabilistic Model Checking of Smart Building Maintenance using Fault Maintenance Trees", in *Proceedings of the fourth ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys)*, ACM, 2017, p. 24
  - A. Abate, C. E. Budde, N. Cauchi, A. van Harmelen, K. A. Hoque, and M. Stoelinga, "Modelling Smart Buildings Using Fault Maintenance Trees", in *European Workshop on Performance Engineering (EPEW)*, Springer, 2018, pp. 110–125
  - A. Abate, C. E. Budde, N. Cauchi, K. A. Hoque, and M. Stoelinga, "Assessment of Maintenance Policies for Smart Buildings: Application of Formal Methods to Fault Maintenance Trees", in *Proceedings of the European Conference of the PHM Society*, PHM society, vol. 4, 2018
  - N. Cauchi, K. A. Hoque, M. Stoelinga, and A. Abate, "Maintenance of Smart Buildings Using Fault Trees", *ACM Transactions on Sensor Networks (TOSN)*, vol. 14, no. 3-4, 28:1–28:25, 2018, ISSN: 1550-4859
- Chapter 4 and 5
  - S. Haesaert, N. Cauchi, and A. Abate, "Certified Policy Synthesis for General Markov Decision Processes: An Application in Building Automation Systems", *Performance Evaluation*, vol. 117, pp. 75–103, 2017
  - N. Cauchi and A. Abate, "Benchmarks for Cyber-Physical Systems: A Modular Model Library for Building Automation Systems", in *Proceedings of the Sixth IFAC Conference on Analysis and Design of Hybrid Systems (ADHS)*, vol. 51, 2018, pp. 49–54

  - A. Abate, H. Blom, N. Cauchi, S. Haesaert, A. Hartmanns, K. Lesser, M. Oishi, V. Sivaramakrishnan, S. Soudjani, C.-I. Vasile, *et al.*, "ARCH-COMP18 Category Report: Stochastic Modelling", in *ARCH18. Fifth International Workshop on Applied Verification of Continuous and Hybrid Systems, ARCH@ ADHS 2018, Oxford, UK, July 13, 2018*, 2018, pp. 71–103
  - N. Cauchi, L. Laurenti, M. Lahijanian, A. Abate, M. Kwiatkowska, and L. Cardelli, "Efficiency Through Uncertainty: Scalable Formal Synthesis for Stochastic Hybrid Systems", in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, New York, NY, USA: ACM, 2019, pp. 240–251, ISBN: 978-1-4503-6282-5
  - A. Abate, H. Blom, N. Cauchi, K. Degiorgio, M. Franzle, E. M. Hahn, S. Haesaert, H. Ma, M. Oishi, C. Pilch, A. Remke, M. Salamati, S. Soudjani, B. van Huijgevoort, and A. P. Vinod, "ARCH-COMP19 Category Report: Stochastic Modelling", *EPiC Series in Computing*, vol. 61, G. Frehse, Ed., pp. 62–102, 2019

- Chapter 6
  - N. Cauchi and A. Abate, "StocHy: Automated Verification and Synthesis of Stochastic Processes", in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, T. Vojnar and L. Zhang, Eds., Cham: Springer International Publishing, 2019, pp. 247–264, ISBN: 978-3-030-17465-1

The following publications are related to the material in this thesis, but are not included within this manuscript. In these publications, we devise a framework for performing predictive maintenance for BAS using deterministic models. This is outside the main focus of the manuscript, that of employing and reasoning over models which are uncertain and thus, nondeterministic for BAS.

- K. Macek, P. Endel, N. Cauchi, and A. Abate, "Long-Term Predictive Maintenance: A Study of Optimal Cleaning of Biomass Boilers", *Energy and Buildings*, vol. 150, pp. 111–117, 2017
- N. Cauchi, K. Macek, and A. Abate, "Model-Based Predictive Maintenance in Building Automation Systems with User Discomfort", *Energy*, vol. 138, pp. 306–315, 2017

2 Background

Contents

In this chapter, we define the term *certification* and how it is applied within the context of this thesis. This is followed by establishing the formal underpinnings of the remaining chapters. We start by introducing notation together with well-known concepts to reason about probabilistic systems. Next, we formally define and explain the behaviours of the probabilistic models used in this thesis. Finally, we discuss the syntax and semantics of the associated temporal logics.

2.1 Certification

BAS consists of multiple physical and engineered components that are controlled and monitored using computing algorithms written in discrete logic. The continuous dynamics are further affected by noise stemming from the physical environment. To reason about such complex systems, *models* need to be constructed.

Model: A representation or abstraction of something such as an entity, a system or an idea. [19]

A model must be able to realistically represent the whole operation of the BAS together with that of the individual subcomponents. Based on such models, engineers construct algorithms that control its performance.

The International Organisation for Standardisation defines a set of rules which govern the operation of systems via the principle of certification.

Certification: A "procedure by which a third party gives written assurance that a product, process or service conforms to specified characteristics." [133]

In industry, there are different certification standards (LEED [94], BREEAM [94], ENERGY STAR [149]) and design handbooks (e.g. ASHRAE [11]) which BAS adhere to. These define levels of operation for BAS in terms of their ability to provide thermal comfort, good air quality, reliable operation and high energy efficiency usage. In order to achieve the certification levels required by these standards current trial-and-error approaches to build computing-centric engineered systems must be replaced by rigorous formal methods and robust system design [134]. Formal methods, such as quantitative verification, provide a mathematical framework to prove correctness of systems components and have become an essential component during the design and development phase of systems in industry [124]. This has not escaped the smart energy domain [8], [150], [153]. In this work, we apply formal methods to generate certificates for different levels of model abstractions. We split the term certification into two:

- *high-level* - reasoning over the BAS as a whole;
- *low-level* - reasoning over individual subcomponents.

In *high-level* certification, we reason about the overall performance of a system. The *performance* criterion is based on certification requirements which can be

formalised as *Key Performance Indicators* (KPI), i.e. a set of metrics which are specified over a time horizon $K > 0$, and quantify failures in the time window $[0, K]$ [90]. Some of the most relevant KPIs are:

- *Reliability* - the probability of the system not failing in the time window $[0, K]$;
- *Availability* - the expected fraction of time in the window $[0, K]$ that the system is operational;
- *Expected number of failures* (ENF) - the expected number of times a failure is observed in the time window $[0, K]$;
- *Expected costs* - the total costs incurred in the time window $[0, K]$, including operational, inspection, maintenance costs, and costs associated to system failures.

In contrast, for achieving *low-level* certification we reason about the behaviour of subsystems within the BAS. At this level, we focus on KPIs defined in terms of

- *Safety* - the probability of the system remaining within an operational region, in the time window $[0, K]$;
- *Reach* - the probability of the system attaining an operational region, in the time window $[0, K]$;
- *Reach and Avoid* - the probability that executions of a system attain an operational region while avoiding another (or conversely never leaving the complementary safe region), during the time horizon $[0, K]$.

2.2 Technical preliminaries

2.2.1 General notation

We denote the set of real numbers by $\mathbb{R}$ and the natural numbers (including 0) by $\mathbb{N}$. Given a set of numbers A, we use subscripts to indicate the restriction to the

non-negative and positive fragments of A, respectively. For example, $\mathbb{R}_{\geq 0}$ refers to the non-negative reals.

We use $[a, b] = \{c \in \mathbb{R} \mid a \leq c \leq b\} \subseteq \mathbb{R}$ to denote the interval of reals between a and b including the boundaries. We denote the empty set as $\emptyset$. Given two homogenous sets A, B, their difference is defined as $A \backslash B = \{x : x \in A \wedge x \notin B\}$. We represent their Cartesian product by $A \times B = \{(a, b) \mid a \in A \wedge b \in B\}$.

A probability measure P for a sample space $\mathbb{X}$ and σ-algebra C defined over $\mathbb{X}$ is a non-negative map, $\mathrm{P} : C \to [0, 1]$ such that $\mathrm{P}(\mathbb{X}) = 1$ and such that for all countable collections of $\{Z_i\}_{i=1}^{\infty}$ of pairwise disjoint sets in C, it holds that $\mathrm{P}(\cup_i Z_i) = \sum_i \mathrm{P}(Z_i)$. The triple $(\mathbb{X}, C, \mathrm{P})$ define the probability space and has realisations $x \sim \mathrm{P}$. Further note, a special instance of C is the Borel σ-algebra. We denote the Borel σ-algebra on the set S as $\mathcal{B}(S)$ and the Borel measurable space is $(S, \mathcal{B}(S))$.

We define a $\mathcal{N}(\mu, \sigma)$ as a normal distribution with mean μ and variance σ. If μ and σ are an n-dimensional vector and an $x \times n$ symmetric, positive semi definite matrix, respectively, $\mathcal{N}(\mu, \sigma)$ is a multi-dimensional distribution.

We highlight general concepts using yellow boxes and new concepts using boxes filled in light blue.

Polytopes and their post images

Let $m \in \mathbb{N}$ and consider the m-dimensional Euclidean space $\mathbb{R}^m$. A full dimensional (convex) *polytope* $\mathbf{P}$ is defined as the convex hull of at least $m+1$ affinally independent points in $\mathbb{R}^m$ [66]. The *set of vertices* of $\mathbf{P}$ is the set of points $\{v_1^{\mathbf{P}}, \ldots, v_{n_{\mathbf{P}}}^{\mathbf{P}} \in \mathbb{R}^m, n_{\mathbf{P}} \geq m + 1\}$, whose convex hull gives $\mathbf{P}$ and with the property that, for any $i = \{1, \ldots, n_{\mathbf{P}}\}$, point $v_i^{\mathbf{P}}$ is not in the convex hull of the remaining points $\{v_1^{\mathbf{P}}, \ldots, v_{i-1}^{\mathbf{P}} v_{i+1}^{\mathbf{P}}, \ldots, v_{n_{\mathbf{P}}}^{\mathbf{P}}\}$. A polytope is completely described by its set of vertices

$$\mathbf{P} = conv(v_1^{\mathbf{P}}, \ldots, v_{n_{\mathbf{P}}}^{\mathbf{P}}), \tag{2.1}$$

where *conv* denotes the convex hull. Alternatively, $\mathbf{P}$ can be described as the bounded intersection of at least $m + 1$ closed half spaces. In other words, there

exists a $k \geq m+1$, $h_i \in \mathbb{R}^m$, and $l_i \in \mathbb{R}$, $i = \{1, \ldots, k\}$ such that

$$\mathbf{P} = \{x \in \mathbb{R}^m \mid h_i^T x \leq l_i,\, i = \{1, \ldots, k\}\}. \tag{2.2}$$

The above definition can be written as the matrix inequality $Hx \leq L$, where $H \in \mathbb{R}^{k \times m}$ and $L \in \mathbb{R}^k$. Given a matrix $\mathcal{T} \in \mathbb{R}^{m \times m}$, the post image of polytope $\mathbf{P}$ by $\mathcal{T}$ is defined as [102]

$$\mathit{Post}(\mathbf{P}, \mathcal{T}) = \{\mathcal{T}x \mid x \in \mathbf{P}\}. \tag{2.3}$$

This post image is a polytope itself under the linear transformation $\mathcal{T}$ and can be computed as

$$\mathit{Post}(\mathbf{P}, \mathcal{T}) = \mathit{conv}\Big(\{\mathcal{T}v_i^{\mathbf{P}} \mid 1 \leq i \leq n_{\mathbf{P}}\}\Big). \tag{2.4}$$

2.2.2 Probabilistic models

To reason over complex systems an abstraction of the system that would yield a simpler model, over which analysis and computation can be performed, is required. There are different abstraction models that can be used. In this subsection, we classify the abstraction models according to the level of complexity the abstraction captures. The different model types can be categorised along three dimensions. The first one is whether the formalism has a discrete or continuous notion of time. The second dimension is the absence or inclusion of nondeterministic choice (also known as inputs or actions). Some models allow the environment or a controller to make choices that govern their behaviour, while others behave fully probabilistically. The third dimension is whether transition probabilities are defined using fixed probability values or described using intervals. We first consider the time dimension and present *continuous-time Markov chain* (CTMC) followed by the *discrete-time Markov chain* (DTMC). We then extend into the remaining dimensions where the other models - *Markov decision processes* (MDP) and *interval Markov decision processes* (IMDP) - can be constructed by extensions of the DTMC.

Continuous-time Markov Chain (CTMC): A tuple defined by

$$\mathcal{C} = (\mathcal{S}, \Upsilon, \mathcal{R}, \bar{\Theta}, \mathcal{L}) \quad \textit{where}, \tag{2.5}$$

- $\mathcal{S} = \{s_0, s_1, \ldots, s_m\}$, $m \in \mathbb{N}$ is a finite set of states;
- Υ is a finite set of transition labels;
- $\mathcal{R} : \mathcal{S} \times \Upsilon \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the transition rate matrix;
- $\bar{\Theta}$ are a finite set of atomic propositions (also referred as state labels);
- $\mathcal{L} : \mathcal{S} \to 2^{\bar{\Theta}}$ is a labelling function.

The rate $\mathcal{R}(s_i, \upsilon_i, s_{i+1})$ defines the time delay before which a transition between states s_i and s_{i+1} under the transition labelled with $\upsilon_i \in \Upsilon$, $i \in \mathbb{N}$ takes place. If $\mathcal{R}(s_i, \upsilon_i, s_{i+1}) \neq 0$ then the probability that a transition between the states s_i and s_{i+1} is defined as $1 - e^{-\mathcal{R}(s_i, \upsilon_i, s_{i+1})t}$, where t is time. No transitions can trigger if $\mathcal{R}(s_i, \upsilon_i, s_{i+1}) = 0$.

Figure 2.1 presents a CTMC described using the tuple $\mathcal{C} = (\mathcal{S}, \Upsilon, \mathcal{R}, \bar{\Theta}, L)$ where the set of states are $\mathcal{S} = \{s_0, s_1\}$ with the transition rate matrix

$$\mathcal{R} = \begin{bmatrix} 3.5 & 0 \\ 0 & 1.5 \\ 0.85 & 0 \\ 0 & 0.15 \end{bmatrix}.$$

The transition labels correspond to $\Upsilon = \{$`do_maintenance, check_maintenance, reset, wait`$\}$, the atomic propositions are $\bar{\Theta} = \{l, b\}$, and the labelling function L assigns the atomic propositions to each of the states in the CTMC such that $\mathcal{L}(s_0) = l$, $\mathcal{L}(s_1) = b$.

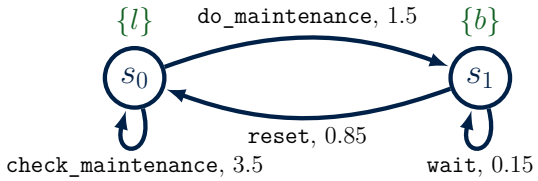


Figure 2.1: An example of a CTMC.

The unrolling of the CTMC is described using paths and are defined by a finite or infinite sequence of states, transition labels, and time $\mu = s_0 \upsilon_0 t_0 s_1 \upsilon_1 t_1 \ldots$, for

$i \in \mathbb{N}$, $s_i \in \mathcal{S}$, $\upsilon_i \in \Upsilon$ and $t_i \in \mathbb{R}_{>0}$ such that $\mathcal{R}(s_i, \upsilon_i, s_{i+1}) > 0$, if μ is an infinite path. We denote the set of all paths generated by the CTMC as Path(s).

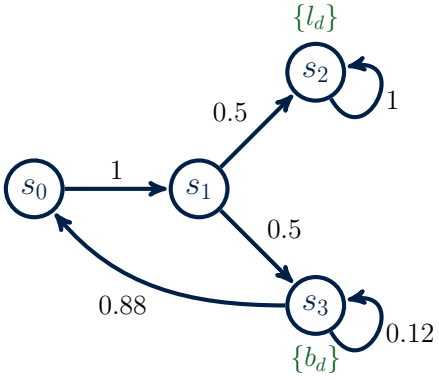
If we reason over the discrete time domain, the most basic probabilistic model is the DTMC. This has the same flavours as a CTMC, however rather than reasoning using transition rates, a DTMC simply defines the probability of making a transition from one state to another [17]. The transition labels are also neglected.

Discrete-time Markov Chain (DTMC): A tuple defined by

$$\mathcal{M}_{\text{DTMC}} = (\mathcal{S}, T_s, \bar{\Theta}, \mathcal{L}) \quad \textit{where} \tag{2.6}$$

- $\mathcal{S} = \{s_0, s_1, \ldots, s_m\}$, $m \in \mathbb{N}$, is a finite set of states;
- $T_s : \mathcal{S} \times \mathcal{S} \to [0, 1]$ is a discrete stochastic kernel that assigns, to each $s \in \mathcal{S}$, a probability distribution over $\mathcal{S} : T_s(\cdot|s)$;
- $\bar{\Theta}$ are a finite set of atomic propositions;
- $\mathcal{L} : \mathcal{S} \to 2^{\bar{\Theta}}$ is a labelling function.

Figure 2.2 depicts an example of DTMC, consisting of four states, $\mathcal{S} = \{s_0, s_1, s_2, s_3\}$, a transition probability function T_s given by (2.7), atomic propositions $\bar{\Theta} = \{l_d, b_d\}$ and a labelling function $\mathcal{L}$ such that $\mathcal{L}(s_0) = \emptyset$, $\mathcal{L}(s_1) = \emptyset$, $\mathcal{L}(s_2) = l_d$, $\mathcal{L}(s_3) = b_d$.



$$T_s = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 1 & 0 \\ 0.88 & 0 & 0 & 0.12 \end{bmatrix}. \tag{2.7}$$

Figure 2.2: An example of a DTMC.

The evolution of a DTMC can be described using paths defined as an infinite-state sequence $\omega = s_0 s_1 s_2 \cdots \in \mathcal{S}^\omega$ such that $\mathrm{P}(s_i, s_{i+1}) > 0$ for all $i \in \mathbb{N}$. We denote the set of all paths by Path(s).

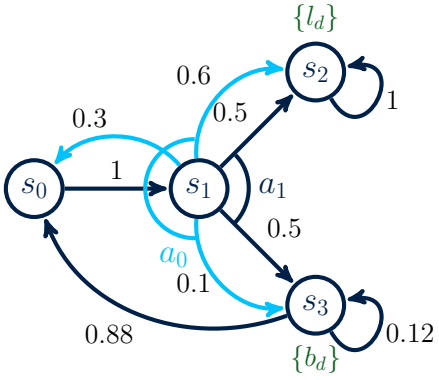
A DTMC can be extended into an MDP, by the inclusion of a finite set of actions (also referred to as non-determinism).

Markov decision process (MDP): A tuple defined by

$$\mathcal{M}_{\text{MDP}} = (\mathcal{S}, \mathcal{A}, T_s, \bar{\Theta}, \mathcal{L}), \quad \textit{where} \tag{2.8}$$

- $\mathcal{S} = \{s_0, s_1, \ldots, s_m\}$, $m \in \mathbb{N}$, is a finite set of states;
- $\mathcal{A} = \{a_0, a_1, \ldots, a_o\}$, $o \in \mathbb{N}$ is a finite set of actions;
- $T_s : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a discrete stochastic kernel that assigns, to each $s \in \mathcal{S}$ and $a \in \mathcal{A}$, a probability distribution over $\mathcal{S} : T_s(\cdot|s, a)$;
- $\bar{\Theta}$ are a finite set of atomic propositions;
- $\mathcal{L} : \mathcal{S} \to 2^{\bar{\Theta}}$ is a labelling function.

An example of a MDP is shown in Figure 2.3. It is composed of four states, $\mathcal{S} = \{s_0, s_1, s_2, s_3\}$, a finite set of actions $\bar{\Theta} = \{a_0, a_1\}$ (further distinguished by the colour coding on the transitions with dark and light blue for each action respectively), the transition probability T_s described using (2.9), the set $\bar{\Theta} = \{l_d, b_d\}$ and a labelling function $\mathcal{L}(s_0) = \emptyset$, $\mathcal{L}(s_1) = \emptyset$, $\mathcal{L}(s_2) = l_d$, $L(s_3) = b_d$.



$$T_s = \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0.3 & 0 & 0.6 & 0.1 \\ 0 & 0 & 0.5 & 0.5 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0.88 & 0 & 0 & 0.12 \\ 0.88 & 0 & 0 & 0.12 \end{array}\right]. \tag{2.9}$$

Figure 2.3: An example of a MDP.

A path ω through an MDP is a sequence of states and actions $\omega = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$ such that $a \in \mathcal{A}(s_i)$ and $P(s_i, a, s_{i+1}) > 0$ for all $i \in \mathbb{N}$. We denote the last state of a finite path ω^{fin} by $last(\omega^{fin})$ and the set of all finite and infinite

paths by $\text{Path}^{fin}(s)$ and $\text{Path}(s)$, respectively. A control strategy (or policy) defines the choice of action at each state of MDP. Its formal definition is given as follows:

Control strategy: A function mapping $\pi : \text{Path}^{fin}(s) \to \mathcal{A}$ that specifies for every finite path, the next action to be applied. If a control strategy depends only on the the last state of ω^{fin}, it is called a stationary (or memoryless) strategy [102].

A MDP can be extended to an uncertain transition matrix T_s, resulting in what are known as *bounded-parameter* [61] or *interval* MDP [145].

Interval Markov decision process (IMDP): A tuple defined by

$$\mathcal{I} = (\mathcal{S}, \mathcal{A}, \check{T}_s, \hat{T}_s, \bar{\Theta}, \mathcal{L}), \quad \textit{where} \tag{2.10}$$

- $\mathcal{S} = \{s_0, s_1, \ldots, s_m\}$, $m \in \mathbb{N}$, is a finite set of states;
- $\mathcal{A} = \{a_0, a_1, \ldots, a_o\}$, $o \in \mathbb{N}$ is a finite set of actions;
- $\check{T}_s : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a function that assigns to each $s \in \mathcal{S}$ a *lower* bound probability distribution over $\mathcal{S} : \check{T}_s(\cdot|s, a)$;
- $\hat{T}_s : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a function that assigns to each $s \in \mathcal{S}$ an *upper* bound probability distribution over $\mathcal{S} : \hat{T}_s(\cdot|s, a)$;
- $\bar{\Theta}$ are a finite set of atomic propositions;
- $\mathcal{L} : \mathcal{S} \to 2^{\bar{\Theta}}$ is a labelling function.

For all $s_i, s_{i+1} \in \mathcal{S}, \forall i \in \mathbb{N}$ and $a \in \mathcal{A}$, it holds that

$$\check{T}_s(s_{i+1}|s_i, a) \leq \hat{T}_s(s_{i+1}|s_i, a)$$

and,

$$\sum_{s_{i+1} \in \mathcal{S}} \check{T}_s(s_{i+1}|s_i, a) \leq 1 \leq \sum_{s_{i+1} \in \mathcal{S}} \hat{T}_s(s_{i+1}|s_i, a).$$

Note that when $\check{T}_s(\cdot|s, a) = \hat{T}_s(\cdot|s, a)$, the IMDP reduces to the MDP with $\check{T}_s(\cdot|s, a) = \hat{T}_s(\cdot|s, a) = T_s(\cdot|s, a)$.

We depict an example of an IMDP using Figure 2.4, which inherits the same structure as $\mathcal{M}_{\text{MDP}}$, with the difference in the transition probability matrix. For the

IMDP, the transition probabilities are characterised using (2.11) representing the minimum and maximum transition probability, respectively.

$$\check{T}_s = \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0.3 & 0 & 0.2 & 0.1 \\ 0 & 0 & 0.4 & 0.5 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0.86 & 0 & 0 & 0.09 \\ 0.86 & 0 & 0 & 0.09 \end{array}\right], \quad \hat{T}_s = \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0.4 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0.5 & 0.6 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0.91 & 0 & 0 & 0.14 \\ 0.91 & 0 & 0 & 0.14 \end{array}\right]. \tag{2.11}$$

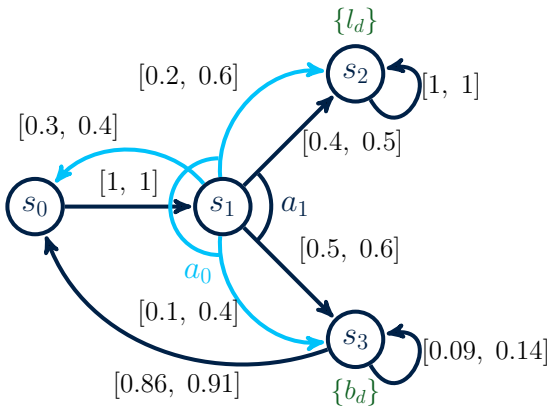


Figure 2.4: An example of an IMDP.

The uncertainty over the transition probabilities requires the introduction of the notion of a *feasible distribution*. Let $\mathcal{P}(\mathcal{S})$ denote the set of discrete probability distributions over $\mathcal{S}$. Given $s_i \in \mathcal{S}$ and $a \in \mathcal{A}(s_i)$, we call $\gamma_{s_i}^a \in \mathcal{P}(\mathcal{S})$ a feasible distribution reachable from s_i by a if

$$\check{T}_s(s, a, s_{i+1}) \leq \gamma_s^a(s_{i+1}) \leq \hat{T}_s(s, a, s_{i+1})$$

for each state $s_{i+1} \in \mathcal{S}$. We denote the set of all feasible distributions for state s and action a by Γ_s^a. The mechanism that selects feasible distributions from interval sets is called an *adversary*. This allows us to analyse IMDPs.

Adversary: Given an IMDP $\mathcal{I}$, an adversary is a function $\kappa : \text{Path}^{\text{fin}}(s) \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$ that, for each finite path $\omega^{\text{fin}} \in \text{Path}^{\text{fin}}(s)$ and action $a \in \mathcal{A}(last(\omega^{\text{fin}}))$, assigns a feasible distribution $\kappa(\omega^{\text{fin}}, a) \in \Gamma^a_{last(\omega^{\text{fin}})}$.

Given a finite path ω^{fin}, a control strategy π, and an adversary κ, the semantics of a path of the IMDP is as follows. At state $s_i = last(\omega^{\text{fin}})$, first an action $a \in \mathcal{A}(s_i)$ is chosen by strategy π. Then, the adversary κ resolves the uncertainties and chooses one feasible distribution $\gamma^a_{s_i} \in \Gamma^a_{s_i}$. Finally, the next state s_{i+1} is chosen according to the distribution $\gamma^a_{s_i}$, and the path ω^{fin} is extended by s_{i+1}. Given a strategy π and an adversary κ, a probability measure P over the set of all finite paths Path(s) (under π and κ) is induced by the resulting Markov chain [102].

2.2.3 Temporal logics

To reason and analyse probabilistic models, appropriate formal logics need to be used. There are different types of formal logics and the selection of which logic to use for analysing a certain probabilistic model is dependent on both the underlying model structure and the requirement at hand. In this work we focus on two different types of logics: *continuous stochastic logic* (CSL) and *co-safe linear temporal logic* (CSLTL). CSL is applied to probabilistic models evolving over the continuous-time domain i.e. CTMCs. On the other hand, we describe CSLTL for probabilistic models evolving over the discrete-time domain.

Continuous stochastic logic

CSL specifies *state*-base properties for probabilistic models evolving along the continuous-time domain such as CTMC. It is built out of propositional logic (with atoms $\theta \in \bar{\Theta}$) and a probabilistic operator P for reasoning about transient state probabilities.

CSL syntax: CSL path and state formulas are given by,

$$\Phi := \texttt{true} \mid \theta \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathrm{P}_{\sim p}[\Phi]$$
$$\Psi := \mathbf{X}\ \Phi \mid \Phi_1\ \mathbf{U}^{\leq T}\ \Phi_2$$

where $\sim \in \{<, \leq, =, \geq, >\}$, $p \in [0,1]$, $T \in \mathbb{R}_{\geq 0}$ is the time horizon, $\wedge$ ("conjunction") is a Boolean operator, P is a Probabilistic operator and **X** ("next"), and **U** ("until") are path operators.

The syntax of CSL is split between state, Φ and path Ψ formula. The state formulas are interpreted over states of a CTMC, whereas the path formulas are interpreted over paths in a CTMC and are only allowed inside the P operator [14], [98]. Given a CTMC, the definition of the satisfaction relation $\models$ over state-formulas is given by induction:

- $s \models \texttt{true}$ for all $s \in \mathcal{S}$;
- $s \models \theta$ iff $\theta \in \mathcal{L}(s)$;
- $s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$;
- $s \models \neg\Phi$ iff $s \not\models \Phi$;
- $s \models \mathrm{P}_{\sim p}[\Phi]$ iff $\mathrm{Pr}_s\{\sigma \in \mathrm{Path}(s) : \mu \models \Phi\} \sim p$;

Recall that μ is a finite or infinite sequence of states and time i.e. $\mu = s_0, t_0, s_1, t_1, \ldots$ $\mathrm{Path}(s)$ corresponds to all the paths that can be generated by the CTMC and Pr_s corresponds to the probability measure over the state-space s. Consequently, the $\mathrm{Pr}_s\{\sigma \in \mathrm{Path}(s) : \mu \models \Phi\}$, informally states the probability of a certain path satisfying Φ. The semantics of the path-formulas are defined as:

- $\sigma \models \mathbf{X}\Phi$ iff $\mu[1] \models \Phi$;
- $\sigma \models \Phi_1\ \mathbf{U}^{\geq T}\ \Phi_2$ iff $\exists k \geq T, \mu[k] \models \Phi_2,$ and $\forall\ 0 \leq i < k, \mu[i] \models \Phi_1$;

Informally, **X** is the next operator and **U** is the until operator. $\mathrm{P}_{\sim p}[\Phi\ \mathbf{U}^{\geq t}\ \Phi]$ 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. States (or particular transitions) can be associated with a real-valued reward, which allows us to compute reward-based queries such as the expected cost of an action in a time window. The extended CSL logic adds reward operators [98]:

CSL reward operators:

$$\mathrm{R}_{\sim r}[\mathbf{C}^{\leq T}] \mid \mathrm{R}_{\sim r}[\mathbf{I}^{=T}] \mid \mathrm{R}_{\sim r}[\mathbf{F}\ \Phi] \mid \mathrm{R}_{\sim r}[\mathbf{S}]$$

where $r, t \in \mathbb{R}_{\leq 0}$ and Φ is a CSL formula.

A state s satisfies $\mathrm{R}_{\sim r}[\mathbf{C}^{\leq T}]$ if, from state s, the expected reward cumulated up until T time units have elapsed satisfies $\sim r$. $\mathrm{R}_{\sim r}[\mathbf{I}^{=T}]$ is true if, from state s, the expected state reward at time instant T meets the bound $\sim r$. $\mathrm{R}_{\sim r}[\mathbf{F}\ \Phi]$ is true if, from state s, the expected reward cumulated before a state satisfying Φ is reached meets the bound $\sim r$[1]. $\mathrm{R}_{\sim r}[\mathbf{S}]$ is true if, from state s, the long-run average expected reward satisfies $\sim r$.

Note, the probabilistic operator $\mathrm{P}_{\sim p}[\Phi]$ and the reward operator $\mathrm{R}_{\sim r}[\mathbf{F}\ \Phi]$ allow for nesting of properties within the CSL logic. In this work, we handle nesting due to use of the PRISM model checker which inherently deals with these complex properties. However, the key performance metrics we are interested in do not require the use of nested properties.

Co-safe linear temporal logic

CSLTL are a set of formal properties based on the linear ordering of time events. CSLTL is defined over path formulas φ which allow the inclusion of multiple temporal operators but cannot specify bounded-time properties [95].

CSLTL syntax: A CSLTL formula φ over a set of atomic propositions $\bar{\Theta}$ is inductively defined as follows:

$$\varphi := \theta \mid \neg\theta \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi\, \mathbf{U}\varphi \mid \mathbf{F}\,\varphi,$$

where $\theta \in \bar{\Theta}$, $\neg$ (negation), $\vee$ (disjunction), and $\wedge$ (conjunction) are Boolean operators, and $\mathbf{X}$ (“next"), $\mathbf{U}$ (“until"), and $\mathbf{F}$ (“eventually") are temporal operators.

[1] If Φ is never reached, the reward would be infinite.

In order to reason over fixed (finite) time horizons, time steps need to be encoded within the path formula φ. This has brought about the existence of a variant the CSLTL logic, known as *bounded-time* linear temporal logic (BLTL) [87].

BLTL syntax: A BLTL formula φ over a set of atomic propositions $\bar{\Theta}$ is inductively defined as following:

$$\varphi := \theta \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi\, \mathbf{U}^{\leq k}\varphi \mid \mathbf{F}^{\leq k}\varphi \mid \mathbf{G}^{\leq k}\varphi,$$

where $\theta \in \bar{\Theta}$ is an atomic proposition, $k \in K$ is the discrete time horizon, $\neg$ (negation) and $\vee$ (disjunction) are Boolean operators, $\mathbf{X}$ ("next"), $\mathbf{U}^{\leq k}$ ("bounded until"), $\mathbf{F}^{\leq k}$ ("bounded eventually"), and $\mathbf{G}^{\leq k}$ ("bounded always") are temporal operators.

The semantics of CSLTL and BLTL path formulas are defined over infinite traces over $2^{\bar{\Theta}}$. Let $\xi = \{\xi_i\}_{i=0}^{\infty}$ with $\xi_i \in 2^{\bar{\Theta}}$ be an infinite trace and $\xi^i = \xi_i\xi_{i+1}\ldots$ be the i-th suffix. Notation $\xi \models \varphi$ indicates that ξ satisfies formula φ and is recursively defined as following:

- $\xi \models \theta$ if $\theta \in \xi_0$;
- $\xi \models \neg\varphi$ if $\xi \not\models \varphi$;
- $\xi \models \varphi_1 \vee \varphi_2$ if $\xi \models \varphi_1$ or $\xi \models \varphi_2$;
- $\xi \models \varphi_1 \wedge \varphi_2$ if $\xi \models \varphi_1$ and $\xi \models \varphi_2$;
- $\xi \models \mathbf{X}\varphi$ if $\xi^1 \models \varphi$;
- $\xi \models \varphi_1\mathbf{U}\varphi_2$ if $\exists k \geq 0$, $\xi^k \models \varphi_2$, and $\forall i \in [0, k)$, $\xi^i \models \varphi_1$;
- $\xi \models \mathbf{F}\varphi$ if $\exists k \geq 0$, $\xi^k \models \varphi$;
- $\xi \models \varphi_1\mathbf{U}^{\leq k}\varphi_2$ if $\exists j \leq k$, $\xi^j \models \varphi_2$, and $\forall i \in [0, j)$, $\xi^i \models \varphi_1$;
- $\xi \models \mathbf{F}^{\leq k}\varphi$ if $\exists j \leq k$, $\xi^j \models \varphi$;
- $\xi \models \mathbf{G}^{\leq k}\varphi$ if $\forall j \leq k$, $\xi^j \models \varphi$.

Informally, $\mathbf{X}\varphi$ states that next φ becomes true; $\varphi_1\mathbf{U}\varphi_2$ states that φ_1 is true until φ_2 becomes true; $\mathbf{F}\varphi$ states that at some time in the future φ becomes true; $\psi_1\mathbf{U}^{\leq k}\psi_2$ states that ψ_2 becomes true within the first k time steps and ψ_1 remains true until that point; $\mathbf{F}^{\leq k}\psi$ states that ψ becomes true within the first k time steps; $\mathbf{G}^{\leq k}\psi$ states that ψ remains true in the first k time steps. A trace ξ satisfies a BLTL formula φ iff there exists a "good" finite prefix $\underline{\xi}$ of ξ such that the concatenation $\underline{\xi}\overline{\xi}$ satisfies φ for every suffix $\overline{\xi}$ [87], [95].

2.2.4 Summary

We conclude this Section via Table 2.1, which presents a synopsis of the classification of the different probabilistic models and their associated formal logics.

Model	Time	Actions	Transitions	Logic
CTMC	continuous	no	fixed rates	CSL
DTMC	discrete	no	fixed probabilities	CSLTL, BLTL
MDP	discrete	yes	fixed probabilities	CSLTL, BLTL
IMDP	discrete	yes	interval of probabilities	CSLTL, BLTL

Table 2.1: Classification of probabilistic models and the associated formal logic.

3
Fault Maintenance Trees

Contents

The standards imposed upon BAS require the enforcement of rigorous inspection and maintenance schemes. Timely maintenance increases both system dependability and its life span [108], [122]. However, the task to find the optimal trade-off between diminishing the chances of downtime and avoiding unnecessary overhead maintenance costs is a complex problem. Within this domain, the correct and dependable operation of the premises is subject to all its sub-components working at sufficient capacity. A crucial component is the Heat, Ventilation, and Air-Conditioning unit (HVAC) that regulates temperature and air circulation. The lifespan and reliability of the HVAC can be improved by coupling early fault detection with maintenance actions [20], [155].

Maintenance: An activity in which repairing is carried out at certain intervals to extend the useful life of the machine [143].

There are three prominent-approaches for performing timely maintenance:

- *preventative* - implemented at predetermined time intervals and prescribed guidelines to reduce the probability of failure or to prevent the degradation of function [122], [172]. [125] proposes a planning methodology for performing timely maintenance actions based on the remaining useful life characteristics of the individual sub-components within an HVAC system. [170] provide a tool that aids in the selection of maintenance policies. The tool makes use of historical failure data and expert input from which a Bayesian network representing conditional failure probabilities given certain maintenance actions is constructed.

- *condition-based* - recommends maintenance decisions based on the information collected through monitoring the current state of the system [86]. [171] makes use of performance indices (PI) to indicate the health condition (normal or faulty) of different sub-systems within the HVAC system: e.g. heating pumps, air handling unit. Regression models are then used to estimate the PIs as benchmarks for comparison with monitored PIs, from which faults are detected and a maintenance action performed. [96] construct a MDP representing the relationship between a component's degradation state and the environment. Using the MDP, dynamic programming is then performed to compute the optimal maintenance strategies.

- *predictive* - a maintenance strategy constructed based on the forecast degradation in the performance of a system. It has been applied to construct optimal policies for an HVAC system as a whole in [157]. A decision tree that combines techniques based on linear regression and neural networks is constructed to predict the degradation of HVAC components, from which an optimal policy is computed. [42], [114] have applied predictive maintenance

for the optimal operation of the boiler within the HVAC system, to maintain user comfort and reliable operation while minimising maintenance costs. [20] presents a holistic framework that combines energy management and comfort-driven maintenance. A dual control formulation is constructed where actions are selected based on the estimation of both the thermal and degradation dynamics.

A thorough review of all the three approaches can be found in [9] and [143].

In this chapter, we introduce a novel framework for performing preventative maintenance and show how this can be applied within the domain of smart buildings: namely by decomposing the various fault modes of the system, encode component degradation together with the relationships between faults into CTMCs, and for a fixed maintenance strategy compute the high-level KPI metrics (see Chapter 2). The work is built upon the novel framework of Fault Maintenance Trees (FMT) introduced by [138]. FMTs extend fault trees by introducing maintenance driven concepts like inspections and partial degradation of components. They enable an in-depth study of the relevant KPI metrics, which then serve as a platform for planning concrete improvements on the implemented maintenance policy. This chapter is structured as follows: Section 3.1 presents the fundamental theoretical basis for FMTs. Section 3.2 introduces a framework for analysing FMTs using probabilistic model checking (PMC) and a state-space reduction technique to alleviate the problem of state space explosion. Next, we present a case study derived from within the domain of smart buildings and further provide a comparison between the devised framework and the current state of the art [138] (see Section 3.3).

3.1 Background: Fault trees and its variants

Fault trees [159] are directed acyclic graphs (DAG) consisting of two types of nodes: *events* and *gates*. An event is an occurrence within the system, e.g. the failure of a subsystem down to an individual component. Events can be divided into *basic events* (BE) and *intermediate events* (IE). BE correspond to the leaves in fault trees

and denote atomic failures in the system, typically modelled via random variables following the exponential distribution. IE correspond to internal events that are caused by one or more other events. The event at the top of the tree called the *top level event* (TLE), represents the main event being analysed. This corresponds to a failure of the whole system under consideration. The internal nodes of the graph are called *gates* and describe how failures in BE and lower level gates interact as they propagate towards the TLE. Each gate has one output and one or more inputs. Gates in (standard) fault trees are *static*, in the sense that their output at any point in time depends solely on the configuration of their inputs at that moment. Different gates model different logical interactions. These include:

- OR gates, which require only one child to fail to propagate a failure to the next level,
- AND gates, which require all children to fail, and
- VOT$_k$ gates, which fail if more than k children fail [160].

In standard fault trees analysis [160], a closed-form solution exists, provided the distribution parameters of the BE is known. Figure 3.1 shows a fault tree instance.

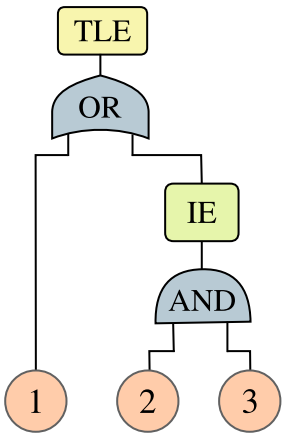


Figure 3.1: Fault tree with three basic events: BEs 2 and 3 are connected to an AND gate which triggers an IE. The IE is connected to an OR gate together with BE 1 and propagates failures to the TLE.

Dynamic fault trees (DFT) are a proper superset of fault trees, extended with gates that exhibit time- or order-dependent behaviour, e.g. spares (composed of a main component and equivalent spares; when the main component fails, a spare is used without triggering a failure at the gate). This adds a level of complexity that rules out analytical approaches for DFT [69], [162]. Consequently, there seems to be relatively scarce literature on DFT that support component health decay combined with preventive maintenance, e.g. acting before component failure [139]. FMT are a superset of DFT enriched with maintenance concepts [138]. This is achieved by the introduction of:

1. *Extended Basic Events* (EBE) - The BE 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. Figure 3.2 depicts the timing diagram showing the progression of degradation within an EBE. The EBE has N discrete degradation levels. Initially, the EBE is in the *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.

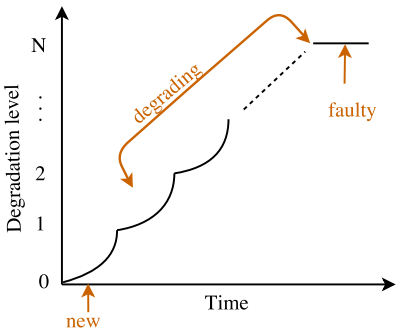


Figure 3.2: Timing diagram of degradation within an EBE.

The subsequent step- and time- wise degradation allows for e.g. identifying lightly degraded components, whose health can be restored before an actual failure (that may trigger a TLE) occurs.

2. *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 are triggered 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 3.3. The effect of performing a maintenance cleaning or replacement action on the EBE is illustrated in Figure 3.4. When a 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.

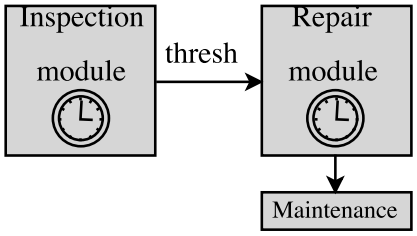


Figure 3.3: High-level description of the inspection and repair modules. The repair module performs maintenance actions periodically (clean or replace). The inspection module performs inspections periodically and when the degradation level of an EBE reaches *thresh* level, it triggers the repair module to perform a maintenance action immediately.

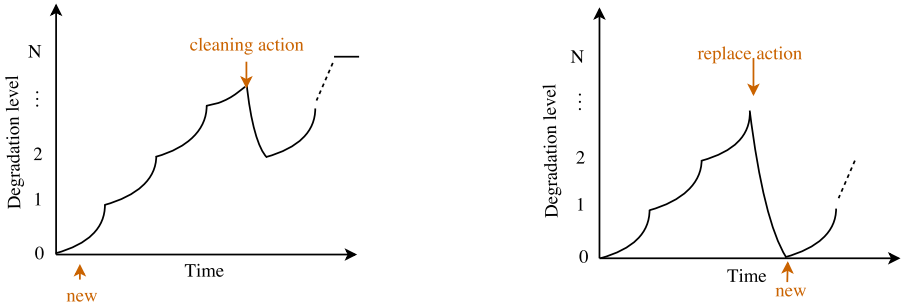


Figure 3.4: Degradation level progression of EBE for different maintenance actions.

A visual rendering of a FMT is given in Figure 3.5. It is composed of five EBEs located at the bottom of the tree, three gates, one repair and inspection module and three events (one TLE and two IEs) which show the different fault stages.

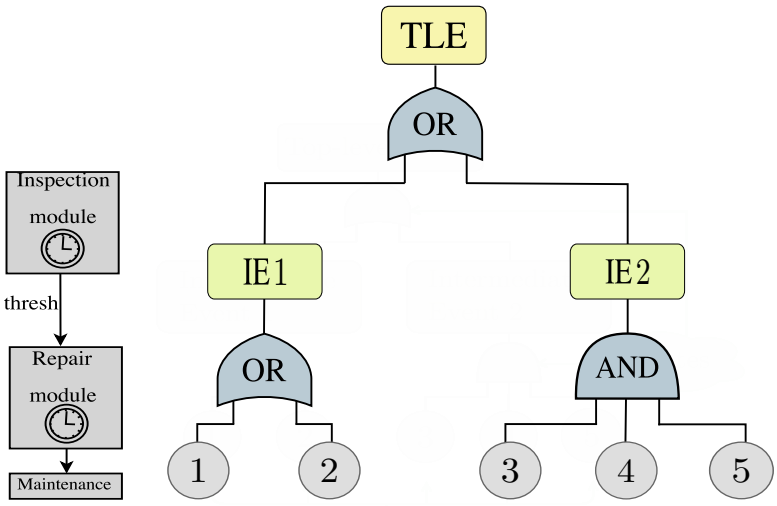


Figure 3.5: An example of a fault maintenance tree with five extended basic events: EBEs 1 and 2 are connected to an OR gate, while EBE 3, 4 and 5 are connected to an AND gate which trigger the IEs 1 and 2 respectively. The IEs are connected together via an OR gate and propagate failure to the TLE. The RM and IM are also included to implement the maintenance actions.

3.2 Probabilistic model checking of fault maintenance trees

A FMT is descriptive and convenient for reasoning about system health decay. In order to reason over this description, a FMT needs to be given *semantics* such that it can be quantitatively analysed. In literature, FMTs are analysed using SMC [138] where *priced time automata* (PTA) [24] are used to instantiate the FMT and the SMC tool UPPAAL [104] is used to measure KPIs. In this work, we provide a novel framework for analysing FMTs PMC. We use CTMCs to instantiate the FMT, encode the KPI metrics into the CSL logic and make use of the PMC tool PRISM [100] to reason about the FMT.

3.2.1 Formalising FMT using CTMC

To formalise the syntax of FMT using CTMC, we first define the set $\mathcal{F}$, characterising 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 EBE, RM and IM. We restrict the set $\mathcal{F}$ to contain the EBE, OR gate, DELAY, RM and IM modules since these will be the components used. In previous work, PTA are used to model FMT which encode time within the formalisation itself, thus the DELAY element is not required.

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

- 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.

- 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 $= (n, T_{rep}, T_{oh}, T_{insp}, T_{cln}, T_{rplc}, \mathit{thresh}, \mathit{trig})$ represents the RM module which acts on n EBE. 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 $= (n, T_{insp}, T_{cln}, T_{rplc}, \mathit{thresh})$ represents the IM module which acts on n EBE. 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 did 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 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 DAG $G = (V, E)$ 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 EBE, IE or a TLE. For G to be a well-formed FMT, we take the following assumptions (i) vertices are composed of the gates, (ii) there is only one TLE, and (iii) RM and IM are not part of the DAG but are modelled separately Note that for different FMTs the same RM and IM modules are used, thus the RM and IM modules are independent of FMT structure.

Definition 1 (Fault Maintenance Tree (FMT))**.** *A* FMT *is a directed acyclic graph* $G = (V, E)$ *composed of vertices* V *and edges* E*.*

Next, we provide the semantics for each FMT element, which are composed using the syntax of CTMC. 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. To model the deterministic time signals, we make use of the *Erlang distribution*. This is a similar approach taken in [138] where degradation phases are approximated by a (k,λ)-Erlang distribution.

Remark 1. *A random variable $Z \in \mathbb{R}_{>0}$ has an Erlang distribution with $k \in \mathbb{N}$ stages and a rate $\lambda \in \mathbb{R}_{>0}, Z \sim Erlang(k, \lambda)$, 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*

$$f(t; k, \lambda) = 1 - \sum_{n=0}^{k-1} \frac{1}{n!} \exp(-\lambda t)(\lambda t)^n \quad \textit{for } t, \lambda \geq 0, \tag{3.1}$$

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 $Z_k \sim Erlang(k, \frac{k}{T})$ [47]. Note that 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) [83]; and that the Erlang stages are only required for time-dependent analysis.

DELAY The DELAY generates an approximate deterministic delay T using N states. Figure 3.6 describes the corresponding CTMC using the set of states given by $\mathcal{S}_T = \{d_0, d_1, \ldots, d_{N+1}\}$, the set of transitions labels $\Upsilon = \{\texttt{trigger}, \texttt{move}\}$, the set of atomic propositions $\Theta = \{T\}$ with $L(d_0) = \cdots = L(d_N) = \{\emptyset\}$, and $L(d_{N+1}) = \{T\}$. The rate matrix $\mathcal{R}$ becomes clear from Figure 3.6 and

$$\mathcal{R}_{ij} = \begin{cases} \eta & i = 0 \wedge j = 1 \wedge \texttt{trigger}, \\ \frac{N}{T} & (((i \geq 1 \vee i < N+1) \wedge j = i+1) \vee (i = N+1 \wedge j = 1)) \wedge \texttt{move}, \\ 0 & \text{otherwise}, \end{cases} \tag{3.2}$$

with i representing the current state, j is the next state and η is a fixed large value corresponding to introducing a negligible delay that is used to trigger all the DELAY modules at the same time [1]

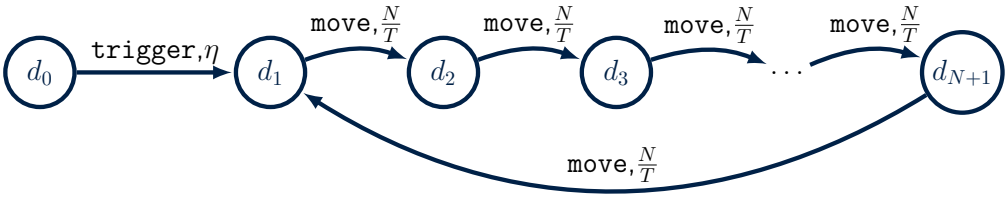


Figure 3.6: CTMC representing DELAY with N states used to approximate a delay equal to T approximated using $Erlang(N, \frac{N}{T})$. The transition labels $\Upsilon = \{\texttt{trigger}, \texttt{move}\}$ are shown on each of the transitions, while for clarity the state labels are not shown.

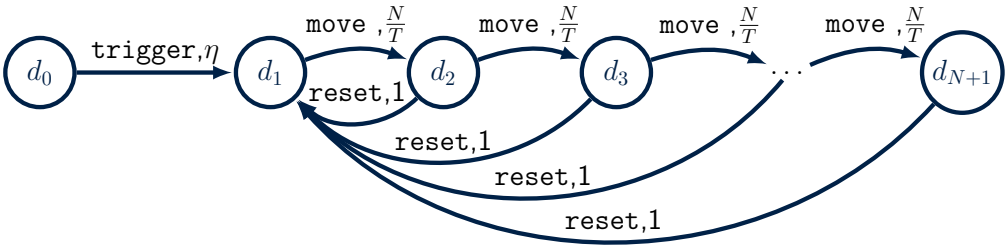


Figure 3.7: CTMC representing the extended DELAY with N states used to approximate a delay equal to T. Delay approximated using $Erlang(N, \frac{N}{T})$. The transition labels $\Upsilon = \{\texttt{trigger}, \texttt{move}, \texttt{reset}\}$ are shown on each of the state transitions, while the state labels are not shown.

In Figure 3.7 we define the semantics of $(\text{DELAY}, T, N)_{ext}$. This results in the CTMC described using the state space $\mathcal{S}_T = \{d_0, d_1, \ldots, d_{N+1}\}$, the set of transition labels $\Upsilon = \{\texttt{trigger}, \texttt{move}, \texttt{reset}\}$, the set of atomic propositions $\bar{\Theta} = \{T\}$, the labelling function $L(d_0) = L(d_1) = \cdots = L(d_N) = \{\emptyset\}$, and $L(d_{N+1}) = \{T\}$, and the rate matrix $\mathcal{R}$ where

$$\mathcal{R}_{ij} = \begin{cases} \eta & i = 0 \wedge j = 1 \wedge \texttt{trigger}, \\ 1 & (i \geq 2 \vee i < N+1) \wedge j = 1 \wedge \texttt{reset}, \\ \frac{N}{T} & (((i \geq 1 \vee i < N+1) \wedge j = i+1) \vee (i = N+1 \wedge j = 1)) \wedge \texttt{move}, \\ 0 & \text{otherwise}, \end{cases} \tag{3.3}$$

with i representing the current state and j is the next state. 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

[1]The use of a large-valued η to simulate instantaneous transitions may lead to stiff models, making the time-bounded analyses computationally hard. To mitigate this potential bottleneck one can make use of interactive Markov chains [67] or Markov automata [68], which allow for modelling of instantaneous transitions.

approximation whenever the guard condition is met at a rate of $1 \times \mathcal{R}_{sync}$ where $\mathcal{R}_{sync}$ is the rate coming from the use of synchronisation with other modules causing the reset to occur. 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.

Extended Basic Events (EBE) In essence, the EBE starts in an initial phase which represents a component in perfect operational condition, aka "as good as new." As time passes the component will eventually *degrade* by internally moving to the next phase, which represents a decay in the health of the component. This process repeats until the component ultimately reaches its final phase; when this happens the EBE *failed* and sends a signal that propagates through the FMT.

Figure 3.8 shows the semantics of the EBE $= (T_{deg}, T_{cln}, T_{rep}, N = 4)$. The corresponding CTMC is described by the tuple $\mathcal{C}_{\text{EBE}} = (\mathcal{S}_{\text{EBE}} = \{s_0, s_1, s_2, s_3, s_4\}, \Upsilon_{\text{EBE}}$ $\mathcal{R}_{\text{EBE}}, \bar{\Theta}_{\text{EBE}}, L_{\text{EBE}})$ with the transition labels

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

the rate matrix

$$\mathcal{R}_{\text{EBE},ij} = \begin{cases} \eta & i = 0 \wedge j = 0 \wedge (\texttt{trigger}), \\ 1 & i = \{1, \ldots, N-1\} \wedge j = i+1 \wedge \texttt{degrade}_i, \\ 1 & i = \{2, \ldots, N\} \wedge j = i-1 \wedge \texttt{perform\_clean}, \\ 1 & i = 1 \wedge j = 1 \wedge \texttt{perform\_clean}, \\ 1 & i = \{1, \ldots, N\} \wedge j = 1 \wedge \texttt{perform\_replace}, \\ 0 & \text{otherwise}, \end{cases},$$

the atomic propositions $\bar{\Theta}_{EBE} = \{new, thresh, failed\}$ and the labelling function L_{EBE} is such that $L_{EBE}(s_0) = \{\emptyset\}$, $L_{EBE}(s_1) = \{new\}$, $L_{EBE}(s_2) = L_{EBE}(s_3) = \{thresh\}$, $L_{EBE}(s_4) = \{failed\}$.

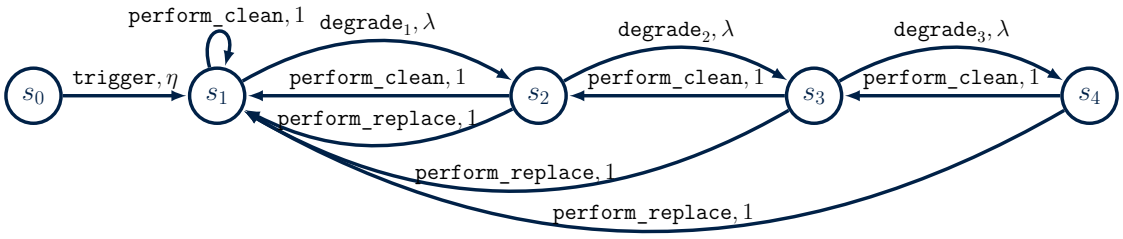


Figure 3.8: CTMC representing the EBE with $N = 4$ degradation phases and transition labels $\Upsilon_{EBE} = \{\texttt{trigger}, \texttt{degrade}_{i\in\{1,2,3,4\}}, \texttt{perform\_clean}, \texttt{perform\_replace}\}$ on each of the state transitions. For clarity, the state labels are not shown. The 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.

Mean Time To Failure [139]: The MTTF describes the expected time from the moment the system becomes operational, to the moment the system subsequently fails and is formally defined using,

$$\text{MTTF}(T) = \frac{\ln(1 - D_e(T))}{-T}. \tag{3.4}$$

where, $D_e(T)$ is the probability of failure at time T.

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 $\texttt{degrade}_N$ such that synchronisation between the two CTMC is performed. 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 [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

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

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 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.

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 IE). We use

$$FAIL = \begin{cases} 0 & E_1 = 1 \wedge \cdots \wedge E_n = 1, \\ 1 & \text{otherwise}, \end{cases} \tag{3.5}$$

where $E_i = 1, \{i \in 1 \ldots n\}$ corresponds to when the sn 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 3.9 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 $\mathcal{S}_{rm} = \{rm_0, rm_1\}$, the transition labels

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

the rate matrix

$$\mathcal{R}_{\text{RM},ij} = \begin{cases} 1 & i = 0 \wedge j = 0 \wedge (\texttt{check\_maintenance,trig = 0} \vee \texttt{inspect,thresh = 0}), \\ 1 & i = 0 \wedge j = 1 \wedge (\texttt{check\_maintenance,trig = 1} \vee \texttt{inspect,thresh = 1}), \\ 1 & i = 1 \wedge j = 1 \wedge \texttt{trigger\_maintenance}, \\ 0 & \text{otherwise}, \end{cases}$$

with `maintenance = {clean, replace}`, the atomic propositions

$$\bar{\Theta} = \{maintenance\}$$

and the labelling function is designed such that

$$L(rm_0) = \{\emptyset\}, L(rm_1) = \{maintenance\}.$$

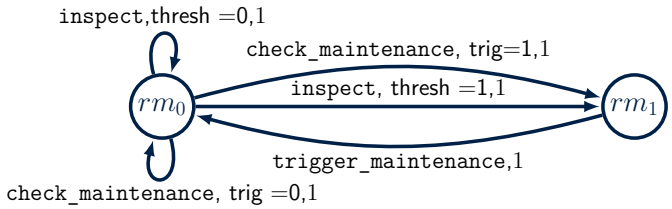


Figure 3.9: CTMC representing the RM with $\Upsilon_{RM} = \{$`inspect`, `check_maintenance`, `perform_maintenance`$\}$ shown on the state transitions. The guard condition $trig = 0/1$ or $thresh = 0/1$ must be satisfied for the corresponding transition to trigger when it is activated via synchronisation with the transition label.

For brevity in Figure 3.9, 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 3.2.1). 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

$$thresh = \begin{cases} 1 & L(s_{j,1}) = thresh \vee \cdots \vee L(s_{j,n}) = thresh, \\ 0 & \text{otherwise}, \end{cases} \tag{3.6}$$

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

$$trig = \begin{cases} 1 & L(s_{j,1}) \neq new \vee \cdots \vee L(s_{j,n}) \neq new, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

Both signals act as guards which when triggered determine which transition to perform (cf. Figure 3.9).

Inspection module (IM) The semantics of the IM $= (n, T_{insp}, T_{cln}, T_{rplc}, thresh)$ is depicted in Figure 3.10. The CTMC is defined using the tuple $\mathcal{C}_{\text{IM}} = (\mathcal{S}_{\text{IM}} = \{im_0, im_1\}, \Upsilon_{IM}, \mathcal{R}_{IM}, \bar{\Theta}_{IM}, L_{IM})$ where

$$\begin{aligned} \Upsilon_{\text{IM}} &= \{\texttt{inspect}, \texttt{perform\_clean}, \texttt{perform\_replace}\}, \\ \mathcal{R}_{\text{IM},ij} &= \begin{cases} 1 & i = 0 \wedge j = 0 \wedge \texttt{inspect,thresh = 0}, \\ 1 & i = 0 \wedge j = 1 \wedge \texttt{inspect,thresh = 1}, \\ 1 & i = 1 \wedge j = 1 \wedge (\texttt{perform\_clean} \vee \texttt{perform\_replace}), \\ 0 & \text{otherwise}, \end{cases}, \\ \bar{\Theta}_{\text{IM}} &= \{\emptyset\}, L_{IM}(s_0) = L_{IM}(s_1) = \{\emptyset\}. \end{aligned}$$

The *thresh* signal corresponds to same signal used by the RM, given using (3.6). In Figure 3.10, 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.

Semantics of composed FMT

Next, we show how to obtain the semantics of a FMT from the semantics of its elements. We designate 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

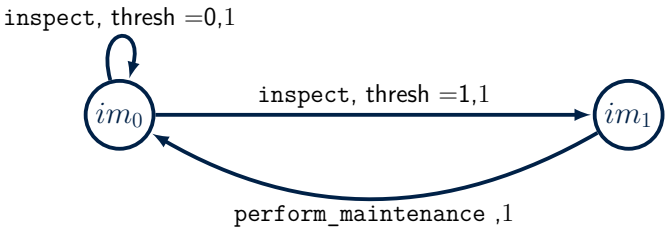


Figure 3.10: CTMC representing the IM with $\Upsilon_{IM} = \{\texttt{inspect}, \texttt{perform\_maintenance}\}$ shown on the state transitions. The guard condition $trig = 0$ and $thresh = 1$ must be satisfied for the corresponding transition to trigger when it is activated via synchronisation with the transition label.

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 (3.6) and (3.7) 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 [82] and defined as follows

Interleaving Synchronization: The interleaving synchronous product of $\mathcal{C}_1 = (\mathcal{S}_1, \mathcal{R}_1, \Upsilon_1, \bar{\Theta}_1, L_1)$ and $\mathcal{C}_2 = (\mathcal{S}_2, \mathcal{R}_2, \Upsilon_2, \bar{\Theta}_2, L_2)$ is $\mathcal{C}_1||\mathcal{C}_2 = (\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{R}, \Upsilon_1 \cup \Upsilon_2, \bar{\Theta}_1 \cup \bar{\Theta}_2, L_1 \cup L_2)$ where $\mathcal{R}$ 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 $s_1, s_1' \in \mathcal{S}_1$, $\alpha_1 \in \Upsilon_1$, $\mathcal{R}_1(s_1, s_1') = \lambda_1$, $s_2, s_2' \in \mathcal{S}_2$, $\alpha_2 \in \Upsilon_2$, $\mathcal{R}_2(s_2, s_2') = \lambda_2$.

Full Synchronization: The full synchronous product of $\mathcal{C}_1 = (\mathcal{S}_1, \mathcal{R}_1, \Upsilon_1, \bar{\Theta}_1, L_1)$ and $\mathcal{C}_2 = (\mathcal{S}_2, \mathcal{R}_2, \Upsilon_2, \bar{\Theta}_2, L_2)$ is $\mathcal{C}_1||\mathcal{C}_2 = (\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{R}, \Upsilon_1 \cup \Upsilon_2, \bar{\Theta}_1 \cup \bar{\Theta}_2, L_1 \cup L_2)$ where $\mathcal{R}$ 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 $s_1, s_1' \in \mathcal{S}_1$, $\alpha \in \Upsilon_1 \wedge \Upsilon_2$, $\mathcal{R}_1(s_1, s_1') = \lambda_1$, $s_2, s_2' \in \mathcal{S}_2$, $\alpha_2 \in \Upsilon_2$, $\mathcal{R}_2(s_2, s_2') = \lambda_2$.

For any pair of states, synchronisation is performed either using interleaving or full synchronisation. For full synchronisation 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 3.9), while the rate of the DELAY modules represent the actual rates (cf. Figure 3.6 and Figure 3.7). The same principle holds for the EBEs and the IM. We refer the reader to Table 3.1 to further understand the synchronisation between the FMT components and the method employed for parallel composition.

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	-	Interleaving synchronisation

Table 3.1: Performing synchronisation between the different FMT components and the synchronisation method used.

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 $\mathcal{C}_{all} = \mathcal{C}_{EBE}|| \ \mathcal{C}_{T_{deg}}||\mathcal{C}_{RM}||\mathcal{C}_{IM}||\mathcal{C}_{T_{cln}} \ ||\mathcal{C}_{T_{insp}}||\mathcal{C}_{T_{rep}}$. The resulting state space is then $\mathcal{S}_{all} = \mathcal{S}_{EBE} \times \mathcal{S}_{T_{deg}} \times \mathcal{S}_{RM} \times \mathcal{S}_{IM} \times \mathcal{S}_{T_{cln}} \times \mathcal{S}_{T_{insp}} \times \mathcal{S}_{T_{rep}}$. The synchronisation between the different components is shown in Figure 3.11 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 are triggered (cf. Figure 3.9 and 3.10). 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`.

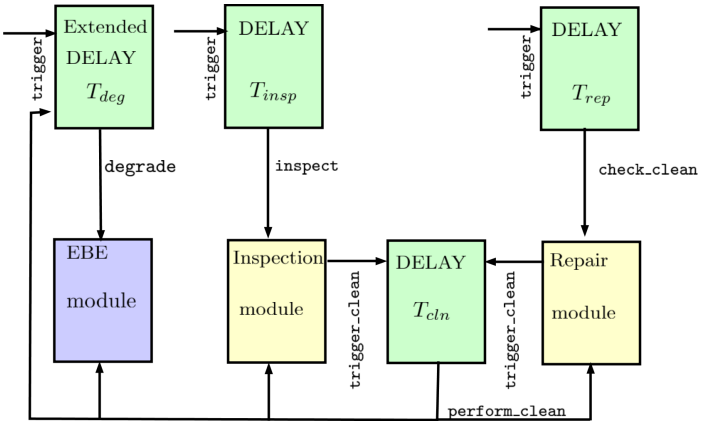


Figure 3.11: Block diagram showing the synchronisation connections between one component and the other, together with the corresponding transition label which triggers synchronisation.

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 affect we propose a state-space reduction framework in Subsection 3.2.2.

3.2.2 Decomposition of FMTs

The use of CTMC and deterministic time delays results in a large state space for modelling the whole FMT. We, therefore, propose an approach that decomposes the large FMT into an equivalent abstract CTMC that can be analysed using the PMC tool PRISM [100]. The process involves (i) splitting the FMT graph into a set of smaller sub-graph, (ii) transforming each sub-graph into an equivalent CTMC and (iii) sequentially composing the smaller sub-graphs to generate the higher-level abstract FMT. The transformation from DAG to CTMC is carried out by making use of the developed FMT components semantics and performing a parallel composition of the individual components based on the underlying structure of the sub-graph. Figure 3.12 depicts a high-level diagram of the decomposition procedure.

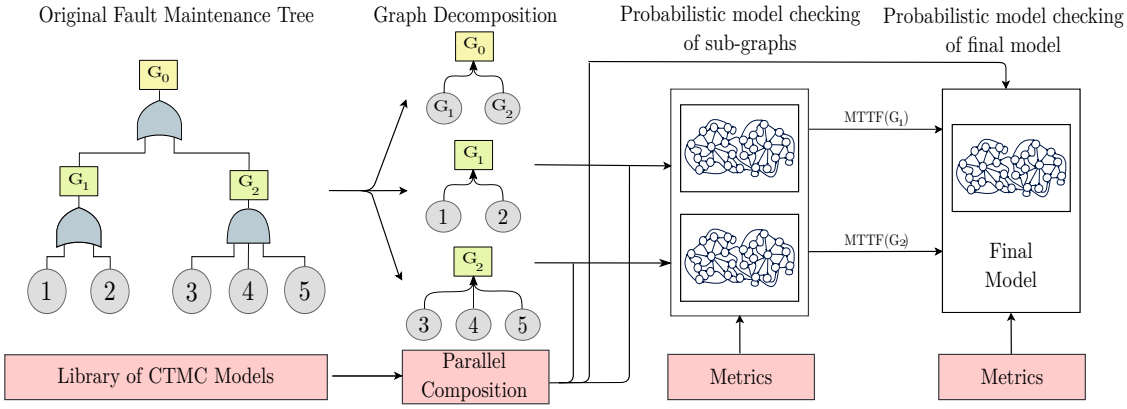


Figure 3.12: Overall developed framework for decomposition of FMTs into the equivalent abstract CTMCs.

Graph decomposition We define modules within the DAG as sub-trees composed of at least two events that have no inputs from the rest of the tree and no outputs to the rest except from its output event [109]. 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 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. 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

```
1: procedure DECOMPOSE(G)                          ▷ The DAG G = (V, E)
2:     identify sub-graphs using depth-first traversal
3:     label all top nodes of each sub-graph i as V_{T_i}
4:     for select the top node of every sub-graph and the child defined at the
   immediate lower level do
5:         if label V_T already found in one of the leaf nodes of sub-graph then
6:             split sub-graph
7:             insert new node V_g which will be used as input
8:         end if
9:     end for
10: end procedure
```

Probabilistic Model Checking of sub-graphs We start from the bottom level sub-graphs and perform the conversion to CTMC using the formal models. Each component is represented by a PRISM module 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. For each sub-graph, we compute the probability of failure

$D_e(T)$ at time T, from which we calculate the MTTF [139] using (3.4). 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 $T_{deg} = \text{MTTF}$, 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 3.13 depicts the steps needed to perform PMC for one of the sub-graphs.

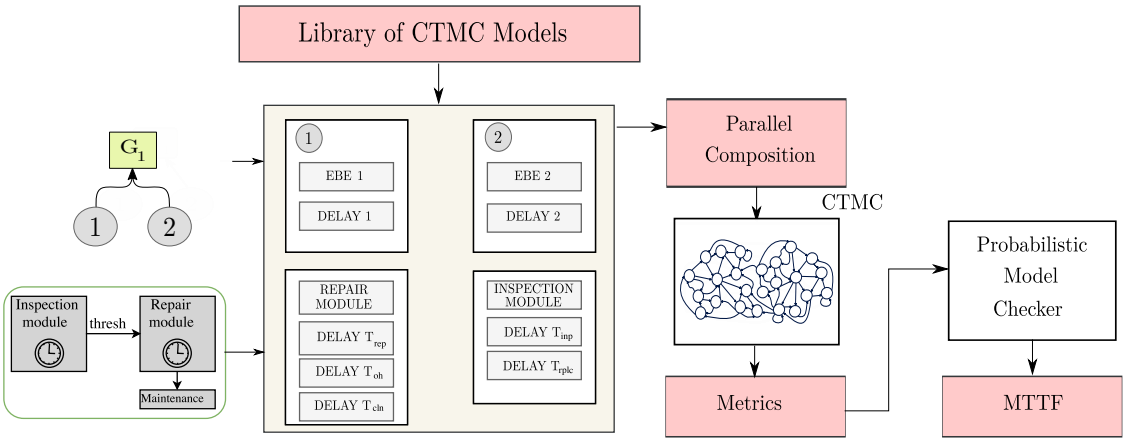


Figure 3.13: Method for performing PMC of sub-graphs. Given the subgraph with IE labelled with G_1, we select the corresponding CTMC models for the two EBEs, the RM and IM modules and the corresponding DELAY modules. We perform parallel composition to obtain the full abstract CTMC and compute the probability of failure (D_e), at time T, via PMC. Using D_e, we attain the MTTF. For a different time horizon we recompute the MTTF without the need of reperforming the parallel composition step.

Probabilistic Model Checking of the final equivalent abstract CTMC

On reaching the top level node V_o, we compute the metrics for the equivalent abstract CTMC with 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 requiring less memory and computational time due to the significantly smaller state space of the underlying CTMC. Figure 3.14 presents a FMT composed of four EBEs, one IE and TLE, 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 3.12).

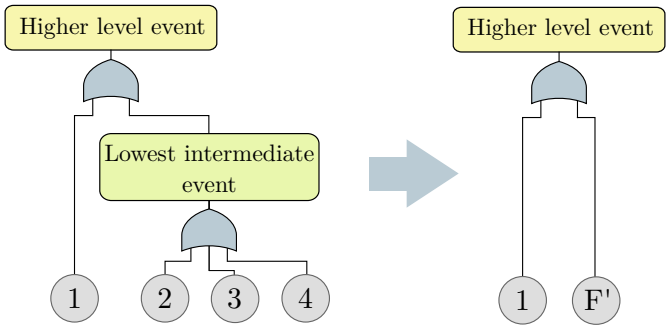


Figure 3.14: The original FMT and the abstract FMT corresponding to the equivalent abstract CTMC generated by the developed framework. The MTTF for F' is computed based on the probability of failure of the lowest intermediate event.

3.2.3 Metrics

We use the probabilistic model checker PRISM [100] to compute the KPIs of the model described. The metrics can be expressed using the CSLand are given by

1. *Reliability*: This can be expressed as the complement of the probability of failure over the time T, $1 - \mathrm{P}_{=?}[\mathbf{F}^{\leq T} failed]$

2. *Availability*: This can be expressed as $\mathrm{R}_{=?}[\mathbf{C}^{\leq T}]/T$, which corresponds to the cumulative reward of the total time spent in states labelled with *okay* and *thresh* during the time T.

3. *Expected number of failure*: This can be expressed using $\mathrm{R}_{=?}[\mathbf{C}^{\leq T}]$, 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. *Expected cost*: This can be expressed using $\mathrm{R}_{=?}[\mathbf{C}^{\leq T}]$, which corresponds to the cumulative reward of the total costs (operational, maintenance and failure) within the time T.

3.3 Application to building automation systems

We consider the HVAC system setup (a subset of the whole BAS) found within the Smart Buildings Laboratory. A graphical description is shown in Figure 3.15. It is composed of two circuits - the airflow circuitry and the water circuit. The gas boiler heats 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 to the boiler.

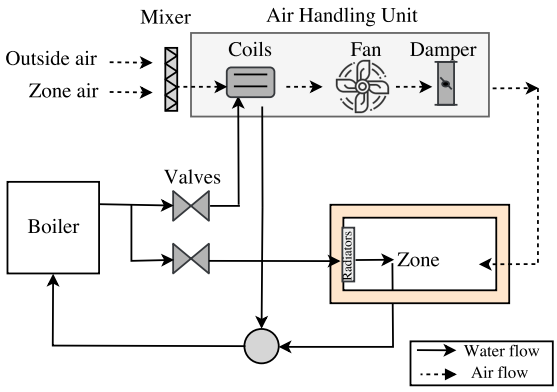


Figure 3.15: High-level schematic of an HVAC system.

The correct and continuous operation of the HVAC depends on the proper operation of its subsystems. Usually, all individual components from these subsystems have different expected lifetimes, as well as different sources of degradation and failures. For instance, the valves can get stuck due to the gradual increase in deposits from the water flow. This limits the rate of water flow in the heating system, consequently reducing the heat exchange rate. The supply fan gradually decreases its efficiency as the lifetime of the fan motor increases, while it may cease

to work if it breaks due to a failure of the fan bearings or the motor itself. The AHU damper may break due to the build up of fouling [158]. The radiator may fail due to rust and leak inhibitors, which form solids that collect in the radiator cooling system and restrict the water flow. The boiler heating capacity reduces with age due to corrosion and the accumulation of dust and dirt in the internal heat exchanger and flue [52]. The aforementioned issues are a subset of possible failure causes that may lead to downtime of the HVAC unit from Figure 3.15. When maintenance actions are being planned, all such (foreseeable) events must be considered to ensure compliance with the KPI requirements. However and in spite of the best efforts, the operation of the individual components—and thus the whole system—degrades as time passes.

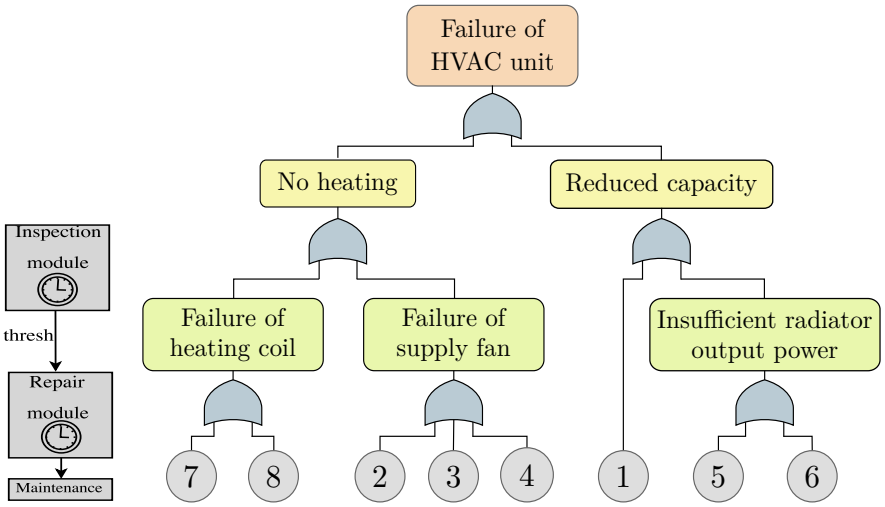


Figure 3.16: Fault maintenance tree for reasoning about failure of HVAC unit.

Based on this HVAC system we construct the corresponding FMT shown in Figure 3.16. For each EBE, all random variables describing the jumps from the initial to the failed phase are independent and have the same rate parameter. The number of phases and MTTF of EBEs is typically derived from measurements or manuals, and uniquely determines the degradation behaviour of the component, viz. the rate of the exponential random variables. All EBEs in Figure 3.16 are obtained from [10], [54]. In the Table 3.2, N is the number of degradation phases:

for instance, EBE 2 models a failure of the supply fan motor via a (random variable with distribution) Erlang($3, 3/35$).

ID	Component	N	MTTF
1	AHU damper broken	4	20
2	Fan motor failure	3	35
3	Supply fan obstructed	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	Heat pump failure	4	20

Table 3.2: Details of the EBEs from Figure 3.16.

3.3.1 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. The cleaning policy consisting of periodic *inspections*, *repair checks*, and *overhauls*. Each of these periodical service activities has a clear and distinct restoration purpose:

- Overhauls enforce a *replace* action that renews the whole HVAC, sending all EBEs back to their *new* phase.
  - When triggered, replace actions take one week (i.e. seven days) to complete.
  - This incurs in a cost of €5000 and is the highest point-wise investment of the whole policy.
- A *clean* action is performed if and only if some EBE is in a *thresh* phase (aka degraded but operational) at the moment when either the periodic cleaning or inspection takes place.
  - Every inspection incurs a cost of €5.

– When triggered, clean actions take a day.

– All degraded EBEs are cleaned on that day, following the same scheme implemented for repairs.

– When an EBE is cleaned, its health is restored by one degradation phase, which makes cleaning the lightest among all maintenance actions.

– A clean costs €100; however notice that, if no component is degraded when the inspections occur, no clean is performed and this cost is not incurred.

These maintenance procedures are implicitly implemented in the corresponding RM of the model. Moreover, all costs described above account for maintenance checks/actions. We also consider operational costs: €1 is accrued per day of system uptime and €4 per day of system downtime [114], [142]. For this set-up, all the KPI 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. We further consider five different maintenance strategies, listed in Table 3.3, such that we can identify the optimal 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 select strategies that have a different combination of repair, inspection and

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	15 years	0.5 years
M_4	4 years	30 years	1 year

Table 3.3: Implemented maintenance strategies.

replacement strategies to highlight the effect the different maintenance actions have on the HVAC system's performance. Figure 3.17 depicts the resulting metrics for the employed strategies.

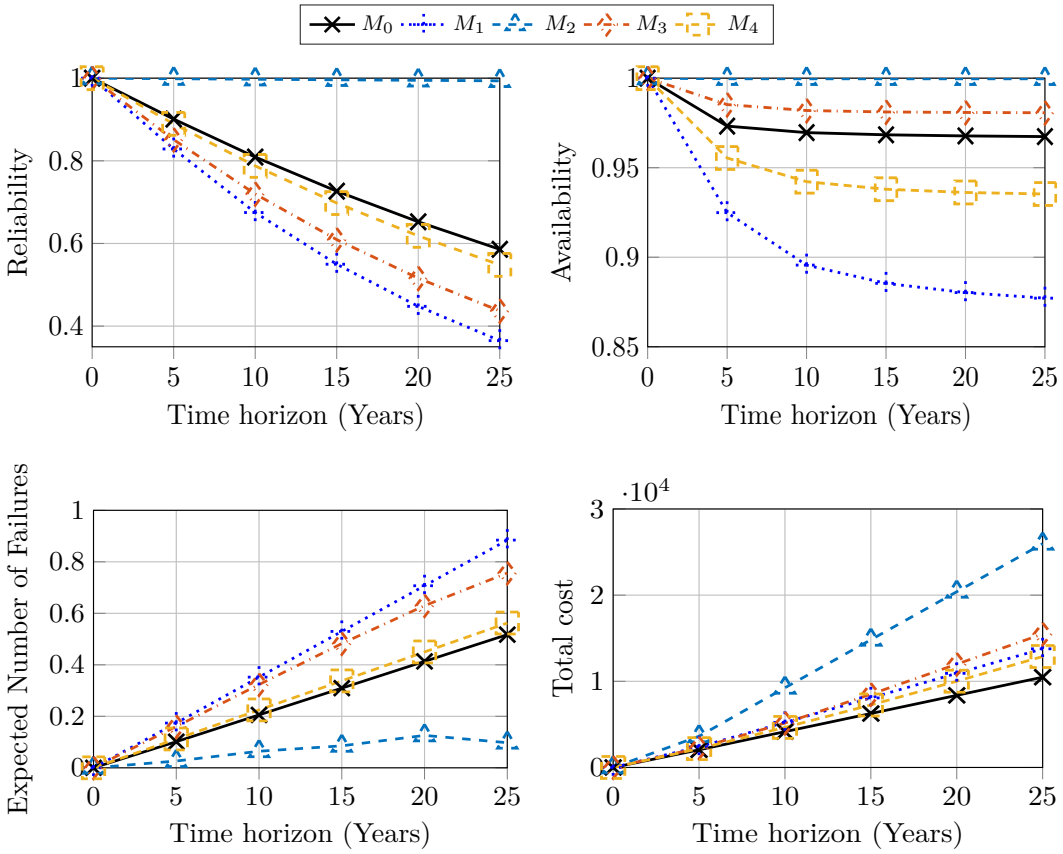


Figure 3.17: Comparing the KPIs under different maintenance strategies for the HVAC system.

We can deduce that the worst-performing strategy is when repair actions are carried out every 5 years with inspection carried out bi-annually and no replacements (corresponding to strategy M_1). Strategy M_2 provides high performance but with a significant increase in the total costs due to the frequent replacement actions. Comparing strategies M_3 and M_4 we can note that M_3 has the highest availability over the whole time horizon but this comes with higher total costs due to the higher frequency of replacements. Strategies M_0 and M_4 have similar performance with M_4 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, over 25 years, is strategy M_0 (i.e. with annual inspections, bi-annual repairs and no replacements).

Error analysis We perform a comparison between computing the KPIs with and without the graph decomposition framework. As highlighted in Section 3.2.2, the state space representation for analysis via PMC may require a large amount of memory. In particular, modelling the whole FMT without the decomposition requires more then 10^8 states, which is an upper-bound for the capabilities of the PRISM tool in the current computer settings [100] and thus, the metrics cannot be computed. However, in order to compare the two methods, we focus on a sub-tree within the BAS FMT with the IE labelled as *Reduced capacity*. This sub-tree is composed of EBEs 1, 5, and 6 (see Figure 3.16) and we fix the maintenance strategy to M_3 (i.e. inspections every 6 months, repairs every 2 years, and an overhaul every 15 years). Next, we compute the KPI metrics using both methods over six time horizons $N_r = \{0, 5, 10, 15, 20, 25\}$ years and present the result in Figure 3.18. We label the metrics PMC EXACT and PMC REDUCED for those generated using with and without the decomposition framework, respectively.

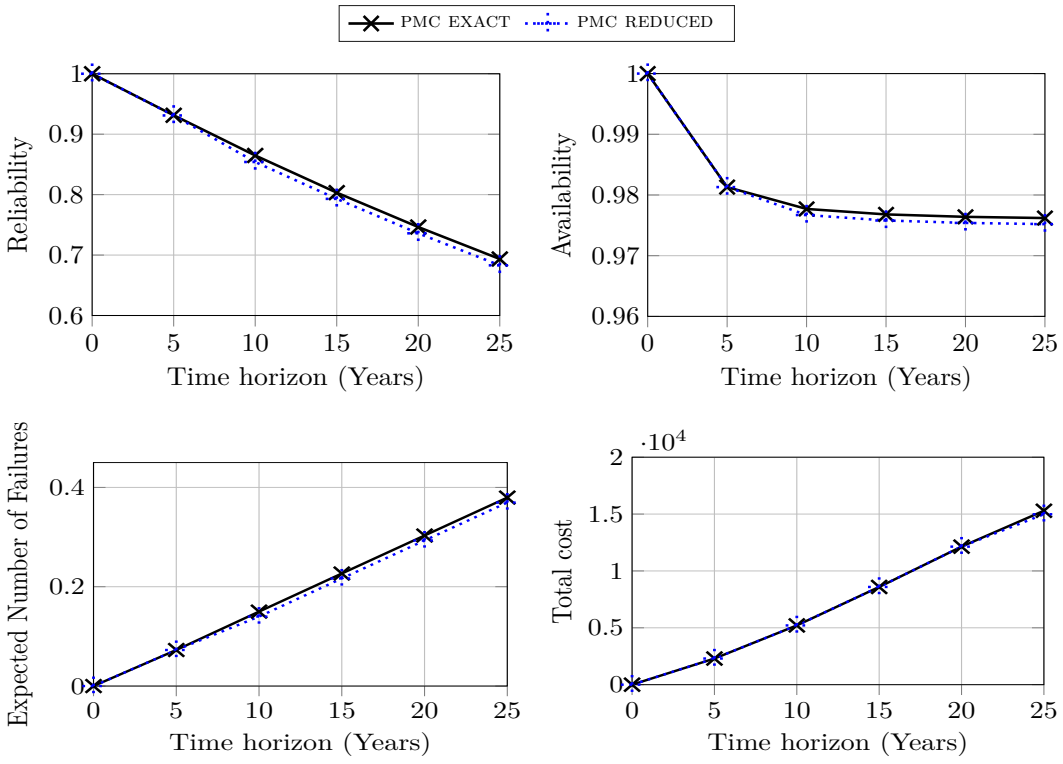


Figure 3.18: Comparing the KPIs computed with vs without the decomposition framework for the sub-graph with IE *Reduced capacity* under M_3.

As expected, a small gap is observed between PMC EXACT and PMC REDUCED. This is a consequence of the reduction technique, which replaces the OR gate on top of EBEs 5 and 6 for an EBE with equivalent MTTF. Consequently, instead of using the fastest firing time between two Erlang distributions, i.e. one for EBE 5 and one for EBE 6, in PMC REDUCED we measure the firing time of a single EBE with the same MTTF. This has the advantage of reducing the number of states of the CTMC analysed (everything below the replaced IE is now a single EBE), but creates a minor deviation in the general model behaviour. Such deviation should increase with the number of times this reduction is performed.

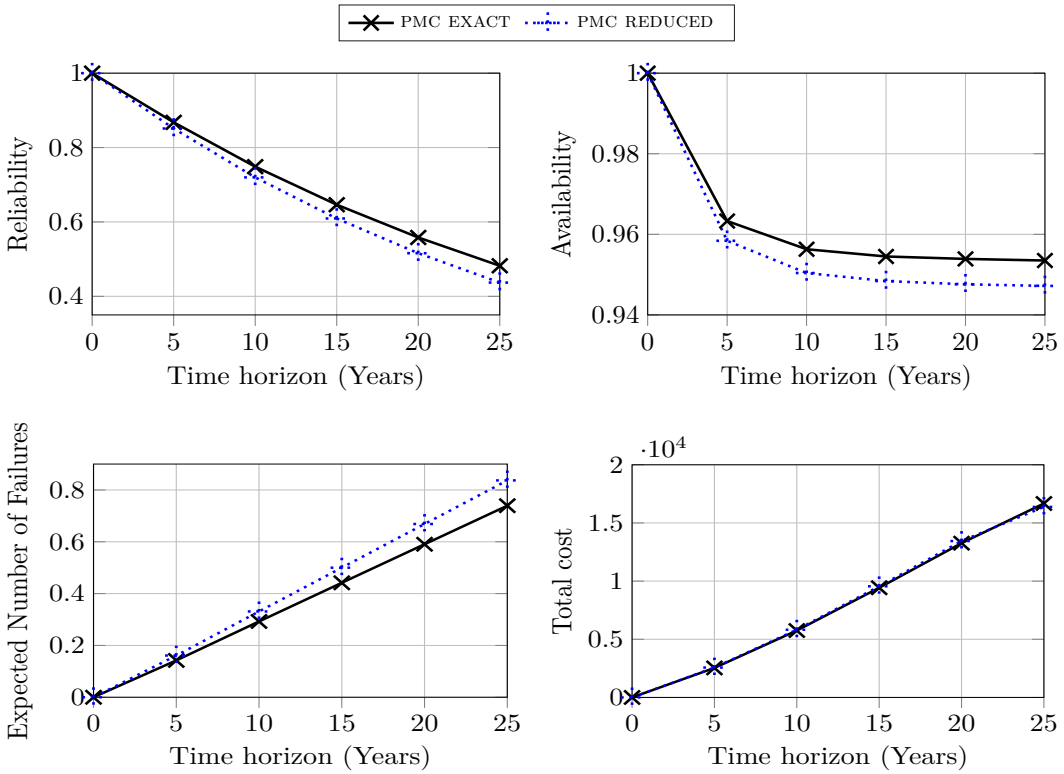


Figure 3.19: Comparing the KPIs computed with vs without the decomposition framework for the sub-graphs corresponding to the IEs *Reduced capacity* & *Failure in heating coil* under M_3.

In Figure 3.19 we show the KPI metrics for a subtree that includes the IEs *Reduced capacity'* and *Failure of heating coil*, i.e. EBEs 1 and 5–8. Again we experiment using M_3 as the maintenance strategy over five time horizons. We observe that the difference between PMC EXACT and PMC REDUCED is exacerbated, since more IEs have been reduced via the abstraction method.

We further split the maintenance costs into the individual sub-components making up the total cost: inspection, repairs, overhaul and cleaning actions and highlight the different contributions in Figure 3.20. The aggregation of these values, plus the operational system costs, composes the total costs from Figure 3.19.

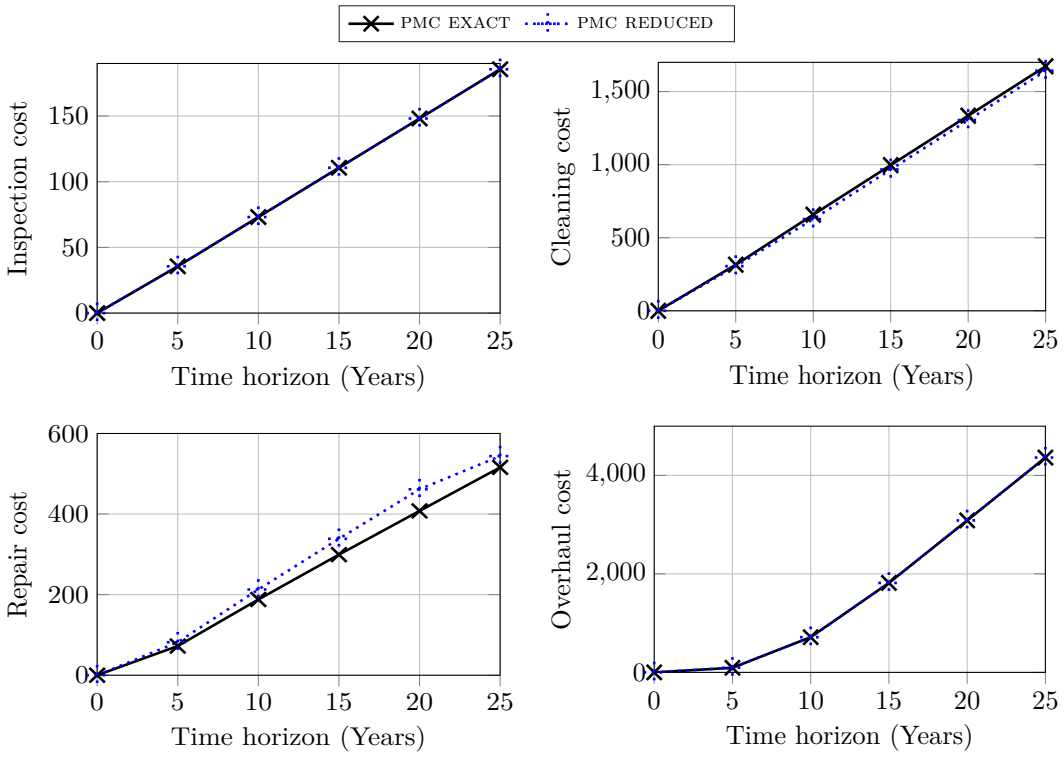


Figure 3.20: Comparing the maintenance costs computed with vs without the decomposition framework for the sub-graphs corresponding to the IEs *Reduced capacity & Failure in heating coil* under M_3.

Notice that the overhaul period for M_3 is 15 years, yet the plot shows positive costs incurred in replacements at 5 and 10 years. This is a consequence of approximating discrete time periods via 3-phase Erlang distributions: the mean time for an occurrence of the event, (e.g. an overhaul triggering a replace, is 15 years), but the effective time observed in particular simulations variates to some degree[3] around such value, causing the (averaged) behaviour shown.

[3] A random variable following an Erlang($3/15, 3$) distribution has standard deviation $\sqrt{\frac{3}{(3/15)^2}} = 5\sqrt{3}$.

Refining the maintenance actions In the basic setting of Subsection 3.3.1 (we call this HVAC-0), inspections and repair checks overlap considerably. Both can trigger the same maintenance action, namely, a clean, and both will do so in the same system configurations. The only situations when a clean would be triggered by a repair check and not by an inspection is when there is at least one failed—but no degraded—component. The likelihood of these scenarios decreases with the amount of EBEs and their number of degradation phases.

A more problematic modelling effect is that, when triggered by an inspection, a clean can "repair" a failed EBE and make it operational again. If e.g. EBE 1 is failed and EBE 2 is degraded, an inspection will trigger a clean because EBE 2 is in a *thresh* state; since cleaning actions are system-wide this also affects EBE 1, which then moves from its *failed* to a *thresh* state, becoming operational. We argue this is not a realistic behaviour. Consequently, we refine the maintenance actions such that there is a clear distinction between inspections and cleans. The periodic cleaning action by the RM now triggers a *cleaning* action if and only if some EBE is in its *failed* phase at the moment when the check takes place. The intuition is that an expert technician visits the premises regularly, fixing any component he finds in a *failed* state, namely broken. When triggered, such an action takes two days and costs €800. All broken EBEs are restored in those two days and have their health significantly restored: once the action is completed they are sent back $N-2$ degradation phases. The resulting phase for the component is the first one labelled *thresh* after the *new* phase, i.e. a repair leaves the component "almost—but not quite—new." The inspection action, will remain as is. The intuition motivating these modifications is that an expert technician fixes periodically all broken components, which is sometimes named *age repair* or *age replacement* and is also related to *block replacement* [13], [21]. We call the new HVAC FMT: HVAC-1.

We select the maintenance strategy, corresponding to M_3 in Table 3.3, in which inspections, repair checks, and overhauls, take place every half, two, and fifteen years respectively. Using this policy we compare HVAC-0 vs HVAC-1, by computing the new KPI metrics for each. We depict the results in Figure 3.21, which shows

un-availability is 2 to 3 times higher than in the HVAC-0; other metrics are much less affected.

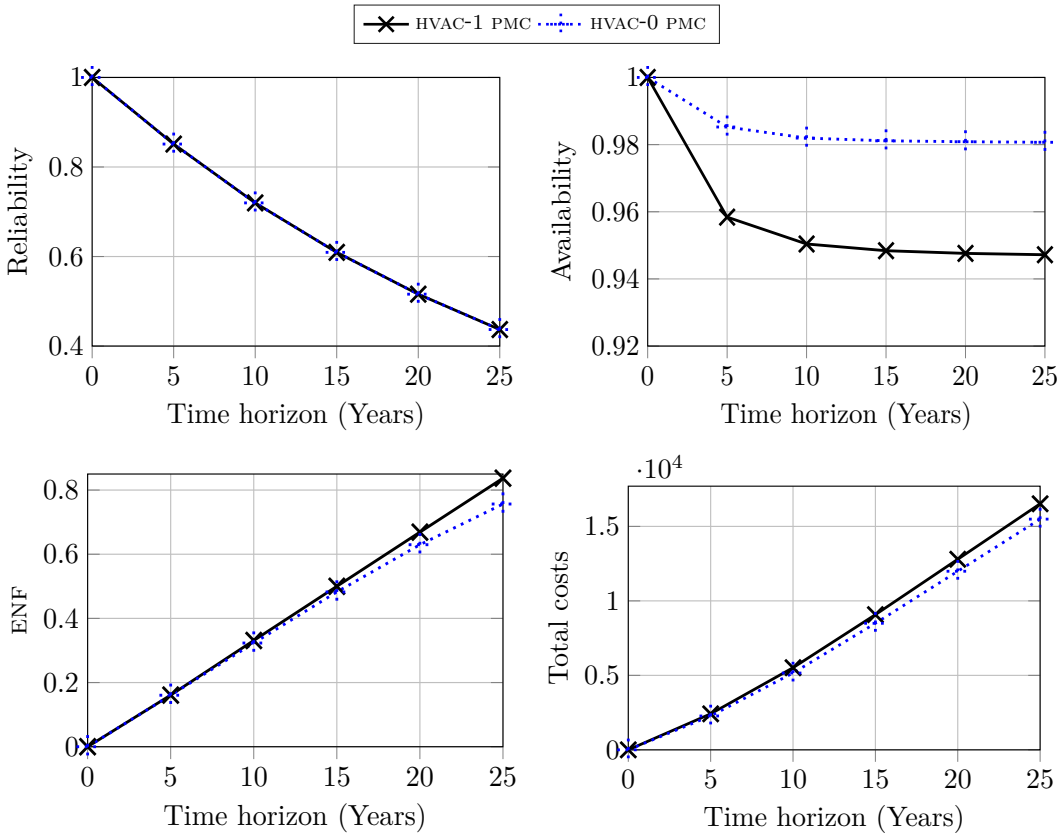


Figure 3.21: Comparing the KPIs between HVAC-0 and HVAC-1.

3.3.2 Comparison with statistical model checking

We perform a comparison with SMC and employ the tool UPPAAL to encode the same FMT HVAC, where the FMT components are modelled using PTA. The equivalence of the models' behaviour developed in both tools, i.e. PRISM and UPPAAL, are extensively and exhaustively evaluated in [3]. Here we present the computational performance of SMC vs. PMC. We consider two maintenance strategies (i) M_3 and (ii) M_4 from Table 3.3. SMC experiments have been run in a cluster with AMD Opteron™ 4386 processors, each with access to 64 GB of dedicated DDR3 RAM. PMC experiments have been run in a cluster with Intel® Xeon® E5-2640 v3 processors, each with access to 64 GB of DDR4 RAM. The resulting execution times in seconds

of both techniques, for both maintenance policies and all KPIs, are listed in Table 3.4.

T	Metric	PMC		SMC		Metric	PMC		SMC	
		M_3	M_4	M_3	M_4		M_3	M_4	M_3	M_4
5	**Reliability**	0.6	0.8	6.3	6.1	**Availability**	1.0	1.1	7.9	13.1
10		1.6	1.2	14.4	17.0		2.3	2.1	15.5	25.4
15		2.0	1.7	17.4	29.0		3.2	3.5	24.9	35.8
20		2.3	2.9	19.5	38.6		4.4	4.1	31.2	43.2
25		3.0	3.2	17.9	46.2		5.5	3.4	31.6	52.2
5	**ENF**	1.1	1.2	7.9	12.9	**Total costs**	5.4	5.2	29.8	47.9
10		2.2	2.3	15.4	24.0		10.5	9.4	56.4	90.0
15		3.2	3.5	21.3	34.2		14.0	13.9	78.5	119.4
20		4.9	4.0	27.7	43.3		17.6	17.1	94.7	145.6
25		5.6	4.4	31.6	50.1		22.1	19.7	105.7	168.3

Table 3.4: Time in seconds to compute KPIs using PMC and SMC.

In the case of PMC, the times listed include the computation of the MTTF used to perform the IE-to-EBE reduction, plus the subsequent time it took PRISM to compute the corresponding KPI from the resulting reduced model. Since the process of plugging in the computed MTTF into the reduced CTMC (from which PRISM computes the metric) is not yet automated, that is omitted from our measurements. We note however that, if the task were fully automated, such time should be negligible with respect to the "true analysis times" required to compute a KPI value.

Table 3.4 shows that, as expected, PMC converges to a result in significantly less time than SMC. For reliability, availability, and ENF, PMC is usually $\approx 10\times$ faster than SMC; for costs computation the difference is $\approx 6\times$ in favour of PMC. In general, this is a consequence of SMC requiring a sufficiently large random sample (viz. number of simulations) to achieve the desired statistical criteria and build the confidence interval. Here, the underlying trade-off is in runtime vs. computer memory: for PMC a reduction technique must be used to alleviate this problem, losing in exchange some precision in the metrics. On the other hand, it must be highlighted that the confidence criteria requested for our SMC analyses are standard but not particularly challenging. Requesting 99% confidence level and more precise (narrower) confidence intervals would greatly increase the computation times of SMC.

For both techniques and all metrics, computation times increase linearly with the time horizon analysed. However, PMC is not greatly affected by the maintenance policy studied, because its computational times are mostly a function of the structure of the underlying model. This structure is given by the set of (CTMC) modules and their communication mechanisms, rather than by the particular parameters like e.g. inspection frequency. Such parameters do play a minor role, since they may influence the number of iterations required to determine convergence to a fixed point in the internal mechanisms of the model checking algorithms. Incidentally, the effect of these parameters in the number of iterations could be counter-intuitive: ENF, availability, and costs computations with PMC are faster when using strategy M_4 than M_3, see Table 3.4. In any case, PMC is much less affected by the maintenance policy than SMC. Since SMC uses discrete event simulation as its backbone, the more events per time unit that need to be attended, the more computation time it takes to advance one-time unit. On average, using M_4 generates twice as many events per time unit than M_3, and therefore it takes longer to compute the metrics for M_4. Consequently, *quantifying* how longer will SMC analyses take for the M_4 is not straightforward. It is nevertheless clear that M_3 is faster to analyse via SMC than M_4 , which is corroborated in practically all time measurements presented in Table 3.4.

3.4 Conclusion

In this chapter, we have shown how we can obtain high-level certification via KPI metrics for a large subcomponent of BAS: the HVAC. We provide a new quantitative method for selecting optimal preventative maintenance strategies via the analysis of FMTs and PMC. We describe novel semantics and syntax details of the FMT which take the form of CTMCs and show how the state-space explosion can be mitigated via a novel state-space reduction algorithm. We further perform a comparison with the traditional SMC method of analysing FMTs.

There is a profuse of extensions that can be derived from the work. We are planning to extend our research in the following directions:

- use continuous stochastic processes [3] to represent the EBE. This will make the modelling framework more realistic and will allow us to reframe the semantics of FMTs into a stochastic hybrid system.
- extend the framework such that optimal policies can be synthesised and thus perform predictive maintenance.
- currently in the PMC community work is being done towards extending quantitative model checking to PTA [97]. Analysing PTA via PMC would reduce the design effort required: by using PTA over CTMC we would no longer need to perform the state-space reduction, and thus the accuracy of the computed values should improve, at the same time maintaining the high computational performance of PMC.
- validate the results computed for our models against KPI metrics obtained from real data. Note however that, due to the slow degradation rate of HVAC components, obtaining the data required to perform the last task could be particularly involved. To circumvent this, an HVAC fault simulation platform such as HVACSIM+ can be employed to analyse different fault conditions [108].

All the code for modelling FMT via PMC is available online at:

`www.gitlab.com/natchi92/fmt`

4

Framework for Stochastic Processes

Contents

We have examined how reasoning about the overall performance of BAS can be performed to generate high-level certificates in Chapter 3. In this chapter, we reduce the level of abstraction and reason at the level of individual BAS components making up the system to achieve low-level certification. We present a library of models that serves as an aid in understanding the dynamical evolution of the sub-components making up the BAS and as a benchmark for SHS.

In literature, a gamut of modelling frameworks are employed to capture the dynamics of buildings, together with their sub-components, from which three main

categories can be identified: *black-box*, *grey-box* and *white-box* models [7], [58]. Black-box modelling identifies model parameters automatically based on the relationships between input and output measurements. Their advantage is the establishment of models with very low development costs [56]. Black box models have been applied towards the understanding of the behaviour of BAS systems [110], [121], simulation and prediction of BAS systems [70], [132] and in conducting fault detection diagnosis [166]. Grey-box models are generated through the combination of insight based on the physical counterparts of the system and on the relationship between input and output measurements [56]. The most common grey-box modelling framework is the construction of state-space models based on resistance-capacitance (RC) network, which represents the different components and thermal energy flows within a building. The RC network is analogous to electrical circuitry and describes the underlying system process dynamics. This framework has been used to model multiple rooms in industrial buildings [15], [123], in large research projects such as the work conducted at ETH ZURICH [152] and at UC BERKELEY [112]. White-box models take into account all physical equations and principles that govern a system. They provide high-performance when compared to the grey- or black-box alternatives, however, come at a high-cost in terms of both computational complexity and implementation [56]. Such models are typically developed using well-established modelling languages, such as MODELICA [118], or software packages such as ENERGYPLUS [44] and BRCM [151]. For instance, [49] employed ENERGYPLUS to build a high-resolution building model using the state-space method, [45] constructed a building model consisting of 27 zones, hydraulic components and air conditioners using MODELICA, while BRCM has been employed to construct buildings heating ventilation and conditioning models for advanced controller design in [62].

The large spectrum of building models possess one common element: the models are merely an approximation of reality and therefore invariably contain uncertainty (inaccuracies) stemming from the mathematical model itself and the sensor measurements used to identify and make predictions on [120]. Consequently, uncertainty plays a crucial role in a model's ability to capture the behaviour of

the BAS. In order to capture such disturbances a framework based on *stochastic processes* is a necessity.

> **Stochastic process**: A collection of random variables $\mathbf{X} = \{\mathbf{X}[k] : k \in K\}$ defined on a common probability space, taking values in a common set D (the state space), and indexed by a set K, often either k or $[0, \infty)$ and thought of as time (discrete or continuous respectively) [136].

Buildings can be seen to be composed of an interleaving of continuous and discrete components (coming from example, the ON/OFF demand and from the continuous supply of energy). A rich mathematical modelling framework that is able to make use of stochastic processes and capture the hybrid element between the continuous and discrete components is known as *Stochastic Hybrid Systems* (SHS) [84]. Fostered by their application in multiple domains studies in SHS has recently flourished and has witnessed advances in areas of Systems and Control [29], [144], Formal Verification [2], [36], [107] and Cyber-Physical Systems [111]. In this chapter, we demonstrate how SHS can also be applied within the domain of BAS and identify verification and synthesis goals.

This chapter is structured as followed: Section 4.1 introduces the semantics of SHS. We present a library of models for BAS in Section 4.2. Section 4.3 delineates examples of SHS models constructed using the library of models and identifies different verification and synthesis requirements that align with the high-level certificates defined in Chapter 2.

4.1 Stochastic hybrid systems

SHS is a formal mathematical model that describes the relationship between discrete features, continuous dynamics and probabilistic uncertainty of a system. It consists of the discrete probabilistic jumps and continuous variables which evolve via stochastic differential equations within each discrete location. Given the generality and broadness of SHS, different definitions can be found across primary literature (the key difference is attributed to the means by which stochasticity enters within

the dynamics). A classification of the different definitions found in literature is given in [111]. In this work, we focus on discrete-time SHS and follow the definition in [5], which we extend to include labels.

Definition 2 (Stochastic Hybrid System (SHS))**.** *A* SHS *is a discrete-time model defined as the tuple*

$$\mathcal{H} = (\mathcal{Q}, n, \mathcal{U}, T_q, T_x, \Theta, L), \quad \textit{where} \tag{4.1}$$

- $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$, $m \in \mathbb{N}$, *represents a finite set of modes (locations);*
- $n \in \mathbb{N}$ *is the dimension of the continuous space* $\mathbb{R}^n$ *of each mode; the hybrid state space is then given by* $\mathcal{D} = \cup_{q \in \mathcal{Q}} \{q\} \times \mathbb{R}^n$*;*
- $\mathcal{U}$ *is a continuous set of actions, e.g.* $\mathbb{R}^v$*;*
- $T_q : \mathcal{Q} \times \mathcal{D} \times \mathcal{U} \to [0, 1]$ *is a discrete stochastic kernel on* $\mathcal{Q}$ *given* $\mathcal{D} \times \mathcal{U}$*, which assigns to each* $d = (q, x) \in \mathcal{D}$ *and* $u \in \mathcal{U}$*, a probability distribution over* $\mathcal{Q} : T_q(\cdot|d, u)$*;*
- $T_x : \mathcal{B}(\mathbb{R}^n) \times \mathcal{D} \times \mathcal{U} \to [0, 1]$ *is a Borel-measurable (continuous) stochastic kernel on* $\mathbb{R}^n$ *given* $\mathcal{D} \times \mathcal{U}$*, which assigns to each* $d \in \mathcal{D}$ *and* $u \in \mathcal{U}$*, a probability measure on the Borel space* $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n)) : T_x(\cdot|d, u)$*;*
- $\Theta = \{\theta_1, \ldots, \theta_n\}$ *is a set of atomic propositions;*
- $\mathcal{L} : \mathcal{Q} \to 2^{\Theta}$ *is a labelling function that assigns to each hybrid state possibly several elements of* Θ*.*

In this model, the discrete component takes values in a finite set $\mathcal{Q}$ of modes, each endowed with a continuous domain (the Euclidean space $\mathbb{R}^n$). As such, a point d over the hybrid state space $\mathcal{D}$ is the pair (q, x), where $q \in \mathcal{Q}$ and $x \in \mathbb{R}^n$. The semantics of transitions at any point, over a discrete-time domain, are as follows: given a point $d \in \mathcal{D}$, the discrete state is chosen from T_q, and depending on the selected mode $q \in \mathcal{Q}$ the continuous state is updated according to the probabilistic law T_x. The evolution of $\mathcal{H}$ for $k \leq K \in \mathbb{N}$ is a stochastic process $\mathbf{d}[k] = (q[k], x[k])$

with values in $\mathcal{H}$ and is defined on the probability space $(\mathcal{D}^{K+1}, \mathcal{B}(\mathcal{D}^{K+1}), \mathrm{P})$ where $\mathcal{B}(\mathcal{D}^{K+1})$ is the product sigma-algebra on the product space $\mathcal{D}^{K+1}$ and P is a probability measure. Non-determinism in the form of actions can also affect both discrete and continuous transitions. Similar to [5], we assume the continuous state is affected by Gaussian noise. This is a reasonable assumption given the nature of the underlying dynamics making up the BAS.

Remark 2 (Continuous transition kernel)**.** *T_x is characterised by the evolution of x i.e the continuous component of $\mathcal{H}$ and is described by a stochastic difference equation*

$$x[k+1] = F(q[k], x[k], u[k]) + G(q[k])w, \qquad (4.2)$$
$$u \in \mathcal{U}, \quad w \sim \mathcal{N}(0, \Sigma_w),$$

where F is a vector valued function and $G = \{G(q[k]) \in \mathbb{R}^{n\times r} \mid q \in \mathcal{Q}\}$ is a collection of diffusion terms, $w \in \mathbb{R}^r$ is a Gaussian noise with zero mean and covariance matrix $\Sigma_w \in \mathbb{R}^{r\times r}$. This characterises the transition kernel such that for any measurable set $\mathcal{B} \subseteq \mathbb{R}^n$, $x \in \mathbb{R}^n$, and $u \in \mathcal{U}$

$$T_x(\mathcal{B} \mid q, x, u) = \int_{\mathcal{B}} \mathcal{N}(t \mid F(q, x, u), G(q)\Sigma_w G(q)^T)\, dt. \qquad (4.3)$$

Then, it holds that P *is uniquely defined by T_x, and for $k < \infty$,*

$$T_x(\mathcal{B} \mid q[k], x[k], u[k]) = \mathrm{P}(x[k+1] \in \mathcal{B} \mid q[k], x[k], u[k]). \qquad (4.4)$$

We note that, for $\kappa = \infty$, P *is still uniquely defined by T_x by the* Ionescu-Tulcea extension theorem *[6].* □

Remark 3 (Special instance)**.** *Actions maybe associated to a deterministic selection of locations, namely $T_q : \mathcal{U} \to \mathcal{Q}$ and $\mathcal{U}$ is a finite set of actions.* □

In the presence of control actions, we further define a control strategy (policy) that maps actions $u \in \mathcal{U}$ to the $d \in \mathcal{H}$. A plethora of strategies exist in literature [16]. Here, we focus on *memoryless Markov* and *switching* strategies.

Memoryless Markov strategy: A sequence $\pi = (\pi_0, \pi_1, \dots)$ of universally measurable maps which assign $\pi_k : d \to \mathcal{U}$ for $k = \{0, 1, \dots\}$. We denote the set of all Markov strategies as Π [146].

We employ switching strategies for the special instance of SHS models (see Remark 3). Switching strategies reason over paths generated by the SHS. For $k \in \mathbb{N} \cup \{\infty\}$, we call $\text{Path}^k_{\mathcal{H}}(s) : \{0, 1, \dots, k\} \to \mathcal{D}$ the set of sample paths of $\mathbf{d}$ of length k. The set of all sample paths with finite and infinite lengths are denoted by $\text{Path}^{\text{fin}}_{\mathcal{H}}(s)$ and $\text{Path}_{\mathcal{H}}(s)$. We denote by $\omega_{\mathcal{H}}$, $\omega^k_{\mathcal{H}}$, and $\omega_{\mathcal{H}}(i)$ a sample path, a sample path of length k, and the $(i+1)$-th state on the path $\omega_{\mathcal{H}}$ of $\mathcal{H}$, respectively. Furthermore, we define the (observation) trace of path $\omega^k_{\mathcal{H}} = d_0 d_1 \dots d_k$ to be $\xi = \xi_0 \xi_1 \dots \xi_k$, where $\xi_i = \mathcal{L}(d_i) \in 2^{\Theta}$ for all $i \leq k$. For a path $\omega_{\mathcal{H}} \in \text{Path}_{\mathcal{H}}(s)$ with infinite length, we obtain an infinite-length trace.

Switching strategy: A function $\pi_{\mathcal{H}} : \text{Path}^{\text{fin}}_{\mathcal{H}}(s) \to \mathcal{U}$ that assigns a discrete mode $u \in \mathcal{U}$ to a finite path $\omega_{\mathcal{H}}$ of the process $\mathbf{d}$. The set of all switching strategies is denoted by $\Pi_{\mathcal{H}}$ [102].

4.1.1 Simulation of stochastic hybrid systems

An execution of a $\mathcal{H} = (\mathcal{Q}, n, \mathcal{U}, T_q, T_x, \Theta, \mathcal{L})$ under a memoryless Markov (policy) strategy is a stochastic process whose sample paths for the duration of $k \in \mathbb{N}$ are obtained according to Algorithm 2. In the presence of a switching strategy, we replace line 4 in Algorithm 2, with $u[k] \to \pi_{\mathcal{H}}(\omega^k_{\mathcal{H}})$, and following each state update (see Algorithm 2 line 6) extend $\omega^k_{\mathcal{H}}$ by $(u[k], x[k+1])$.

To reason about the behaviour of the SHS, Monte Carlo techniques need to be adopted. Monte Carlo techniques generate numerical sampled trajectories (sampled executions) representing the evaluation of a stochastic process over a predetermined time horizon. Trajectories are generated by sampling, at each point in time, from the current distribution representing the underlying system dynamics (see Algorithm 2 lines 5 - 6). Given a sufficient number of trajectories, one can approximate the statistical properties of the solution process with a required confidence level. This

Algorithm 2 Algorithm describing the execution of SHS over K time steps.

procedure SIMULATE($\mathcal{H}$)
  set $\mathbf{d}[0] \rightarrow (q[0], x[0])$ and $k = 0$;
  while k < K **do**
    set $u[k] \rightarrow \pi(d[k])$;
    extract from $\mathcal{Q}$ a value $q[k+1]$ according to $T_q(.|\mathbf{d}[k], u[k])$;
    extract from a value $x[k+1]$ according to $T_x(.|\mathbf{d}[k], u[k])$;
    $\mathbf{d}[k+1] \rightarrow (q[k+1], x[k+1])$
    $k \rightarrow k+1$;
  end while
end procedure

approach has been adopted for simulation of different types of SHS. [93] applies sequential Monte Carlo simulation to SHS to reason about rare-event probabilities. [53] performs Monte Carlo simulations of classes of SHS described as Petri nets. [31] proposes a methodology for efficient Monte Carlo simulations of continuous-time SHS.

4.1.2 Formalisation of low level certificates

We are interested in performing analysis on SHS over the low-level certificates defined in Section 2.1. Here we show how the low-level certificates can be formalised using temporal logical operators (CSLTL and BLTL) and evaluated over the probabilistic operator $\mathrm{P}_{=?}$ and sample space $\mathbb{X}$.

1. *Safety*: This is also referred to as the probabilistic invariance property. For finite time horizon, we employ BLTL and define safety as

$$\mathrm{P}_{=?}(\forall j \in \mathbb{N}_{\leq K} : x[j] \in Z) = \mathrm{P}_{=?}(\mathbf{G}^{\leq K}\ Z), \tag{4.5}$$

   for a given time horizon K, realisations of $\{x[k] | k \in \mathbb{N}_{\leq K}\}$ and safe set $Z \in \mathcal{B}(\mathbb{X})$. Conversely for infinite horizon safety, we employ CSLTL and define safety as

$$\mathrm{P}_{=?}(\forall j \in \mathbb{N} : x[j] \in Z) = 1 - \mathrm{P}_{=?}\left(\texttt{true}\ \mathbf{U}\ \neg Z\right). \tag{4.6}$$

2. *Reach*: This corresponds to the probabilistic reachability property for a given target set $T \in \mathcal{B}(\mathbb{X})$, over a finite time horizon K. It can be expressed via

BLTL as

$$\mathrm{P}_{=?}(\exists i \in \mathbb{N}_{\leq K} : x[i] \in T) = \mathrm{P}_{=?}\left(\mathbf{F}^{\leq K}\ T\right), \tag{4.7}$$

for the realisation $\{x[k] | k \in \mathbb{N}_{\leq K}\}$.

3. *Reach and Avoid*: This corresponds to the probabilistic reach-avoid property and quantifies the probability of reaching T while staying in Z. For finite properties this is expressed by probabilistically evaluating the BLTL formula

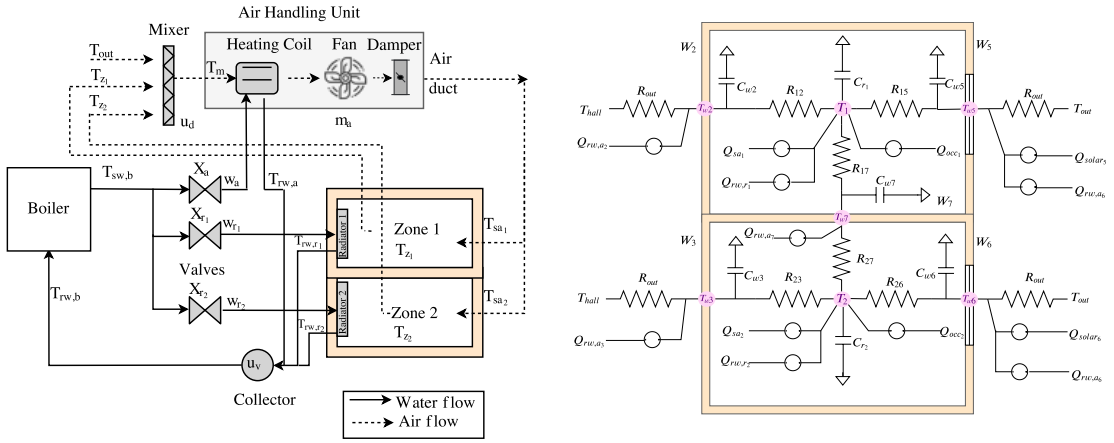
$$\mathrm{P}_{=?}(\exists i \in \mathbb{N}_{\leq K} : x[i] \in T \wedge \forall j \in \mathbb{N}_{\leq i-1} : x[j] \in Z) = \mathrm{P}_{=?}\left(\neg Z\ \mathbf{U}^{\leq K}\ T\right). \tag{4.8}$$

The complement of Z is the *unsafe* set that is avoided in the probabilistic reach-avoid property. Similarly, for infinite horizon properties this is expressed using CSLTL formula as,

$$\mathrm{P}_{=?}(\exists i \in \mathbb{N} : x[i] \in T \wedge \forall j \in \mathbb{N}_{\leq i-1} : x[j] \in Z) = \mathrm{P}_{=?}\left(\neg Z\ \mathbf{U}\ T\right). \tag{4.9}$$

4.2 Library of models for building automation systems

In this section, we present a library of models for BAS that is inspired by and built around our Smart Buildings Laboratory. Focus is on modelling both temperature and carbon dioxide (CO_2) dynamics - key elements for ensuring thermal comfort and good air quality. The library facilitates the construction of models with different configurations and features: for instance, we can build low- to high-dimensional models, with discrete or continuous inputs and states. The models are endowed with stochasticity which can represent un-modelled components, unknown parameters, random continuous effects, or likelihoods of transitioning to a certain discrete state from the current location. Using this library of models, SHS models of varying complexities with different analysis goals can be easily constructed and tested.



(a) Two-zone boiler-based heating system with air handling unit and radiators.

(b) Resistance-capacitance circuit for the thermal dynamics within the two zones.

Figure 4.1: Building automation system (BAS) setup.

4.2.1 BAS: Structure and components

BAS models clearly depend on the size and topology of the building and on its climate control setup. In this work, we consider the BAS setup shown in Figure 4.1a, which consists of two teaching rooms that are connected to a boiler-heated system. The boiler supplies heat to the heating coil within the AHU and to two radiators. Valves control the rate of water flow within the heating coils and the radiators. The AHU supplies air to the two zones that are connected back to back and adjacent both to the outside and to an interior hall (see Figure 4.1a). The zone air of both rooms can mix with the outside air and exchanges circulating air with the AHU. The zone air is further affected by the presence or absence of occupants within the zone. Occupants generate both CO_2 and thermal energy which directly impacts the thermal evolution and air quality in the zone. The combination of the mixer, damper, fan and opening or closing of windows and doors allows for regulation of the zone air. Return water from the AHU heating coils and radiators is collected and pumped back to the boiler.

Figure 4.1b presents the RC network circuit of the two zones, which underpins the dynamics for temperature in the zone component - corresponding equations are in Table 4.3. The heat level in each room is modified by (i) radiative solar energy

absorbed through the walls, (ii) occupants, (iii) AHU input supply air, (iv) radiators and (v) AHU return water. The effect of heat stored in the walls and in rooms is depicted with capacitors, whereas thermal resistance to heat transfer by the walls is depicted by resistor elements.

4.2.2 BAS: Dynamics and configurations

We define models for the individual components in the BAS system. Single components are intended as separate physical structures within the BAS. Their models are built from the underlying physics and are improved via industrial feedback and from existing literature [71]. We obtain models with a number of unknown parameters: these are estimated and validated using data collected from the BAS setup [92].

We list indices in Table 4.1, while all the quantities (variables, parameters, inputs) are listed in Table 4.2. Table 4.3 presents all the relations among variables in the model: algebraic relations define static couplings whereas differential relations define the dynamics for the corresponding variables. The structure in Figure 4.1a, the quantities in Table 4.2 and the variables (with associated dynamics) in Table 4.3, together allow to construct global models for the complete BAS setup. We refer to the set of models describing the individual components (cf. Table 4.3) as a library of models. One can select the individual components and models from the library, and build different BAS configurations. One can also use the same modelling principles to consider more complex BAS structures for e.g. buildings with large number of rooms or multiple heating elements.

A global model of the BAS set-up can be complex, comprising both algebraic and differential relations that are further affected by process noise. A model also contains several inputs which can either be construed as control signals or as exogenous signals. Some of the dynamics are non-linear in view of continuous variables that are bi-linearly coupled (cf. AHU damper model in Table 4.3). Furthermore, the model features multiple components that present switching discrete behaviours, effecting the dynamics of the continuous variables. In order to tackle the complexity of global

Index	Reference	Index	Reference	Index	Reference
a	AHU	adj	adjacent zone	b	boiler
c	window closed	d	mixer	e	zone unoccupied
f	zone occupied	h	water	$hall$	hallway
$i \in \{1, 2\}$	individual zones	$jn \in \{2, 3, 7\}$	zone walls with no windows	$jw \in \{5, 6\}$	zone walls with windows
$j \in \{jn \cup jw\}$	all zone walls	$l \in \{1, 2\}$	adjacent interior zone	o	window open
occ	occupants	out	outside	r	radiator
ref	reference	rw	return water	sa	supply air
$solar$	solar energy	sp	set point	sw	supply water
v	collector	w	wall	z	zone

Table 4.1: Indices.

Symbol	Quantity	Type	Symbol	Quantity	Type
A_i	area of windows of each zone	constant	B_{en}	boiler switch	discrete
C	capacitance	constant	C_{pa}, C_{pw}	specific heat capacity of air and water	constant
CO_2	carbon-dioxide measurements	input	F_{en}	fan switch	discrete
k_b	steady-state of the boiler	constant	m	mass air flow rate	input
n	number of zones	constant	P_{out}	radiator rated power output	constant\input
Q	heat gain	input	R	thermal resistance to heat from walls	constant
T	temperature	state\input	u	mixing ratio	input
(UA)	overall transmittance factor of	constant	V	volume	constant
w	water flow rate	input	w_{max}	maximum water-flow permitted by the valve	constant
W_{en}	windows & doors switch	discrete	X	valve position	input
$\{\alpha, \beta, \mu\}$	de-rating and offset factors	constants	σ	process noise	constant
ρ	density	constant	τ	time constant	constant
ϱ	natural air drift	constant			

Table 4.2: List of variables, inputs, and parameters.

BAS models and to add a level of flexibility to the modelling framework, we consider each BAS component as a separate module, characterised by inputs and output elements, and by internal variables. We make use of individual modules describing the component type and then connect different modules based on possible physical couplings. Coupling is also achieved via input-output relationships for e.g., in the zone module we have coupling between two zones through the continuous variable $T_{adj,l}$ corresponding to the adjacent zones, which for the wall separating the two zones (cf. W_7 in Figure 4.1b) corresponds to the individual zone temperatures of the two-zone modules (cf. Table 4.3 zone equations). Having such a modular structure for the individual components, provides an added level of versatility since we can connect different components to create various new models. Modularisation also allows (i) to perform analysis of the whole setup by executing analysis of individual modules and (ii) to extend the library of models by defining new modules that connect to existing modules via their input-output relations.

Component	Continuous variables	Relation
Boiler	$dT_{sw,b}(t) = \begin{cases} 0 & B_{en}(t) = 0 \\ (\tau_{sw})^{-1}\left[(-T_{sw,b}(t) + k_b)dt\right] + \sigma_{sw}dW & B_{en}(t) = 1 \end{cases}$	differential
Valve	$w(t) = (\tau)^{-1}\left[\exp(\ln(\tau)X(t))w_{max}\right]$	algebraic
Mixer	$T_d(t) = u_d T_{out}(t) + (1 - u_d)(\sum_i T_{z_i}(t))(n)^{-1}$	algebraic
AHU heating coil	$dT_{rw,a}(t) = (C_{pw}\rho_h V_a)^{-1}\left[(C_{pw}w_a(t)(T_{sw,b}(t) - T_{rw,a}(t)) + (UA)_a(T_d(t) - T_{rw,a}(t)))dt\right] + \sigma_{rw,a}dW$	differential
AHU damper	$dT_{sa_i}(t) = (C_a\rho_a V_a)^{-1}\left[m_a(t)C_{pa}(T_d(t) - T_{sa_i}(t)) + (UA)_a(T_{z_i}(t) - T_{sa_i}(t))\right]dt + \sigma_{sa_i}dW \quad \text{open}$	differential
Radiator	$dT_{rw,r_i}(t) = (C_{pw}\rho_h V_{r_i})^{-1}\left[(C_{pw}w_{r_i}(t)(T_{sw,b}(t) - T_{rw,r_i}(t)) + (UA)_{r_i}(T_{z_i}(t) - T_{rw,r_i}(t)))dt\right] + \sigma_{rw,r_i}dW$	differential
AHU fan	$m_a(t) = \begin{cases} m_{sa}(t) & F_{en}(t) = 0 \\ 0 & F_{en}(t) = 1 \end{cases}$	algebriac
Window & doors	$\varrho(t) = \begin{cases} \varrho_o & W_{en}(t) = 0 \\ \varrho_c & W_{en}(t) = 1 \end{cases}$	discrete
Zone	$dT_{z_i}(t) = (C_{z_i})^{-1}\left[\frac{T_{w_{jn}}(t) - T_{z_i}(t)}{R_{ij}} + Q_{rw,r_i}(t) + Q_{occ_i}(t) + Q_{sa_i}(t)\right]dt + \sigma_{z_i}dW$ $dT_{w_{jn}}(t) = \left(C_{w_{jn}}\right)^{-1}\left[\frac{T_{adj,out}(t) - T_{z_i}(t)}{R_{out}} + \sum_l \frac{T_{adj_l}(t) - T_{w_{jn}}(t)}{R_{lj}} + Q_{rw,a_{jn}}(t)\right]dt + \sigma_{w_{jn}}dW$ $dT_{w_{jw}}(t) = \left(C_{w_{jw}}\right)^{-1}\left[\frac{T_{adj,out}(t) - T_{z_i}(t)}{R_{out}} + \sum_l \frac{T_{adj_l}(t) - T_{w_{jw}}(t)}{R_{ljw}} + Q_{solar_{jw}}(t) + Q_{rw,a_{jw}}(t)\right]dt + \sigma_{w_{jw}}dW$ $dCO_{2_{z_i}}(t) = (V_{z,i})^{-1}\left[-m_a(t)CO_{2_{z_i}}(t) + \varrho(t)(CO_{2,out} - CO_{2_{z_i}}(t)) + CO_{2,occ}(t)\right]dt + \sigma_{2_{z_i}}dW$ $Q_{rw,r_i}(t) = P_{rad_i}(\alpha_2(T_{rw,r_i}(t) - T_{z_i}(t)) + \alpha_1), \qquad Q_{occ_i}(t) = \mu_i(CO_{2_i}(t)) + \beta_{1_i}$ $Q_{sa_i}(t) = m_a(t)C_{pa}(T_{sa_i}(t) - T_{z_i}(t))), \qquad Q_{rw,a_j}(t) = \alpha_3(T_{rw,a}(t) - T_{w_j}(t)),$ $Q_{solar_{jw}}(t) = (\alpha_0 A_i T_{out}(t) + \beta_2)$	differential
Collector	$T_{rw,b}(t) = u_v T_{rw,a}(t) + (1 - u_v)(\sum_i T_{rw,r_i}(t))(n)^{-1}$	algebraic

Table 4.3: Dynamics and functional relations among component variables.

4.2.3 BAS: Description of model library

The library of BAS components comes in the form of MATLAB scripts. Each script represents an individual BAS component. The models are in state-space form and are of two types: linear or bilinear depending on the component they represent. They are defined using the MATLAB symbolic toolbox, are parametrised and can be described both in discrete and continuous time. We provide the estimated parameters, from data gathered from the Smart Buildings laboratory, to construct the individual models. Users can easily input their own parameters and construct their own model. Different components can be connected together based on their input-output relations by cascading the different symbolic models for each component. The

scripts are available online at:

`www.gitlab.com/natchi92/BASBenchmarks`

4.3 Case studies

We set up three case studies, which trade-off different levels of complexity. For each case study, we map the constructed models into the SHS modelling framework and introduce requirements (expressing example low-level certification goals).

4.3.1 Two zone heating setup with stochastic dynamics

We consider two zones, each heated by one radiator and with a common air supply, as portrayed in Figure 4.2. From Table 4.3 we select two components and corresponding models: the radiator and the zone. The models are simplified by taking the following assumptions: (i) the wall temperature is constant across the zones and is a fixed value ($T_w = 22$ [$^\circ C$]), (ii) the boiler is switched ON providing a supply temperature ($T_{sw,b} = 75$ [$^\circ C$]), (iii) we fix both the mass air flow rate $m_{sa} = 9.6$ [m^3/min] and the radiator water flow rate $w_r = 0.12$ [m^3/min], and (iv) we assume a fixed occupancy heat gain ($Q_{occ_1} = 0.0016$ [$^\circ C/min$]) in each zone. The fixed occupancy gain is computed based on the steady state CO_2 levels in a teaching room [129] and the affine function Q_{occ_1} as described in Table 4.3.

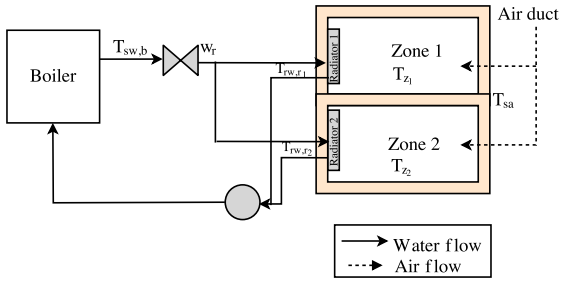


Figure 4.2: BAS setup for the first case study.

Consequently, the resulting model has the four continuous-state variables $x = (T_{z_1},\ T_{z_2},\ T_{rw,r_1},\ T_{rw,r_2})$ and with a common supply temperature $u = T_{sa}$ as an input. We discretise the dynamics using a Euler-Maruyama scheme having a uniform

sampling time $\Delta = 15$ $[min]$, to obtain a Gaussian perturbed linear discrete-time model. One should note that the model is not fully observable since the individual zone temperatures (variables of interest) are the only variables provided as outputs. The dynamics of the continuous variables are described by

$$\mathbf{M}_s : \begin{cases} x_s[k+1] & = A_s x_s[k] + B_s u_s[k] + Q_s + \Sigma_s w_s[k] \\ y_s[k] & = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x_s[k]. \end{cases} \tag{4.10}$$

The matrices A_s, B_s are properly sized and Q_s represents the constant additive term within the model and corresponds to

$$Q_s = \begin{bmatrix} \frac{T_w \Delta}{C_{z_1} R_1} & \frac{T_w \Delta}{C_{z_2} R_2} & \frac{C_{pw} w_{r_1} \Delta}{C_{pw} \rho_h V_{r_1}} T_{sw,b} & \frac{C_{pw} w_{r_2} \Delta}{C_{pw} \rho_h V_{r_2}} T_{sw,b} \end{bmatrix}^T.$$

$w_s^T = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix}^T$ are independent Gaussian random variables, which are independent of the initial condition of the process and correspond to white noise (i.e. $w_i \sim \mathcal{N}(0, \mathbf{I}), i = \{1, \ldots, 4\}$ iid.) and

$$\Sigma_s = diag([(\sqrt{\Delta}\sigma_{z_1})^2 \ (\sqrt{\Delta}\sigma_{z_2})^2 \ (\sqrt{\Delta}\sigma_{rw,r_1})^2 \ (\sqrt{\Delta}\sigma_{rw,r_2})^2]).$$

The system matrices are all given in Appendix A.1 and are constructed based on the models in Table 4.3.

Requirement

The ASHREA building's handbook [11] recommends that:

> Assuming slow air movement (less than $12.5 m/min$) and 50% indoor relative humidity, the operative temperatures recommended range from 20 $^\circ C$ and 24 $^\circ C$ in the winter.

We translate this recommendation into a verification framework and consequently we would like to check whether traces generated by the models remain within a specified safe set for a given time period of 1.5 hours. The safe set is described as an interval [20 24] $^\circ C$ for each zone and we constrain the input u to lie within the set

$\{T_{sa} \in \mathbb{R} | 15 \leq T_{sa} \leq 30\}$ (corresponding to physical constraints). This requirement corresponds to probabilistic safety defined using (4.5).

$$\text{Safety requirement } \varphi_1 := \mathrm{P}_{=?}\left(\mathbf{G}^{\leq K=6}\, X_{safe}\right)$$

where $K = 6 \times \Delta = 1.5$ hours and

$$X_{safe} = \begin{bmatrix} 20 & 24 \\ 20 & 24 \\ 33 & 37 \\ 33 & 37 \end{bmatrix}.$$

Note, the model already satisfies the slow air movement and relative humidity due to parameters used to construct the model. We also restrict the temperature deviation of the radiators to lie between [33 37] $°C$, in order to be able to define a closed safe set over the four continuous variables.

Mapping into a stochastic hybrid system

The constructed model defined using (4.10) can be mapped into SHS following Definition 2.

$$\mathcal{H}_1 = (\mathcal{Q}_1, n_1, \mathcal{U}_s, T_{q_1}, T_{x_s}, \Theta_1, \mathcal{L}_1) \tag{4.11}$$

with

- $\mathcal{Q}_1 = \{q_0\}$ since we only have one operational mode;
- the state variable x consists of four continuous variables, hence $n_1 = 4$;
- we have one continuous control input $\mathcal{U}_s = \mathbb{R}$;
- $T_{q_1} = 1$ given that we only have one discrete mode;
- the continuous stochastic kernel is derived from (4.10)

$$T_{x_s}(\cdot | x_s, u_s) = \mathcal{N}(\cdot | A_s x_s + B_s u_s + Q_s, \Sigma_s);$$

- $\Theta_1 = \{Safe\}$ corresponding to Θ_1 in φ_1;
- $\mathcal{L}_1$ is a labelling function which associates the $\theta_1 = Safe$ to regions within X_{safe}.

4.3.2 Two zone heating setup with large number of continuous variables

In this second case study, we focus on the dynamics of the zone component from Table 4.3 and consider the two zones shown in Figure 4.1b. We assume that (i) a central fan pumps in air in both rooms with a common supply temperature $15 \leq T_{sa} \leq 30$ [$^{\circ}C$], (ii) the input mass airflow is constant and a fixed value ($m_{sa} = 9.6$ $[m^3/min]$), (iii) the return water temperature of the AHU heating coils is constant ($T_{rw,a} = 35$ [$^{\circ}C$]) and (iv) the CO_2 generated in each zone is treated as external disturbance. As in the previous case study, the selected model is discretised using Euler-Maruyama with a sampling time $\Delta = 15$ minutes, to obtain the discrete-time model

$$\mathbf{M}_{s,7} : \begin{cases} x_s[k+1] & = A_{s,7}x_{s,7}[k] + B_{s,7}u_{s,7}[k] + G_{s,7}w_{s,7}[k] + Q_{s,7} \\ y_{s,7}[k] & = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x_{s,7}[k]. \end{cases} \tag{4.12}$$

Here the continuous state variables are $x_{s,7} = (T_{z_1}, T_{z_2}, T_{w_5}, T_{w_6}, T_{w_2}, T_{w_3}, T_{w_7})$ and a common fan supplies the two zones with a supply rate $u_{s,7} = T_{sa}$. Matrices $A_{s,7},\ B_{s,7},\ G_{s,7}$ are properly sized, $Q_{s,7}$ represents constant additive terms within the model and corresponds to

$$Q_{s,7} = \begin{bmatrix} \frac{q_{c_0}\Delta}{C_{z_1}} & \frac{q_{c_0}\Delta}{C_{z_2}} & \frac{q_{c_2}\Delta}{C_{w_5}} & \frac{q_{c_2}\Delta}{C_{w_6}} & \frac{q_{c_1}\Delta}{C_{w_2}} & \frac{q_{c_1}\Delta}{C_{w_3}} & \frac{q_{c_1}\Delta}{C_{w_7}} \end{bmatrix}^T,$$

with $q_{c_0} = \beta_{1_i} + \alpha_1 P_{rad_i}$, $q_{c_1} = \alpha_3 T_{rw,a}$ and $q_{c_2} = \alpha_0 A_2 \beta_2 + q_{c_1}$. Here, $w_{s,7}$ corresponds to the disturbance signals

$$w_{s,7} = \begin{bmatrix} T_{out} & T_{hall} & CO_{2_{z_1}} & CO_{2_{z_2}} & T_{rw,r_1} & T_{rw,r_2} \end{bmatrix}^T,$$

which are modelled as random external effects. At each time instant these disturbances are modelled as realisations of identical and independent distributions

(i.i.d.),

$$\begin{bmatrix} T_{out}[k] \\ T_{hall}[k] \\ CO_{2_{z_1}}[k] \\ CO_{2_{z_2}}[k] \\ T_{rw,r_1}[k] \\ T_{rw,r_2}[k] \end{bmatrix} \sim \mathcal{N}(w_m, \Sigma_w), \text{with mean } \; w_m = \begin{bmatrix} 9 \\ 15 \\ 500 \\ 500 \\ 35 \\ 35 \end{bmatrix}, \text{and variance}$$

$$\Sigma_w = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}. \tag{4.13}$$

Requirement

The ASHREA building's handbook [11] defines thermal comfort as the *"condition of mind which expresses satisfaction with the thermal environment."* The effect of thermal comfort on occupants' performance and a building's energy performance has been widely studied in literature and it is suggested that the performance of occupants decreases outside the thermal comfort zone [25], [119]. Consequently, for $\mathbf{M}_{s,7}$ we would like to synthesise a policy ensuring that the temperature within zone 1 does not deviate from the setpoint ($T_{sp} = 22$ [$^\circ C$]) by more than 1 [$^\circ C$] over a time horizon equal to 2.25 hours (i.e $K = 9$). This requirement can be translated into computing the optimal control actions that maximise the probability p of satisfying a BLTL formula according to (4.5).

$$\text{Synthesis requirement } \varphi_2 := \mathrm{P}_{\geq p}\left(\mathbf{G}^{\leq K=9} \; [|T_{sp} - T_{z_1}| \leq 1]\right)$$
$$\varphi_2 := \mathrm{P}_{\geq p}\left(\mathbf{G}^{\leq K=9} \; \mathit{Deviation}\right)$$

Mapping into a stochastic hybrid system

The constructed model defined using (4.10) can be mapped into SHS following Definition 2.

$$\mathcal{H}_2 = (\mathcal{Q}_2, n_2, \mathcal{U}_{s,7}, T_{q_2}, T_{x_{s,7}}, \Theta_2, \mathcal{L}_2) \tag{4.14}$$

with

- $\mathcal{Q}_2 = \{q_0\}$ since we only have one operational mode;
- the state variable $x_{s,7} \in \mathbb{R}^7$, hence $n_2 = 7$;
- we have one continuous control input $\mathcal{U}_{s,7} = \mathbb{R}^1$;
- $T_{q_2} = 1$ given that we only have one discrete mode;
- the continuous stochastic kernel is derived from (4.12)

  $$T_{x_{s,7}}(\cdot|x_{s,7}, u_{s,7}) = \mathcal{N}(\cdot|A_{s,7}x_{s,7} + B_{s,7}u_{s_c} + Q_{s,7} + G_{s,7}w_m, G_{s,7}\Sigma_w G_{s,7}^T);$$

- $\Theta_2 = \{Deviation\}$ corresponding to Θ_2 in φ_2;
- $\mathcal{L}_2$ is a labelling function which associates the $\theta_2 = Deviation$ to regions satisfying $|T_{sp} - T_{z_1}| \leq 1$.

4.3.3 Air quality model capturing CO_2 and temperature dynamics in zone

In this third case study we focus on the AHU fan, windows and doors and zone components from Table 4.3 and we select the AHU as the only source of heat within the zone (the boiler is disconnected). We also set the mixer as circulating air by pumping in the outside air ($u_d = 1$). To reason about the air quality of a zone, we model the dynamics of both the temperature and CO_2 levels both of which are affected by (i) pumped air by fan within the AHU and (ii) the opening or closing of the windows and doors in the zone. The BAS configuration is shown in Figure 4.3. This can be described as a SHS comprising two continuous variables over discrete modes in the set

$$\mathcal{Q}_3 = \{q_0 = (\neg F_{en}, \neg W_{en}), q_1 = (F_{en}, W_{en}), q_2 = (\neg F_{en}, W_{en}), q_3 = (F_{en}, \neg W_{en})\}$$

describing possible configurations of the zone (fan on (F_{en}) or off ($\neg F_{en}$), and windows and doors open (W_{en}) or closed ($\neg W_{en}$)). The continuous dynamics are

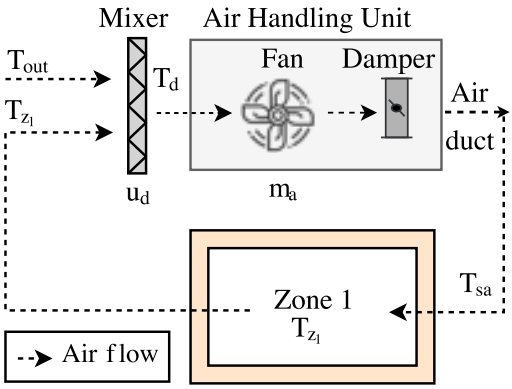


Figure 4.3: Graphical depiction of BAS setup for third case study.

built from Table 4.3 and evolve according to

$$
\begin{aligned}
dCO_{2_{z_1}}(t) &= (V_{z,1})^{-1}\left[-m_a(t)CO_{2_{z_1}}(t) + \varrho(t)(CO_{2,out} - CO_{2_{z_1}}(t)) + CO_{2,occ}(t)\right]dt \\
&\quad + \sigma_{2_{z_1}}dW, \\
dT_{z_1}(t) &= (C_{z_1})^{-1}\left[\frac{T_{w_{jn}}(t) - T_{z_1}(t)}{R_{ij}} + Q_{occ_1}(t) + Q_{sa_1}(t)\right]dt + \sigma_{z_1}dW, \\
Q_{occ_1}(t) &= \mu CO_{2_{z_1}}(t) + \beta_1, \\
Q_{sa_1} &= m_a(t)C_{pa}(T_{sa_1}(t) - T_{z_1}(t)) \\
\varrho(t) &= \begin{cases} \varrho_{op} & W_{en}(t) = 1 \\ \varrho_{cls} & W_{en}(t) = 0 \end{cases}, \quad m_a(t) = \begin{cases} m_{sa}(t) & F_{en}(t) = 1 \\ 0 & F_{en}(t) = 0 \end{cases}.
\end{aligned} \tag{4.15}
$$

From (4.15), one can note that depending on which discrete mode in the DTMC we are in, the continuous variable evolution changes. We discretise (4.15) and rewrite the evolution of continuous dynamics in state-space form, namely

$$
\mathbf{M}_{s,q_i} : \begin{cases} x_{s,q}[k+1] &= A_{s,q_i}x_{s,q}[k] + Q_{s,q_i} + G_{s,q_i}w_{s,q_i}[k] \\ y_{s,q_i}[k] &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_{s,q_i}[k]. \end{cases} \tag{4.16}
$$

The matrices A_{s,q_i} Q_{s,q_i} G_{s,q_i}, Σ_{s,q_i} $\forall i = \{0, 1, \ldots, 3\}$ are properly sized matrices whose values depend on the current discrete mode $q_i \in \mathcal{Q}_3$. The disturbance vector, at each time instance, is sampled from a Gaussian distribution described using $w_{s,q_i}[k] \sim \mathcal{N}(0, \Sigma_{s,q_i})$ and is also a function of the discrete mode. The exact matrix values are given in Appendix A.3.

Requirement

The LEED [117] requirements recommend that:

> Assuming the CO_2 sensors are measuring using intervals no longer than 30 minutes, the BAS must generate an alarm if the CO_2 levels in any zone exceed more than 15% of the corresponding minimum outdoor air concentration [117].

With the premise of ensuring good air quality within a zone, we define a synthesis problem. The requirement is now of synthesising an optimal policy that ensures the CO_2 levels do not exceed 15% of the minimum outdoor air concentration (400 [ppm] [129]), while the temperature remains within the comfort region ($20 - 24$ [$^\circ C$] (see Requirement in Section 4.3.1)). We translate the requirement into finding the maximal probability p of satisfying the reach-avoid property φ_3, which takes the shape of (4.9). It is designed to ensure CO_2 levels never exceed the 15% threshold, while maintaining the thermal requirements.

$$\text{Reach-avoid requirement } \varphi_3 := \mathrm{P}_{\geq p}\left(\neg X_{unsafe}\ \mathbf{U}^{\leq K=32}\ X_{target}\right)$$

where $K = 8 \times \Delta = 32$ steps,

$$X_{unsafe} = \begin{bmatrix} 450 & 460 \\ 20 & 24 \end{bmatrix}, \quad X_{target} = \begin{bmatrix} 400 & 450 \\ 20 & 24 \end{bmatrix}.$$

Note, we satisfy the assumption of "*Assuming the CO_2 sensors are measuring using intervals no longer than 30 minutes*" by employing a discrete-time model identified using five-minute sensor measurement readings and having a sampling time of 15 minutes.

Mapping into a stochastic hybrid system

The constructed model defined using (4.10) can be mapped into SHS following Definition 2 and Remark 3.

$$\mathcal{H}_3 = (\mathcal{Q}_3, n_3, \mathcal{U}_{s,q}, T_{q_3}, T_{x_{s,q}}, \Theta_3, \mathcal{L}_3) \tag{4.17}$$

with

- $\mathcal{Q}_3 = \{q_0 = (\neg F_{en}, \neg W_{en}), q_1 = (F_{en}, W_{en}), q_2 = (\neg F_{en}, W_{en}), q_3 = (F_{en}, \neg W_{en})\}$;
- the state variable $x_{s,q}$ is common to all discrete modes and consists of two continuous variables, hence $n_3 = 2$;
- we have four discrete control modes $\mathcal{U}_{s,q} = \mathcal{Q}_3$;
- the discrete stochastic kernel is $T_{q_3} : \mathcal{U}_{s,q} \to \mathcal{Q}_3$;
- the evolution of the continuous stochastic kernel within each discrete mode $q_i,\ i = \{0, 1, \ldots, 3\}$ is derived from

$$T_{x_{s,q}}(\cdot | x_{s,q}, q_i) = \mathcal{N}(\cdot | A_{s,q_i} x_{s,q},\ G_{s,q_i} \Sigma_{s,q_i} G_{s,q_i}^T);$$

- $\Theta_3 = \{\mathit{Unsafe},\ \mathit{Target}\}$ corresponding to Θ_3 in φ_3;
- $\mathcal{L}_3$ is a labelling function which associates the $\theta_0 = \mathit{Unsafe}$ to regions within X_{unsafe} and $\theta_1 = \mathit{Target}$ to regions within X_{target} .

4.4 Conclusion

This chapter sets up the formal modelling framework of SHS that will be used throughout this thesis. We further provide a library of models for BAS aimed at simplifying the modelling process for such systems. Finally, we construct three case studies with varying degrees of complexities and certification goals.

5

Formal Verification and Synthesis via Abstractions

Contents

In the previous chapter, we present the main modelling framework of SHS. Building upon this, here we set up a schema for performing verification and synthesis. Looking more closely into the semantics of SHS, we can note that SHS evolve over uncountable spaces. Consequently, directly applying formal verification and strategy synthesis is in general not decidable [5], [154]. To mitigate this impediment,

quantitative finite abstractions into Markov processes are required. These precise approximations come in two main different flavours: abstractions into DTMC or MDP [5], [147] and into IMDP [75], [102]. Both approximations require the definition of a compact set over which analysis is performed. This is a consequence of partitioning the state-space to generate the underlying abstraction. Once the finite abstractions are obtained formal verification or strategy synthesis can be performed.

We divide this chapter into four main sections. Section 5.1 presents an overview of the classical approach of abstracting SHS into MDPs and delineates how verification and policy synthesis is performed. Section 5.2 extends the work in [102] to cater for multiple discrete modes and presents a novel framework for performing verification and policy synthesis of SHS using IMDP. Section 5.3 presents a numerical comparison between the two abstraction methods; while Section 5.4 solves the BAS case studies presented in the previous chapter.

5.1 Abstractions into Markov Decision Processes

Building on the seminal work in [5], which describes a method for approximate model checking of SHS with provable guarantees, abstractions of SHS into MDPs has been formalised in [146] and implemented within the tool FAUST^2 [147]. Given a SHS $\mathcal{H} = (\mathcal{Q}, n, \mathcal{U}, T_q, T_x, \Theta, \mathcal{L})$, we abstract $\mathcal{H}$ into a Markov process $\mathcal{M}_{\text{MDP}} = (\mathcal{S}, \mathcal{A}, T_s, \bar{\Theta}, \mathcal{L})$ via *uniform* or *adaptive and sequential* partitioning of the hybrid state space $\mathcal{D}$. We further constrain the evolution of the $\mathcal{H}$ to a compact set $X \times \mathcal{U}$ over which the property of interest φ is expressed. X has the same dimensions as n and $\mathcal{U}$ has v dimensions. We partition $\mathcal{D}$ based on (i) a desired maximal abstraction error ε_{max} between the the original SHS and the abstract MDP, which is defined a priori and (ii) the continuity of the underlying continuous dynamics, quantified by the computation of the Lipschitz constants h_s.

In uniform partitioning, we discretise $\mathcal{D} \times \mathcal{U}$ along each $n \times v$ dimension to obtain finite and non-overlapping partitions

$$\mathcal{D} \times \mathcal{U} = \bigcup_{i=1}^{m} D_i \times \bigcup_{j=1}^{m_u} U_j \tag{5.1}$$

where m and m_u are the cardinality of the partition along the hybrid and input (actions) domain. On generating the grid, we can in turn construct the underlying $\mathcal{M}_{\text{MDP}} = (\mathcal{S}, \mathcal{A}, T_s, \bar{\Theta}, \mathcal{L})$ as follows:

- $\mathcal{S} = \{s_1, \ldots, s_m\}$ where $s_i \in D_i$ is a representative point for each cell in the partition in $i \in \{1, \ldots, m\}$;
- $\mathcal{A} = \{u_1, \ldots, u_{m_u}\}$ where $u_j \in U_j$ is a representative point for each cell in the partition in $j \in \{1, \ldots, m_u\}$;
- T_s is computed via marginalisation of the transition probability kernel and corresponds to quantifying (4.4) for each cell within the constructed grid;
- $\bar{\Theta} = \Theta$ is inherited from the original SHS;
- $\mathcal{L}$ assigns states s in the MDP to corresponding regions in the SHS.

Note, if $\mathcal{U} = \emptyset$, the abstraction procedure results in a DTMC.

Adaptive and sequential partitioning extends uniform partitioning by first constructing a coarse uniform grid for a maximal error which is significantly greater than the desired value (typically corresponding to $10\varepsilon_{max}$). For each cell, the local abstraction error ε_l is computed. We then split cells whose $\varepsilon_l > \varepsilon_{max}$ into smaller regions. This is repeated until $\varepsilon_l \leq \varepsilon_{max}$ across all the cells. The two approaches display an intuitive trade-off, where the first in general requires more memory but less time, whereas the second generates smaller abstractions. A pictorial summary of the abstraction methods is shown in Figure 5.1.

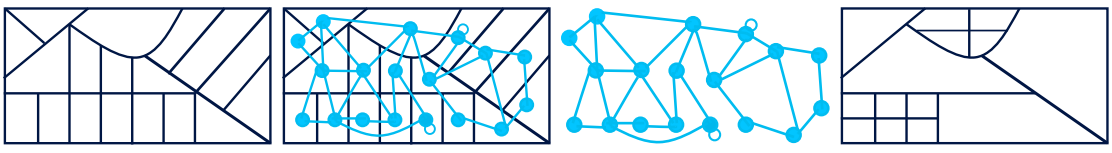


Figure 5.1: Overview of abstraction procedure for DTMCs or MDPs: using uniform partitioning (a) partition state space, (b) select representative points and compute transition probabilities and (c) combine to obtain the DTMC (for $\mathcal{U} = \emptyset$) or MDP, while (d) shows the resulting extension to adaptive and sequential partitioning.

Abstraction error

The K step error of this abstraction is given by

$$\varepsilon_{max} = h_s \Delta x_{\max} \mathcal{L}(A) K \tag{5.2}$$

where $\Delta x_{\max}$ is the max diameter of partition and $\mathcal{L}(A)$ volume of set [146].

5.1.1 Verification via Markov Decision Processes

Verification of Markov processes such as DTMCs and MDPs (when the action set is trivial) has been widely studied and adopted in literature [98]. The focus here is on the computation of probabilistic safety which can be formalised as a finite-horizon probabilistic invariance problem and is solved via Bellman recursion using the algorithms in [147]. This can be easily extended for reach-avoid properties which can be formulated as a probabilistic safety problem. For more complex requirements, one can leverage off-the-shelf probabilistic model checking tools, such as PRISM [100], STORM [48] or EPMC (formerly known as ISCASMC) [76]. The verification result is mapped back to the original SHS using

$$p_{\varphi}(s) = p(s) - \varepsilon_{max} \tag{5.3}$$

where p is the resulting probability of satisfaction for the MDP and p_{φ} is the refined result.

5.1.2 Strategy synthesis via Markov Decision Processes

Strategy synthesis for deterministic and memoryless Markov strategies $\pi = (\pi_0, \pi_1, \dots)$ where $\pi_k : d \to \mathcal{U}$ for $k = \{0, 1, \dots\}$ involves computing the strategy $\pi^\star$ that maximises the probability of satisfying a formula. This is achieved by solving a stochastic dynamic programming scheme, over the abstract MDP, such that the final value function provides the maximal probability p^{π^*} for the specification φ. The policy is refined back to the original SHS with a maximal probability $p_{\varphi}^{\pi^*} = p^{\pi^*} - \varepsilon_{max}$. We refer the reader to [146] for the exact algorithms for this computation.

For a subclass of SHS models where

$$\mathcal{H} = (\{q_0\}, n, \mathcal{U}, 1, \mathcal{N}(.|Ax[k] + Bu[k], G\Sigma G^T), \Theta, \mathcal{L}), \tag{5.4}$$

and an output map $y : \mathbb{R}^n \to \mathbb{R}^m,\ m \leq n$ exists on the continuous domain, we can alleviate the complexity of strategy synthesis for properties taking the form of $\varphi := \mathrm{P}_{\geq p}\left(\mathbf{G}^{\leq K}[|y| \leq \varpi]\right)$ where $\varpi \in \mathbb{N}$. The problem of state-space explosion introduced by having to discretise the control actions and replicating the state space for each discrete action is ameliorated via the use of hierarchical control that is formal and provides guarantees [73], [156]. The essence of the approach is to approximate the original (concrete) models, describing the evolution of the continuous dynamics, by simpler (reduced-order) models that are prone to be analysed or algorithmically verified. The original and simpler models are related using (ϵ, δ)-similarity relations introduced in [73]. A strategy synthesised on the simpler model can then be certifiably refined over the concrete model. The pair (ϵ, δ) represents the deviation in the output trajectories between complex and abstract models and the differences in probability distribution of the processes, respectively. Such metrics allow the designer to select which of the considered abstract models provides the best trade-off in precision: it is desirable to achieve little deviation in both the output trajectories (small ϵ) and the probability distributions (small δ).

In Section 5.4.2, we move beyond the purely theoretical treatment of these relations presented in [73] and show (i) how these relations can be employed to BAS models with a large number of continuous variables and (ii) perform a comparison with the classical approach, where strategy synthesis is performed directly over the whole high-dimensional continuous domain.

5.2 Abstractions into Interval Markov Decision Processes

The classical abstraction approach overviewed in Section 5.1, is a conservative method that over approximates the error between the original SHS and its abstraction. Consequently leading to (i) errors which increase linearly with time, (ii) state-space

explosions as the level of the abstraction error gets smaller and the number of continuous variables grows and (ii) limiting the analysis of SHS to finite time properties [146]. In this section, we present a novel abstraction method that allows for tighter abstraction errors, allowing us to analyse SHS with a larger number of continuous variables and to reason over both finite and infinite time properties. The work builds upon [102] which introduces an abstraction-refinement algorithm for performing formal verification and synthesis on discrete-time stochastic systems via IMDP. More specifically, we (i) extend the abstraction technique in [102] to reason over SHS with countable control actions, (ii) present a novel algorithm for computing the upper and lower bound probabilities of the abstract IMDP efficiently, and (iii) expand the analysis for reasoning over finite and infinite time properties expressed using CSLTL or BLTL.

Given $\mathcal{H} = (\mathcal{Q}, n, \mathcal{U}, T_q, T_x, \Theta, \mathcal{L})$ with a deterministic selection of locations $T_q : \mathcal{U} \to \mathcal{Q}$ (see Remark 3), continuous dynamics which are described with

$$T_x = \mathcal{N}(\cdot; F(q,x), G(q)\Sigma_w G(q)^T), \tag{5.5}$$

where, $F(q,x)$ is a linear function which takes the form of $A(q)x + Q(q)$ with $A \in \mathbb{R}^{n\times n},\ x \in \mathbb{R}^n,\ Q \in \mathbb{R}^n$ and recall, $G(q) \in \mathbb{R}^{n\times r}, \Sigma_w \in \mathbb{R}^{r\times r}$; and a continuous compact set X over which the formula φ corresponding to the property of interest is expressed, we abstract $\mathcal{H}$ into an IMDP $\mathcal{I} = (\mathcal{S}, \mathcal{A}, \check{T}_s, \hat{T}_s, \bar{\Theta}, \mathcal{L})$ such that verification and synthesis goals can be performed. This is a three step process as highlighted via Figure 5.2, consisting of:

- discretising the state space $X \times \mathcal{Q}$;
- quantifying the abstraction error due to discretisation, representing it as uncertainty and constructing the corresponding $\mathcal{I}$;
- performing verification or strategy synthesis via a two-player stochastic game, and refining the result back to the original SHS $\mathcal{H}$.

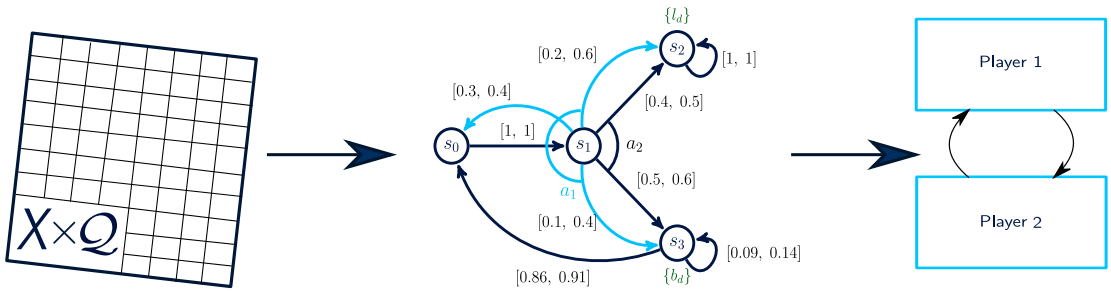


Figure 5.2: Overview of the general framework to analyse SHS by applying formal abstractions into IMDP.

5.2.1 Discretisation of hybrid state space

For each of the finite control actions, $u \in \mathcal{U}$, we use $T_q : \mathcal{U} \to \mathcal{Q}$ to obtain a finite set of discrete modes $\mathcal{Q}$. Typically, this is set to $\mathcal{U} = \mathcal{Q}$. Next, we discretise the hybrid state space $X \times \mathcal{Q}$. For each discrete mode $q \in \mathcal{Q}$, the corresponding set of interest X is partitioned into a set of cells (regions) that are non-overlapping, except for trivial sets of measure zero (their boundaries). We assume that each region is a bounded polytope and denote the resulting set of regions in mode q using $\mathcal{S}^q = \{s_1^q, ..., s_{|\mathcal{S}^q|}^q\}$.

5.2.2 Construction of Interval Markov Decision process

In this subsection, we show how each individual component making up $\mathcal{I} = (\mathcal{S}, \mathcal{A}, \check{T}_s, \hat{T}_s, \bar{\Theta}, \mathcal{L})$ is constructed.

States making up the Interval Markov Decision process

We associate a state of the IMDP s_i^q for each discretised cell in each mode $q \in \mathcal{Q}$ and overload the notation by using s_i^q for both a region in X, and a state of $\mathcal{I}$, i.e., $s_i^q \in \mathcal{S}^q$. Consequently, the set $(X \times \mathcal{Q}) \subset \mathcal{D}$ can be represented by $\bar{\mathcal{S}} = \bigcup_{q \in \mathcal{Q}} \mathcal{S}^q$. The set of IMDP states is $\mathcal{S} = \bar{\mathcal{S}} \cup \{s_u\}$ with s_u representing $\mathcal{D} \setminus (X \times \mathcal{Q})$, namely the complement of $X \times \mathcal{Q}$.

The set of IMDP states is $\mathcal{S} = \bar{\mathcal{S}} \cup \{s_u\}$.

Actions and transition probabilities

We define the set of actions $\mathcal{A}$ of $\mathcal{I}$ to be the set of modes $\mathcal{Q}$ of $\mathcal{H}$, and allow all actions to be available in each state of $\mathcal{I}$.

The set of IMDP actions is $\mathcal{A}(s) = \mathcal{Q}$ for all $s \in \mathcal{S}$.

We define the one-step transition probability from a continuous state $x \in X$ to region $s \in \bar{\mathcal{S}}$ under action (mode) $a \in \mathcal{A}$ to be defined by the transition kernel $T_x(s \mid x, a)$ in (5.5). The caveat is that states of $\mathcal{I}$ correspond to regions in $\mathcal{H}$ and there are uncountably many possible (continuous) initial states (here x) in each region, resulting in a range of feasible transition probabilities to the region s. Subsequently, the transition probability from one region to another can be characterised by a range given by the min and max of (5.5) over all the possible points x in the starting region. We can thus bound the transition of state $s_i \in \bar{\mathcal{S}}$ to state $s_j \in \bar{\mathcal{S}}$ from below by

$$\gamma^a_{s_i}(s_j) \geq \min_{x \in s_i} T_x(s_j \mid x, a), \tag{5.6}$$

and from above by

$$\gamma^a_{s_i}(s_j) \leq \max_{x \in s_i} T_x(s_j \mid x, a). \tag{5.7}$$

For $s_i, s_j \in \bar{\mathcal{S}}$, we can define the extrema $\check{T}_s$ and $\hat{T}_s$ of the transition probability of $\mathcal{I}$ according to these bounds. Similarly, we define the bounds of the feasible transition probabilities to states outside X as

$$\gamma^a_{s_i}(s_u) \geq 1 - \max_{x \in s_i} T_x(X \mid x, a), \tag{5.8}$$

$$\gamma^a_{s_i}(s_u) \leq 1 - \min_{x \in s_i} T_x(X \mid x, a), \tag{5.9}$$

and consequently set the bounds in $\mathcal{I}$ to be

$$\check{T}_s(s_i, a, s_u) = 1 - \max_{x \in s_i} T_x(X \mid x, a), \tag{5.10}$$

$$\hat{T}_s(s_i, a, s_u) = 1 - \min_{x \in s_i} T_x(X \mid x, a), \tag{5.11}$$

for all $a \in \mathcal{A}$ and $s_i \in \bar{\mathcal{S}}$. Finally, since we are not interested in the behaviour of $\mathcal{H}$ outside of $X \times \mathcal{Q}$, we render the state s_{u} of $\mathcal{I}$ absorbing, i.e., $\check{T}_s(s_u, a, s_u) = \hat{T}_s(s_u, a, s_u) = 1$, $\forall a \in \mathcal{A}$.

In summary, the upper bound and lower bound probabilities of $\mathcal{I}$ are computed using

$$\check{T}_s(s_j, a, s_i) = \begin{cases} \min_{x \in s_i} T_x(X \mid x, a) & s_j \neq s_u \wedge s_i \neq s_u \\ 1 - \min_{x \in s_i} T_x(X \mid x, a) & s_j = s_u \wedge s_i \neq s_u \\ 1 & s_j = s_u \wedge s_i = s_u \end{cases},$$

$$\hat{T}_s(s_j, a, s_i) = \begin{cases} \max_{x \in s_i} T_x(X \mid x, a) & s_j \neq s_u \wedge s_i \neq s_u \\ 1 - \max_{x \in s_i} T_x(X \mid x, a) & s_j = s_u \wedge s_i \neq s_u \\ 1 & s_j = s_u \wedge s_i = s_u \end{cases}$$

$\forall s_i, s_j \in \mathcal{S}$.

Atomic propositions and labelling function

We are interested in the properties of $\mathcal{H}$ in set $(X \times \mathcal{Q}) \subset \mathcal{D}$, where $X \subset \mathbb{R}^m$ is a continuous compact set. Specifically, we analyse the behaviour of $\mathcal{H}$ with respect to a set of closed regions of interest $R = \{r_1, \ldots, r_n\}$, where $r_i \subseteq X$. We associate to each region r_i the atomic proposition (label) θ_i, i.e., $\theta_i \in \mathcal{L}(d = (q, x)) \Leftrightarrow x \in r_i$. For discretisation of $X \times \mathcal{Q}$ that do not respect the regions in R, we represent (possibly conservatively) each r_i as well as its complement relative to X through the labelling of the states of $\mathcal{I}$. Let $r_{n+i} = X \setminus r_i$ be the complement region of r_i with respect to X. We associate to each r_{n+i} a new atomic proposition θ_{n+i} for $1 \leq i \leq n$. Intuitively, θ_{n+i} represents $\neg\theta_i$ with respect to X. We define the set of atomic propositions for $\mathcal{I}$ to be

$$\bar{\Theta} = \Theta \cup \{\theta_{n+1}, \ldots, \theta_{2n}\}. \tag{5.12}$$

Then, we design $L : \mathcal{Q} \to 2^{\bar{\Theta}}$ of $\mathcal{I}$ such that

$$\theta_i \in L(s) \quad \Leftrightarrow \quad s \subseteq r_i, \tag{5.13}$$

for all $s \in \bar{\mathcal{S}}$ and $0 \leq i \leq 2n$, and $\mathcal{L}(s_u) = \emptyset$.

With this modelling, we capture (possibly conservatively) all the property regions of $\mathcal{H}$ by the state labels of $\mathcal{I}$. Then, a formula φ over Θ of $\mathcal{H}$ can be easily translated to a formula $\bar{\varphi}$ on $\bar{\Theta}$ of $\mathcal{I}$ by rewriting φ into its negation normal form and replacing $\neg\theta_i$ with θ_{n+i}. We note that all CSLTL and BLTL formulas can be written in negation normal form without any loss in their expressive power [81].

Remark 4. *The extension of the atomic propositions in* (5.12) *is not necessary if the discretisation of $X \times \mathcal{Q}$ respects all the regions in R, i.e., $\exists \mathcal{S}_r \subseteq \mathcal{S}$ s.t. $\cup_{s \in \mathcal{S}_r} s = r$ for all $r \in R$.*

> $\bar{\Theta}$ and $\mathcal{L}(s)$ are generated by associating labels with the corresponding region R in X. When the discretisation does not respect R, add extra labels (conservatively) by converting φ into negation normal form and associate labels with the negation of propositions.

5.2.3 Efficient computation of the transition probabilities

In this section, we introduce an efficient and scalable method for space discretisation and computation for

$$\min_{x \in s_i} T_x(s_j \mid x, a), \qquad \max_{x \in s_i} T_x(s_j \mid x, a). \tag{5.14}$$

We first define a *hyper-rectangle* and *proper transformation function* as follows.

Definition 3 (Hyper-rectangle)**.** *A hyper-rectangle in $\mathbb{R}^n$ is an n-dimensional rectangle defined by the intervals*

$$[v_l^{(1)}, v_u^{(1)}] \times [v_l^{(2)}, v_u^{(2)}] \times \cdots \times [v_l^{(n)}, v_u^{(n)}], \tag{5.15}$$

where vectors $v_l, v_u \in \mathbb{R}^n$ capture the lower and upper values of the vertices of the rectangle in each dimension, and $v^{(i)}$ denotes the i-th component of vector v.

Definition 4 (Proper transformation)**.** *For a polytope $s \subset \mathbb{R}^n$, the transformation function $\mathcal{T} \in \mathbb{R}^{n \times n}$ is proper if $Post(s, \mathcal{T})$ is a hyper-rectangle.*

From (5.5), we note that process x in mode a is Gaussian with one-step covariance matrix

$$\Sigma_x(a) = G(a)\Sigma_w G(a)^T. \tag{5.16}$$

Then, we can characterise $T_x(s \mid x, a)$ analytically as follows.

Proposition 1. *For process x in mode $a \in \mathcal{A}$, let $\mathcal{T}_a = \Lambda_a^{-\frac{1}{2}} V_a^T$ be a transformation function (matrix), where $\Lambda_a = V_a^T \Sigma_x(a) V_a$ is a diagonal matrix whose entries are eigenvalues of $\Sigma_x(a)$ and V_a is the corresponding orthonormal (eigenvector) matrix. For a polytopic region $s \subset \mathbb{R}^n$, if $\mathcal{T}_a$ is proper, then it holds that*

$$T_x(s \mid x, a) = \frac{1}{2^n} \prod_{i=1}^{n} \Big(\mathrm{erf}(\frac{y^{(i)} - v_l^{(i)}}{\sqrt{2}}) - \mathrm{erf}(\frac{y^{(i)} - v_u^{(i)}}{\sqrt{2}}) \Big), \tag{5.17}$$

where $\mathrm{erf}(\cdot)$ is the error function, and $y^{(i)}$ is the i-th component of vector $y = \mathcal{T}_a F(a, x)$, and $v_l^{(i)}$, $v_u^{(i)}$ are as in (5.15).

A direct consequence of Proposition 1 is that the optimisations in (5.14) can be performed on (5.17) through a proper transformation, as stated by the following corollary.

Corollary 1. *For polytopic regions $s_i, s_j \subset \mathbb{R}^n$ and process x in mode a, assume $\mathcal{T}_a$ is a proper transformation function with respect to s_j, and define $s_i' = Post(s_i, F(a, s_i))$ and*

$$f(y) = \frac{1}{2^n} \prod_{i=1}^{n} \Big(\mathrm{erf}(\frac{y^{(i)} - v_l^{(i)}}{\sqrt{2}}) - \mathrm{erf}(\frac{y^{(i)} - v_u^{(i)}}{\sqrt{2}}) \Big), \tag{5.18}$$

where v_l and v_u are as in (5.15). Then, it holds that

$$\begin{aligned} \min_{x \in s_i} T_x(s_j \mid x, a) &= \min_{y \in Post(s_i', \mathcal{T}_a)} f(y), \\ \max_{x \in s_i} T_x(s_j \mid x, a) &= \max_{y \in Post(s_i', \mathcal{T}_a)} f(y). \end{aligned}$$

The above proposition and corollary show that, for a particular proper transformation function $\mathcal{T}_a$, an analytical form can be obtained for the discrete kernel of the IMDP. This is an important observation because it enables efficient computation for the min and max values of the kernel. Therefore, we use a space discretisation to satisfy the condition in Proposition 1 as described next.

Space discretisation

For each action $a \in \mathcal{A}$ (discrete mode $q \in \mathcal{Q}$), we define the linear transformation function (matrix) of

$$\mathcal{T}_a = \Lambda_a^{-\frac{1}{2}} V_a^T, \tag{5.19}$$

where $\Lambda_a = V_a^T \Sigma_x(a) V_a$ is a diagonal matrix whose entries are the eigenvalues of $\Sigma_x(a)$, and V_a is the corresponding orthonormal (eigenvector) matrix. The discretisation of the continuous set X in mode a is achieved by using a grid in the transformed space by $\mathcal{T}_a$. That is, we first transform X by $\mathcal{T}_a$, and then discretise it using a grid.

This method of discretisation guarantees that, for each $s^q \in \mathcal{S}^q$, $Post(s^q, \mathcal{T}_a)$ is a hyper-rectangle, i.e., $\mathcal{T}_a$ is proper. Hence, we can use the result of Proposition 1 and Corollary 1 for the computation of the values in (5.14).

Remark 5. *For an arbitrary geometry of X, it may not be possible to obtain a discretisation such that $\bigcup_{s^q \in \mathcal{S}^q} s^q = X$. Nevertheless, by using a discretisation that under-approximates X, i.e., $\bigcup_{s^q \in \mathcal{S}^q} s^q \subseteq X$ or over-approximates $\neg X$, in each action a (discrete mode q), we can compute a lower bound on the probability of satisfaction of a given property φ. For a better approximation, the grid can be non-uniform, allowing in particular for smaller cells near the boundary of X (as in the adaptive and sequential gridding technique described in Section 5.1).*

Transition probability bounds

We distinguish between transitions from $s \in \bar{\mathcal{S}}$ to the states in $\bar{\mathcal{S}}$ and to s_{u}.

Transitions to $s \in \bar{\mathcal{S}}$

We present two approaches to solving the values for (5.14). The first approach is based on *Karush-Kuhn-Tucker* (KKT) conditions [28], which shed light on the optimisation problem and lays down the conditions on where to look for the optimal points, giving geometric intuition. This method boils down to solving systems of non-linear equations, which turns out to be efficient and exact for low-dimensional systems. In the second approach, we show that the problem reduces to a convex

optimisation problem, allowing the adoption of existing optimisation tools and hence making the approach suitable for high-dimensional systems.

KKT Optimisation Approach: In the next theorem, we use the result of Corollary 1 and the KKT conditions [28] to compute the exact values for (5.14).

Theorem 5.2.1. *For polytopic regions $s_i, s_j \subset \mathbb{R}^n$ and proper transformation matrix $\mathcal{T}_a$ with respect to s_j, let*

$$Post(s'_i, \mathcal{T}_a) = \{y \in \mathbb{R}n \mid Hy \leq b\},$$

where $s'_i = Post(s_i, F(a, s_i))$, $H \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^n$, and $k \geq n+1$, and introduce the following conditions:

- **Condition 1***: y is at the centre of $Post(s_j, \mathcal{T}_a)$, i.e.,*

$$y = (\frac{v_u^{(1)} + v_l^{(1)}}{2}, \ldots, \frac{v_u^{(n)} + v_l^{(n)}}{2}).$$

- **Condition 2***: y is a vertex of $Post(s'_i, \mathcal{T}_a)$.*

- **Condition 3***: y is on the boundary of $Post(s'_i, \mathcal{T}_a)$, where $r \geq 1$ of the k half-spaces that define $Post(s'_i, \mathcal{T}_a)$ intersect, and*

$$\nabla f(y) = \bar{H}^T \eta,$$

  for vector $\eta = (\eta_1, \ldots, \eta_r)$ of non-negative constants, and sub-matrix $\bar{H} \in \mathbb{R}^{r \times n}$ that contains only the rows of H that correspond to the r-intersecting half-spaces at y.

- **Condition 4***: y is as in Condition 3, and*

$$\nabla f(y) = -\bar{H}^T \eta,$$

  for vector $\eta = (\eta_1, \ldots, \eta_r)$ of non-negative constants, and $\bar{H}$ is defined as in Condition 3.

Then, it follows that the point $y \in Post(s'_i, \mathcal{T}_a)$ that satisfies Condition 1 necessarily maximises $f(y)$. If Condition 1 cannot be satisfied, then the maximum is necessarily given by one of the points that satisfy Condition 2 or 3. Furthermore, the point $y \in Post(s'_i, \mathcal{T}_a)$ that minimises $f(y)$ necessarily satisfies Condition 2 or 4.

The proof of Theorem 5.2.1 can be found in Appendix B.2. Theorem 5.2.1 identifies the arguments (points $y \in Post(s'_i, \mathcal{T}_a)$) that give rise to the optimal values of T_x in (5.14). Then, the actual optimal values of T_x can be computed by (5.18) as guaranteed by Corollary 1. Thus, from Theorem 5.2.1, an algorithm can be constructed to generate a set of finite candidate points based on Conditions 1-4 and to obtain the exact values of (5.14) by plugging those points into (5.18).

In short, Condition 1 maximises the unconstrained problem and gives rise to the global maximum. Hence, if the centre of s_j is contained in $Post(s'_i, \mathcal{T}_a)$, no further check is required for maximum. If not, the maximum is given by a point on the boundary of $Post(s'_i, \mathcal{T}_a)$. It is either a vertex (Condition 2) or a boundary point that satisfies Condition 3. The minimum is always given by a boundary point, which can be either a vertex or a boundary point that satisfies Condition 4. Note that Conditions 3 and 4 are similar and both state that the optimal value of T_x is given by a point where the gradient of T_x becomes linearly dependent on the vectors that are defined by the intersecting half-spaces of $Post(s'_i, \mathcal{T}_a)$ at that point. Each of these two conditions defines a system of n equations and $r < n$ variables, which may have a solution only if some of the equations are linear combinations of the others.

The above algorithm computes the exact values for the transition probability bounds. It is computationally efficient for small dimensional systems, e.g., $n < 4$. For large n, however, the efficiency drops because of the number of boundary constraints that need to be checked and solved for in Conditions 3 and 4 increases, in the worst case, exponentially with n. Below, we propose an equivalent but more efficient method to compute min and max of T_x for large dimensional systems, e.g., $n \geq 4$.

Convex Optimisation Approach: In order to show how upper and lower bounds of $f(y)$ can be efficiently computed using convex optimisation tools, we need to introduce the definition of *concave* and *log-concave* functions.

Definition 5 (Concave Function)**.** *A function $g : \mathbb{R}^n \to \mathbb{R}$ is said to be concave if and only if for $y_1, y_2 \in \mathbb{R}^n$, $\lambda \in [0, 1]$*

$$g(\lambda y_1 + (1 - \lambda)y_2) \geq \lambda g(y_1) + (1 - \lambda)g(y_2).$$

Definition 6 (Log-concave Function)**.** *A function $g : \mathbb{R}^n \to \mathbb{R}$ is said to be log-concave if and only if $log(g)$ is a concave function. That is, for $y_1, y_2 \in \mathbb{R}^n$, $\lambda \in [0, 1]$*

$$g(\lambda y_1 + (1 - \lambda)y_2) \geq g(y_1)^{\lambda} g(y_2)^{(1-\lambda)}.$$

In the following proposition, we show that $f(y)$, as defined in Corollary 1, is log-concave. This enables efficient computation of the upper and lower bounds of $f(y)$ through standard convex optimisation techniques such as gradient descent or semi- definite programming [32]. Consequently, readily available software tools, e.g., CVX [64], NLOPT [88], which have been highly optimised in terms of efficiency and scalability, can be used.

Proposition 2. *$f(y)$, as defined in Corollary 1, is a log-concave function.*

Transitions to sink state s_{u}

Here, focus is on the transition probabilities to state s_u in (5.10) and (5.11). Correspondingly, we need to compute

$$\max_{x \in s_i} T_x(X \mid x, a), \quad \min_{x \in s_i} T_x(X \mid x, a). \tag{5.20}$$

The bounds for these quantities can be efficiently computed using the results obtained above and the following proposition shows this efficient method of computation.

Proposition 3. *Let $\check{\mathcal{S}}^q$ and $\hat{\mathcal{S}}^q$ be two sets of polytopic regions in action a (mode q) such that*

$$\bigcup_{s \in \check{\mathcal{S}}^q} s \;\subseteq\; X \;\subseteq\; \bigcup_{s \in \hat{\mathcal{S}}^q} s,$$

and $\mathcal{T}_a$ be a proper transformation function for every $s \in \check{\mathcal{S}}^q \cup \hat{\mathcal{S}}^q$, and call

$$f(y, s) = \frac{1}{2^n} \prod_{i=1}^{n} \Big(\mathrm{erf}(\frac{y^{(i)} - v_{l,s}^{(i)}}{\sqrt{2}}) - \mathrm{erf}(\frac{y^{(i)} - v_{u,s}^{(i)}}{\sqrt{2}}) \Big), \tag{5.21}$$

where $v_{l,s}$ and $v_{u,s}$ are as in (5.15) *for s. Then, it holds that*

$$\max_{x \in s_i} T_x(X \mid x, a) \leq \max_{y \in Post(s_i', \mathcal{T}_a)} \sum_{s \in \hat{\mathcal{S}}^a} f(y, s), \tag{5.22}$$

$$\min_{x \in s_i} T_x(X \mid x, a) \geq \min_{y \in Post(s_i', \mathcal{T}_a)} \sum_{s \in \hat{\mathcal{S}}^a} f(y, s), \tag{5.23}$$

where $s_i' = Post(s_i, F(a, s_i))$.

The proof of this proposition is in Appendix B.4 and is direct consequence of Proposition 1. Intuitively, Proposition 3 states that, with a particular choice of discretisation, i.e., a grid in the transformed space, the transition probability to X is equal to the sum of the transition probabilities to the discrete regions, where each discrete transition kernel is given by the close-form function $f(y, s)$ in (5.21). If X cannot be precisely discretised with a grid (in the transformed space), then the upper and lower bounds of the transition probabilities are given by the over- and under-approximating grids ($\hat{\mathcal{S}}^q$ and $\check{\mathcal{S}}^q$), respectively.

Remark 6. *For the computation of the values in* (5.22) *and* (5.23)*, Proposition 2 can be applied, making both methods of* KKT *and convex optimisation applicable.*

Abstraction error

We define the local abstraction error to be equal to the difference in the upper ($\hat{p}_\varphi$) and lower ($\check{p}_\varphi$) probability of satisfying the formula φ

$$\varepsilon_s(s) = \hat{p}_\varphi(s) - \check{p}_\varphi(s) \tag{5.24}$$

for each state $s \in \mathcal{S}$, and the global error to be

$$\varepsilon_{max} = \max_{s \in \mathcal{S}} \varepsilon_s(s). \tag{5.25}$$

Adaptive and sequential gridding

Similar to Section 5.1, we extend the abstraction procedure to generate the underlying grid using adaptive and sequential refinements. This is performed as follows: (i) define a required minimal maximum abstraction error ε_{max}; (ii) generate a coarse abstraction using the steps described in Section 5.2.2 and compute

the local error ε_s that is associated to each abstract state s; (iii) split all cells where $\varepsilon_s > \varepsilon_{max}$ along the main axis of each dimension, and update the probability bounds (and errors); and (iv) repeat this process until $\forall s,\ \varepsilon_s \leq \varepsilon_{max}$.

5.2.4 Verification via Interval Markov Decision Processes

Given the compact set X and a CSLTL or BLTL formula φ, we would like to verify whether the $\mathcal{H}$ satisfies φ. The abstract IMDP $\mathcal{I}$, as constructed in Section 5.2.2, captures (possibly conservatively) the behaviour of the SHS $\mathcal{H}$ with respect to the regions of interest R within X, and the probabilities of exiting X are encompassed via the state s_{u}. The analysis on $\mathcal{I}$ narrows the focus to dynamics within set X, by making state s_{u} absorbing, since the paths of $\mathcal{I}$ are not allowed to exit and re-enter X. Consequently, the verification task reduces to computing the extremal (maximal or minimal) reachability probabilities of satisfying φ, against all the uncertainties (errors) introduced by the discretisation of $X \times \mathcal{Q}$.

In order to capture the *best-* and *worst*-case behaviour that can arise, for all possible strategies, we need to reason about paths and strategies. To this end, we say that a finite path $\omega_{\mathcal{H}}$ of $\mathcal{H}$, initialised at state $d_0 \in \mathcal{D}$, satisfies a formula φ if the path remains in the compact set X and its corresponding finite trace $\xi \models \varphi$. Under a switching strategy $\pi_{\mathcal{H}}$, the probability that the SHS satisfies φ is given by

$$P(\varphi \mid d_0, X, \pi_{\mathcal{H}}) = P\Big(\omega_{\mathcal{H}} \in \mathrm{Path}_{\mathcal{H}}^{\mathrm{fin},\pi_{\mathcal{H}}}(s) \mid \omega_{\mathcal{H}}(0) = d_0, \\ \omega_{\mathcal{H}}(k) \in (X \times \mathcal{Q})\ \forall k \in [0, |\omega_{\mathcal{H}}|\,],\ \xi \models \varphi\Big), \quad (5.26)$$

where $\mathrm{Path}_{\mathcal{H}}^{\mathrm{fin},\pi_{\mathcal{H}}}(s)$ denotes the set of all finite paths under strategy $\pi_{\mathcal{H}}$, and ξ is the (observation) trace of $\omega_{\mathcal{H}}$.

Furthermore, the uncertainties introduced by the discretisation of $X \times \mathcal{Q}$, can be viewed as the nondeterministic choice of a feasible transition probability from one IMDP state to another under a given action. Therefore, we interpret the evolution of the IMDP as a *two-player stochastic game*, where Player 1 chooses an action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$, and Player 2 chooses a feasible transition probability distribution $\gamma_s^a \in \Gamma_s^a$. For computing the minimal probability, we need to choose

both the actions and the feasible transition probability distribution that minimise satisfaction over the under-approximation of the discretisation region. In contrast, to compute the maximal probability, we need to select the actions and feasible transition probability distribution which maximise the satisfaction, respectively, using an over-approximation of the discretisation region.

To this end, we first translate φ over Θ into its equivalent formula $\bar{\varphi}$ over $\bar{\Theta}$. Then, we construct a *deterministic finite automaton* (DFA) $\mathcal{A}_{\bar{\varphi}}$ that precisely accepts all the good prefixes that satisfy $\bar{\varphi}$ [95].

Definition 7 (Deterministic Finite Automata (DFA))**.** *A* DFA *constructed from a* CSLTL *or* BLTL *formula $\bar{\varphi}$ is a tuple*

$$\mathcal{A}_{\bar{\varphi}} = (\mathcal{Z}, 2^{\bar{\Theta}}, \tau, z_0, \mathcal{Z}_{\text{ac}}), \quad \textit{where} \tag{5.27}$$

$\mathcal{Z} = \{z_0 z_1 \dots z_l\}$, $l \in \mathbb{N}$ is a finite set of states, $2^{\bar{\Theta}}$ is the set of input alphabets, $\tau : \mathcal{Z} \times 2^{\bar{\Theta}} \to \mathcal{Z}$ is the transition function, $z_0 \in \mathcal{Z}$ is the initial state and $\mathcal{Z}_{\text{ac}} \subseteq \mathcal{Z}$ is the set of accepting states.

A *finite run* of $\mathcal{A}_{\bar{\varphi}}$ on a trace $\xi = \xi_1 \cdots \xi_l$, where $\xi_i \in 2^{\bar{\Theta}}$, is a sequence of states $\zeta = \{z_0 z_1 \dots z_l\}$ with $z_i = \tau(z_{i-1}, \xi_i)$ for $i = \{1, \dots, l\}$. Run ζ is called *accepting* if $\zeta_l \in \mathcal{Z}_{\text{ac}}$. Trace $\xi \models \bar{\varphi}$ iff its corresponding run ζ in $\mathcal{A}_{\bar{\varphi}}$ is accepting.

Next, we construct the product IMDP $\mathcal{I}_{\bar{\varphi}} = \mathcal{I} \times \mathcal{A}_{\bar{\varphi}}$, which is a tuple $\mathcal{I}_{\bar{\varphi}} = (\mathcal{S}_{\bar{\varphi}}, \mathcal{A}_{\bar{\varphi}}, \check{T}_{s,\bar{\varphi}}, \hat{T}_{s,\bar{\varphi}}, \mathcal{S}_{\bar{\varphi}\text{ac}})$, where

$$\mathcal{S}_{\bar{\varphi}} = \mathcal{S} \times \mathcal{Z}, \quad \mathcal{A}_{\bar{\varphi}} = \mathcal{A}, \quad \mathcal{S}_{\bar{\varphi}\text{ac}} = \mathcal{S} \times \mathcal{Z}_{\text{ac}},$$

$$\check{T}_{s,\bar{\varphi}}((s,z),a,(s',z')) = \begin{cases} \check{T}_s(s,a,s') & \text{if } z' = \tau(z, L(s')) \\ 0 & \text{otherwise,} \end{cases}$$
$$\hat{T}_{s,\bar{\varphi}}((s,z),a,(s',z')) = \begin{cases} \hat{T}_s(s,a,s') & \text{if } z' = \tau(z, L(s')) \\ 0 & \text{otherwise,} \end{cases}$$

for all $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, $z \in \mathcal{Z}$ and $\mathcal{S}_{\bar{\varphi}\text{ac}}$ is the set of accepting states with respect to the product IMDP. Intuitively, $\mathcal{I}_{\bar{\varphi}}$ contains both $\mathcal{I}$ and $\mathcal{A}_{\bar{\varphi}}$ and hence can identify all the paths of $\mathcal{I}$ that satisfy $\bar{\varphi}$, i.e., the satisfying paths terminate in $\mathcal{S}_{\bar{\varphi}\text{ac}}$ since their

corresponding $\mathcal{A}_{\bar{\varphi}}$ runs are accepting. Therefore, the verification problem reduces to finding the minimal and maximal probability of reaching $\mathcal{S}_{\bar{\varphi}\text{ac}}$.

Let $\check{p}(s)$ and $\hat{p}(s)$ denote lower and upper bounds for the probability of reaching a state in $\mathcal{S}_{\bar{\varphi}\text{ac}}$ starting from $s \in \mathcal{S}_{\bar{\varphi}}$ under π. Derived from the Bellman equation, we can compute the worst lower bound by recursive evaluations of

$$\check{p}(s) = \begin{cases} 1 & \text{if } s \in \mathcal{S}_{\bar{\varphi}\text{ac}} \\ \min\limits_{a \in \mathcal{A}(s)} \min\limits_{\gamma_s^a \in \Gamma_s^a} \sum\limits_{s' \in \mathcal{S}_{\bar{\varphi}}} \gamma_s^a(s') \check{p}(s') & \text{otherwise,} \end{cases} \tag{5.28}$$

for all $s \in \mathcal{S}_{\bar{\varphi}}$ using the original labelling function $\mathcal{L}$.

To compute the best upper bound, we need to take special care for the assumption in Theorem 5.2.2 as to whether the discretisation $\mathcal{S}$ respects the regions in R is violated. In this case, the upper bound $\hat{p}_\varphi$ is valid with respect to the under-approximate representation of R by $\mathcal{L}$ but possibly being under-approximated with respect to the actual R. Consequently, to account for this we need to design a new labelling function that over-approximates the labels of each region, as follows. Let $L' : \mathcal{S} \to \bar{\Theta}$ be this labelling function with

$$\theta_i \in \mathcal{L}'(s) \quad \Leftrightarrow \quad \exists (a, x) \in s \text{ s.t. } x \in r_i, \tag{5.29}$$

where $\theta_i \in \bar{\Theta}$ is the associated proposition to $r_i \in R$. Then, we can compute the over-approximated upper bound $\hat{p}$ via

$$\hat{p}(s) = \begin{cases} 1 & \text{if } s \in \mathcal{S}_{\bar{\varphi}\text{ac}} \\ \max\limits_{a \in \mathcal{A}(s)} \max\limits_{\gamma_s^a \in \Gamma_s^a} \sum\limits_{s' \in \mathcal{S}_{\bar{\varphi}}} \gamma_s^a(s') \hat{p}(s') & \text{otherwise,} \end{cases} \tag{5.30}$$

on the product IMDP $\mathcal{I}'_{\bar{\varphi}}$ constructed using $\mathcal{L}'$. In both instances, the Bellman equations describing the upper and lower probability bounds are guaranteed to converge in finite time [102], [165]. Moreover, we obtain that

$$\check{p}_\varphi(s) = \check{p}((s, z_0)), \quad \hat{p}_\varphi(s) = \hat{p}((s, z_0)). \tag{5.31}$$

For verification we are typically interested in the lower bound probability (worst case scenario), while the upper bound is used to compute the abstraction error. Consequently, the resulting probability of satisfaction on the original SHS is then,

$$p_\varphi(s) = \check{p}_\varphi(s), \tag{5.32}$$

for each $s \in \mathcal{S}$.

5.2.5 Strategy synthesis via Interval Markov Decision Processes

Given the IMDP abstraction $\mathcal{I}$, our goal is to synthesise a switching strategy $\pi^*_{\mathcal{H}}$ that maximises the probability of satisfying a property expressed as a CSLTL or BLTL formula φ,

$$\pi^*_{\mathcal{H}} = \arg \max_{\pi_{\mathcal{H}} \in \Pi_{\mathcal{H}}} P(\varphi \mid d_0, X, \pi_{\mathcal{H}})$$

for all initial states $d_0 \in X \times \mathcal{Q}$. The strategy is to be robust against all the introduced uncertainties (errors) by the discretisation of the space domain. Once again, we interpret the evolution of the IMDP as a two-player stochastic game, where Player 1 chooses an action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$, and Player 2 chooses a feasible transition probability distribution $\gamma^a_s \in \Gamma^a_s$. This game is adversarial, where the objectives of Players 1 and 2 are to maximise and minimise the probability of remaining in the safe set, respectively. Hence, the goal becomes to synthesise a strategy for Player 1 that is robust against all adversarial choices of Player 2 and maximises the probability of achieving φ.

In order to compute this strategy, we first translate φ over Θ into its equivalent formula $\bar{\varphi}$ over $\bar{\Theta}$. Then, much like for verification, we construct a DFA $\mathcal{A}_{\bar{\varphi}}$ that precisely accepts all the good prefixes that satisfy $\bar{\varphi}$ (see Definition 7). Next, we construct the product IMDP $\mathcal{I}_{\bar{\varphi}} = \mathcal{I} \times \mathcal{A}_{\bar{\varphi}} = (\mathcal{S}_{\bar{\varphi}}, \mathcal{A}_{\bar{\varphi}}, \check{T}_{s,\bar{\varphi}}, \hat{T}_{s,\bar{\varphi}}, \mathcal{S}_{\bar{\varphi}\text{ac}})$, which contains both $\mathcal{I}$ and $\mathcal{A}_{\bar{\varphi}}$ and hence can identify all the paths of $\mathcal{I}$ that satisfy $\bar{\varphi}$, i.e., the satisfying paths terminate in $\mathcal{S}_{\bar{\varphi}\text{ac}}$ since their corresponding $\mathcal{A}_{\bar{\varphi}}$ runs are accepting. Consequently, the synthesis problem reduces to computing a robust strategy on $\mathcal{I}_{\bar{\varphi}}$ that maximises the probability of reaching $\mathcal{S}_{\bar{\varphi}\text{ac}}$. This problem is equivalent to solving the *maximal reachability probability problem* [102], [165] as explained below.

Given a strategy π on an IMDP, the probability of reaching a terminal state from each state is necessarily a range for all the available adversarial choices of Player 2. Let $\check{p}^\pi(s)$ and $\hat{p}^\pi(s)$ denote lower and upper bounds for the probability of reaching a state in $\mathcal{S}_{\bar{\varphi}\text{ac}}$ starting from $s \in \mathcal{S}_{\bar{\varphi}}$ under π. Derived from the Bellman equation,

we can compute the optimal lower bound by recursive evaluations of

$$\check{p}^{\pi^*}(s) = \begin{cases} 1 & \text{if } s \in \mathcal{S}_{\bar{\varphi}\text{ac}} \\ \max\limits_{a \in \mathcal{A}(s)} \min\limits_{\gamma_s^a \in \Gamma_s^a} \sum\limits_{s' \in \mathcal{S}_{\bar{\varphi}}} \gamma_s^a(s') \check{p}^{\pi^*}(s') & \text{otherwise,} \end{cases} \tag{5.33}$$

for all $s \in \mathcal{S}_{\bar{\varphi}}$. Each iteration of this Bellman equation involves a minimisation over the adversarial choices, which can be computed through an ordering of the states of $\mathcal{I}_{\bar{\varphi}}$ [61], [102], and a maximisation over the actions. This Bellman equation is guaranteed to converge in finite time [102], [165] and results in the lower-bound probability $\check{p}^{\pi^*}(s)$ for each $s \in \mathcal{S}_{\bar{\varphi}}$ and in a stationary (memoryless in the product) strategy π^*. The upper bounds are similarly given by recursive evaluations of

$$\hat{p}^{\pi^*}(s) = \begin{cases} 1 & \text{if } s \in \mathcal{S}_{\bar{\varphi}\text{ac}} \\ \max\limits_{\gamma_s^{\pi^*} \in \Gamma_s^{\pi^*}} \sum\limits_{s' \in \mathcal{S}_{\bar{\varphi}}} \gamma_s^{\pi^*}(s') \hat{p}^{\pi^*}(s') & \text{otherwise,} \end{cases} \tag{5.34}$$

which is also guaranteed to converge in finite time.

The optimal strategy π^* on $\mathcal{I}_{\bar{\varphi}}$ can be mapped onto the states and actions of the abstraction IMDP $\mathcal{I}$, resulting in a (history-dependent) strategy. By construction, then the optimal lower and upper probability bounds of satisfying φ from the states of $\mathcal{I}$ are

$$\check{p}_{\varphi}^{\pi^*}(s) = \check{p}^{\pi^*}((s, z_0)), \quad \hat{p}_{\varphi}^{\pi^*}(s) = \hat{p}^{\pi^*}((s, z_0)), \tag{5.35}$$

for all $s \in \mathcal{S}$ of $\mathcal{I}$.

The complexity of the above strategy synthesis algorithm is polynomial in the size of the IMDP $\mathcal{I}_{\bar{\varphi}}$ [102], [165] and exponential in the size of the formula φ (in the worst case) [95]. Note that the size of φ used to express the properties of SHS is typically small.

Correctness

We show that the strategy π^* computed over $\mathcal{I}$ can be refined over (mapped onto) $\mathcal{H}$ and the lower probability bound $\check{p}_{\varphi}^{\pi^*}$ on $\mathcal{I}$ always holds for the hybrid system $\mathcal{H}$. The upper bound $\hat{p}_{\varphi}^{\pi^*}$ also holds for $\mathcal{H}$ if the discretisation respects the regions in R. In the case that the discretisation is not R-respecting, we use the modified upper bound that holds for $\mathcal{H}$.

Let $\mathcal{L} : \mathcal{D} \to \mathcal{S}$ be a function that maps the hybrid states $d \in \mathcal{D}$ to their corresponding discrete regions (states of $\mathcal{I}$), i.e, $\mathcal{L}(d) = s \in \mathcal{S}$ if $d \in s$. With a slight abuse of notations, we also use $\mathcal{L}$ to denote the mapping from the finite paths of $\mathcal{H}$ to their corresponding paths of $\mathcal{I}$, i.e.,

$$\omega^k_{\mathcal{H}} = d_0 d_1 \dots d_k \quad \Rightarrow \quad \mathcal{L}(\omega^k_{\mathcal{H}}) = \mathcal{L}(d_0)\mathcal{L}(d_1)\dots\mathcal{L}(d_k).$$

Then, the IMDP strategy π^* correctly maps to a switching strategy $\pi^*_{\mathcal{H}}$ for $\mathcal{H}$ via

$$\pi^*_{\mathcal{H}}(\omega^k_{\mathcal{H}}) = \pi^*(\mathcal{L}(\omega^k_{\mathcal{H}})). \tag{5.36}$$

The following theorem shows that for a given φ, the probability bounds $\check{p}^{\pi^*}_{\varphi}$ and $\hat{p}^{\pi^*}_{\varphi}$ are guaranteed to hold for the process $\mathbf{d}$ under $\pi^*_{\mathcal{H}}$ as constructed above.

Theorem 5.2.2. *Given a* SHS $\mathcal{H}$*, a continuous set* X*, and a* CSLTL *or* BLTL *formula* φ*, let* $\mathcal{I}$ *be the* IMDP *abstraction of* $\mathcal{H}$ *as described in Section 5.2 through a discretisation that respects the regions of interest in* R*. Further, let* π^* *be the strategy on* $\mathcal{I}$ *computed by* (5.33) *and* (5.34) *with probability bounds* $\check{p}^{\pi^*}_{\varphi}$ *and* $\hat{p}^{\pi^*}_{\varphi}$ *in* (5.35)*. Refine* π^* *into a switching strategy* $\pi^*_{\mathcal{H}}$ *as in* (5.36)*. Then, for any initial hybrid state* $d_0 \in \mathcal{D}$*, where* $d_0 \in s_0 \in \mathcal{S}$*, it holds that*

$$P(\varphi \mid d_0, X, \pi^*_{\mathcal{H}}) \in \left[\check{p}^{\pi^*}_{\varphi}(s_0),\, \hat{p}^{\pi^*}_{\varphi}(s_0)\right]. \tag{5.37}$$

Note that an assumption in Theorem 5.2.2 is that the discretisation $\mathcal{S}$ respects the regions in R. If this assumption is violated, then the lower bound $\check{p}^{\pi^*}_{\varphi}$ still holds, unlike the upper bound $\hat{p}^{\pi^*}_{\varphi}$. We design the labelling function $\mathcal{L}$ of $\mathcal{I}$ to under-approximate the regions of interest $r \in R$, making the upper bound $\hat{p}^{\pi^*}_{\varphi}$ valid with respect to the under-approximate representation of R by $\mathcal{L}$ but possibly under-approximated with respect to the actual R. To compute an upper bound that accounts for this, we use the $\mathcal{L}' : \mathcal{S} \to \bar{\Theta}$ defined by (5.29). Then, we can compute the over-approximated upper bound $\hat{p}'^{\pi^*}_{\varphi}$ via (5.34) on the product IMDP $\mathcal{I}'_{\bar{\varphi}}$ constructed using $\mathcal{L}'$.

Lemma 1. *If abstraction $\mathcal{I}$ is constructed through a discretisation that does not respect the regions in R, then*

$$P(\varphi \mid d_0, X, \pi^*_{\mathcal{H}}) \in \left[\check{p}^{\pi^*}_{\varphi}(s_0), \hat{p}'^{\pi^*}_{\varphi}(s_0)\right], \tag{5.38}$$

where $\hat{p}'^{\pi^*}_{\varphi}$ is computed via (5.34) *using the labels in* (5.29).

Theorem 5.2.2 and Lemma 1 guarantee that the satisfaction probability of φ for the process $\mathbf{d}$, solution of the SHS $\mathcal{H}$, is contained in the probability interval computed on the abstraction $\mathcal{I}$. The size of this interval depends on the difference of the one-step transition probability bounds of $\check{T}_s$ and $\hat{T}_s$ as well as the embedded approximations in the labelling functions $\mathcal{L}$ and $\mathcal{L}'$ in $\mathcal{I}$, which can be viewed as the error induced by space discretisation of $\mathcal{H}$ cast into the abstraction $\mathcal{I}$. This error can be tuned by the size of the discretisation: in particular, in the limit of an infinitely fine grid, the error of the abstraction goes to zero, and the IMDP abstraction is refined into an MDP, namely for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, $\check{T}_s(s, a, s') \to T_s(s, a, s') \leftarrow \hat{T}_s(s, a, s')$.

Remark 7. *In practice, the interest in both verification and synthesis problems is typically on deriving lower bounds for the probability, whereas the upper bound computation is useful for error analysis.*

5.3 Comparison of abstraction frameworks

In order to highlight the efficacy of the novel IMDP abstraction framework, we present a comparison against the classical approach of performing abstractions into DTMCs or MDPs. We first compare the abstraction methods based on the computational time and the quality of the abstraction generated, in terms of the global abstraction error, for fixed grid size. Second, we analyse the scalability of the IMDP framework as the continuous dimension of the SHS is increased.

The implementation of the abstraction algorithm is in MATLAB and C++: more precisely, the approach based on KKT method is in MATLAB (proof of concept), and the convex optimisation method with *gradient decent* (GD) is in C++. The

IMDP synthesis and verification algorithm is also implemented in C++. The C++ implementation uses the linear algebra library Armadillo [140] for the main algorithm and NLOPT [88] to compute the bounds for the transition probabilities. For the MDP abstractions we make use of the standard tool FAUST^2 when comparing against the KKT method, since both are implemented in MATLAB. For comparison between the IMDP method with GD, we compare against a re-implementation of the abstraction based methods in FAUST^2 in the C++ language.

5.3.1 Efficiency and quality of abstractions

We consider a SHS where the continuous dynamics evolve according to

$$\begin{matrix} x_1[k+1] \\ x_2[k+1] \end{matrix} = \begin{bmatrix} 0.85 & 0 \\ 0 & 0.90 \end{bmatrix} \begin{matrix} x_1[k] \\ x_2[k] \end{matrix} + \begin{bmatrix} 0.15 & 0 \\ 0 & 0.05 \end{bmatrix} \begin{matrix} w_1[k] \\ w_2[k] \end{matrix}, \tag{5.39}$$

with $w_i \sim \mathcal{N}(0, \mathbf{I})$, $i = \{1, 2\}$ being i.i.d Gaussian perturbations, a single discrete mode ($\mathcal{Q} = \{q_0\}$), with $X = [-1, 1] \times [-1, 1]$ and safety property

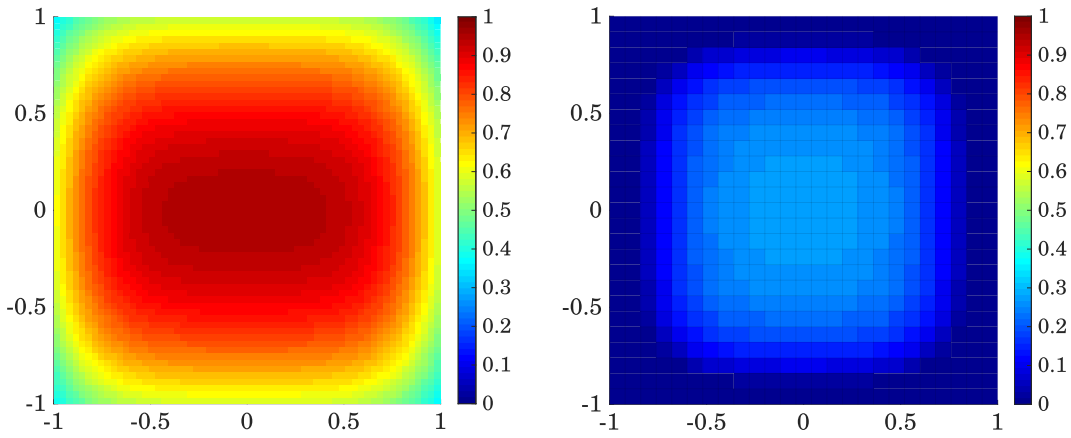
$$\varphi := \mathrm{P}_{=?} \left(\mathbf{G}^{\leq K} \, X \right).$$

We compare the verification results of the above model using our method against those of the state-of-the-art tool FAUST^2 [147]. Namely, we compare the probability of satisfaction of φ, computation times, and errors for a range of values for time horizon k and grid sizes. To obtain the IMDP abstraction of our method, we used a uniform grid discretisation following Section 5.2.2. Tool FAUST^2 abstracts the model into an MDP, treats the error as a separate parameter and performs abstraction into MDPs following the technique delineated in Section 5.1. Similarly, the error of the IMDP method is computed using (5.25) and corresponds to the maximal difference between the upper and lower bounds of the probability of satisfaction (i.e. maximum error over all the states). The results are shown in Table 5.1 for $K = 2$ and various grid sizes. For the particular grid $|\mathcal{S}| = 3722$, the lower bound probabilities of satisfying φ are shown in Figure 5.3.

We saturate conservative errors output by FAUST^2 that are greater than 1 to this value. As evident in Table 5.1, and Figure 5.3, our approach greatly outperforms

Tool **Method**	**Impl.** **Platform**	$\lvert\bar{\mathcal{S}}\rvert$ [states]	**Time taken** [s]	**Error** $\varepsilon_{\max}$
IMDP(KKT)	MATLAB	361	19.8	0.211
IMDP(GD)	C++	361	29.0	0.211
FAUST^2	MATLAB	361	108.3	1.000
FAUST^2	C++	361	136.7	1.000
IMDP(KKT)	MATLAB	625	145.6	0.163
IMDP(GD)	C++	625	117.7	0.163
FAUST^2	MATLAB	625	285.8	1.000
FAUST^2	C++	625	302.9	1.000
IMDP(KKT)	MATLAB	1444	4464.8	0.109
IMDP(GD)	C++	1444	510.9	0.109
FAUST^2	MATLAB	1444	1445.4	1.000
FAUST^2	C++	1444	1201.9	1.000
IMDP(KKT)	MATLAB	2601	28127.3	0.082
IMDP(GD)	C++	2601	2939.1	0.082
FAUST^2	MATLAB	2601	5274.6	0.995
FAUST^2	C++	2601	3305.5	0.995
IMDP(KKT)	MATLAB	3721	Time out	-
IMDP(GD)	C++	3721	3973.3	0.068
FAUST^2	MATLAB	3721	11285.3	0.832
FAUST^2	C++	3721	7537.8	0.832

Table 5.1: Comparison of verification results of our IMDP algorithms against FAUST^2 for φ with $K = 2$. The maximal computational time before a time out is 9 hours.

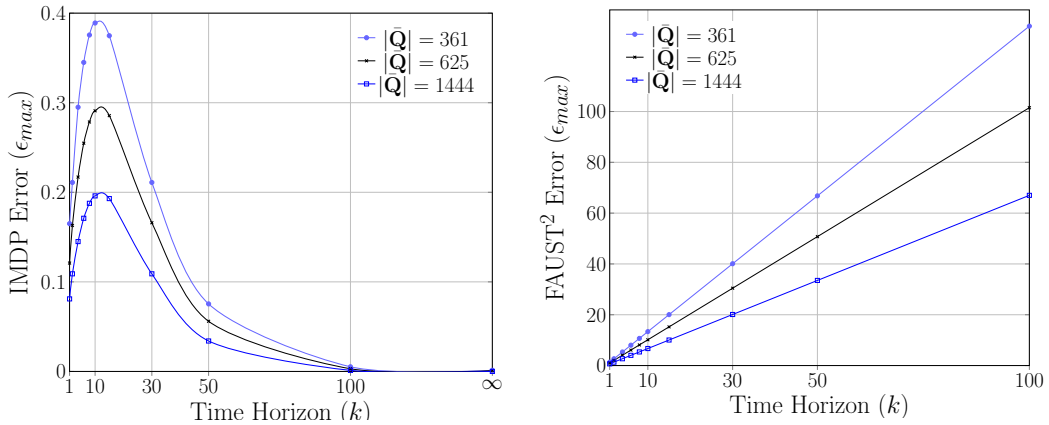


(a) IMDP **(b)** FAUST^2

Figure 5.3: Lower bound probabilities for satisfying φ computed using (a) IMDP and (b) FAUST^2 with $|\mathcal{S}| = 3722$ states.

the state of the art: the IMDP method has significantly (an order of magnitude) smaller error than FAUST^2, while also requiring lower computation times, for the same grid size. We also note that, as guaranteed by the theory (Theorem 5.2.1 and Proposition 2), both KKT and GD approaches compute the same error.

In Figure 5.4, we show the error of each method as a function of the time horizon K in φ. From these figures, it is evident that our approach again greatly outperforms FAUST^2. The IMDP method embeds the error in the abstraction and performs computations according to feasible transition probabilities, which prevents the error from exploding over time, whereas the error of FAUST^2 keeps increasing monotonically with the time horizon. An interesting aspect in Figure 5.4a is that the error of our method goes to zero as K increases. This is a consequence of the system under consideration being an unbounded Gaussian process, and despite its stable dynamics, the probability of it remaining within the bounded set X approaches zero as time grows larger. This is meaningfully captured by the upper and lower probability bounds of our method. On the other hand, FAUST^2 is not able to capture this behaviour and its error explodes.



(a) IMDP **(b)** FAUST^2

Figure 5.4: Maximum error incurred in satisfying φ as a function of the time horizon K.

5.4 Solving the BAS case studies

In this section, we show how we solve the BAS case studies presented in Section 4.3, using the methods and frameworks devised in this chapter. All the experiments are run on a standard laptop, with an Intel Core i7-8550U CPU at 1.80GHz × 8 and with 8 GB of RAM.

5.4.1 Two zone heating setup with stochastic dynamics

In this case study, we are tasked with quantifying the probability of remaining within a safe set over a fixed time horizon via model checking. The safety requirement is defined using the logical formula $\varphi_1 := \mathrm{P}_{=?}\left(\mathbf{G}^{\leq K=6}\ X_{safe}\right)$ and the dynamics are described using SHS model consisting of four continuous variables which evolve according to $\mathbf{M}_s$, given by (4.10), one discrete mode and one continuous control input evolving over $\mathbb{R}$. We can solve this requirement by performing probabilistic reachability analysis using either of the two formal abstraction methods.

Probabilistic reachability using Markov Decision Processes First, we opt to perform this analysis by abstracting $\mathbf{M}_s$ into an MDP, in view of the continuous input variable. More concretely, we make use of FAUST^2 [147] to perform probabilistic reachability analysis of $\mathbf{M}_s$ by defining the same safe set and assuming an input set of [15 30]. $\mathbf{M}_s$ is first abstracted via uniform gridding to obtain the discretised transition kernel and then the maximal safety probability for each partition is computed recursively over the whole time horizon $K = 6$. However, due to the large number of continuous variables ($n = 4$), we obtain an abstraction that requires 500 GB of memory to compute, making it computationally infeasible. To mitigate this, we note that the variables T_{rw,r_1}, T_{rw,r_2} , have negligible effect on the evolution of the zone temperatures (i.e. the variables of interest). Hence, we abstract $\mathbf{M}_s$ such that $x_s^T = [T_{z_1}\ T_{z_2}]^T$ (the states of interest and we fix the rest to steady-state). The resulting model $\mathbf{M}_{s'}$ is given in Appendix A.1. Based on this model, we construct a stochastic kernel taking the form of a Gaussian conditional distribution where $T_x = \mathcal{N}(\cdot|A_{s'}x_s + B_{s'}u_s + Q_{s'}, \Sigma_{s'})$. The input set is retained, while the safe set is defined as $X_{safe} = [20\ 24] \times [20\ 24]$.

We perform the abstraction using both uniform and adaptive-sequential partitioning of the safe set. Figure 5.5a and 5.5b represent the probability of satisfaction for each cell within the generated grid via uniform and adaptive-sequential gridding respectively, for the same abstraction error. The quantitative results of the abstraction are given in Table 5.2. It can be noted that using the adaptive and

sequential method, we end up with a significantly smaller number of states and in this instance lower computational time as a consequence. One should note, that we employ a modified version of the adaptive and sequential gridding method, where rather than defining a single coarse grid over which we refine until we reach ε. We generate multiple coarse grids sequentially and at each step reduce the required abstraction error until we reach the desired level. More specifically, we first generate a grid with $10\varepsilon_{max}$, refine this to $\{8\varepsilon_{max}, 4\varepsilon_{max}, 2\varepsilon_{max}\}$ until we reach a global abstraction error of ε_{max}. For this model, this resulted in a small number of states being required and hence shorter computation time. In Table 5.2, we saturate the abstraction error being generated to 1. This is a consequence of the abstraction requiring a significantly large number of states for small abstraction error.

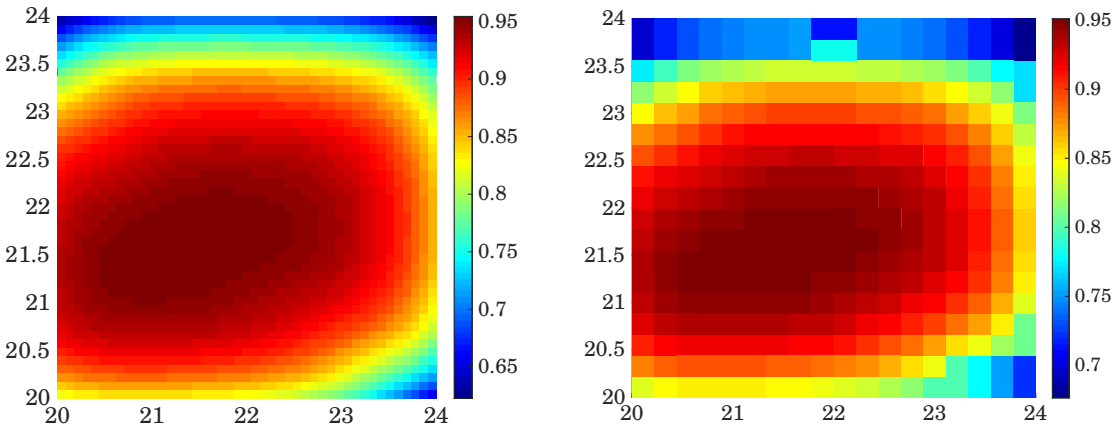


Figure 5.5: Partition of the safe set for $\mathbf{M}_{s'}$, along with the optimal safety probability for each partition set when using (a) uniform and (b) adaptive and sequential gridding.

Method	$\lvert\mathcal{S}\rvert$ [states]	$\lvert\mathcal{A}\rvert$ [actions]	**Time taken** [s]	**p(s)**	**Error** ε_{max}
Uniform	2025	19	15330.9	≥ 0.60	1
Adaptive	302	8	224.6	≥ 0.65	1

Table 5.2: Quantitative results for computing probabilistic reachability problem via formal abstractions using MDP.

Probabilistic reachability using Interval Markov Decision Processes We repeat the process by transforming $\mathbf{M}_s$ into its steady state and transient components as follows

$$\mathbf{M}_s : \begin{cases} x_{ss} & = A_s x_{ss} + B_s u_{ss} + Q_s, \\ x_{tr}[k+1] & = A_s x_{tr,i}[k] + \Sigma_s^{0.5} e[k], \end{cases} \tag{5.40}$$

where $e[k] \sim \mathcal{N}(0, \mathbf{I})$, $x_{ss} = [22\ 22\ 35\ 35]^T$ is designed such that it lies in the middle of X_{safe} via the appropriate selection of the control variable u_{ss}. Note that, the transient part of $\mathbf{M}_s$, now takes the form of (5.5) and hence we can apply formal abstractions via IMDP. Consequently, we need to transform X_{safe} to operate over the transient region i.e. $X_{safe} = [-2\ 2] \times [-2\ 2] \times [-2\ 2] \times [-2\ 2]$.

We proceed by performing probabilistic reachability of φ_1 over the transformed X_{safe} set using the IMDP method. We execute this process over both the original four-dimensional model and the reduced two-dimensional model used for MDP abstractions (i.e. $\mathbf{M}_{s'}$). In both instances, we discretise the state space using a uniform grid with Δx partition for each dimension, over which we perform probabilistic reachability and compute the upper and lower bound probabilities of satisfying φ_1 within each region. For the four-dimensional model $\Delta x^T = [0.35\ 0.35\ 1\ 1]^T$, while for the two-dimensional model, we set $\Delta x^T = [0.20\ 0.20]^T$.

The quantitative results for both models are given in Table 5.3. The resulting partition of the safe set along with the lower bound safety probability for each partition set, for the two-dimensional model, is depicted in Figure 5.6. Note, we also perform this analysis using MDP abstractions on the transient states of $\mathbf{M}_{s'}$ using uniform gridding. This resulted in a grid consisting of 13924 states for a maximal abstraction error of 1 and computational time in the excess of 8 hours. Evidently, the IMDP outperforms the MDP in terms of computational times, abstraction errors and memory requirements.

Method	$\lvert\mathcal{S}\rvert$ [states]	**Time taken** [s]	$\check{\mathbf{p}}_{\varphi_1}(\mathbf{s})$	**Error** ε_{max}
IMDP ($\mathbf{M}_{s'}$)	484	56.9	≥ 0.55	0.16
IMDP ($\mathbf{M}_s$)	2132	1587.7	≥ 0.55	0.18

Table 5.3: Quantitative results for computing probabilistic reachability problem via formal abstractions using IMDP.

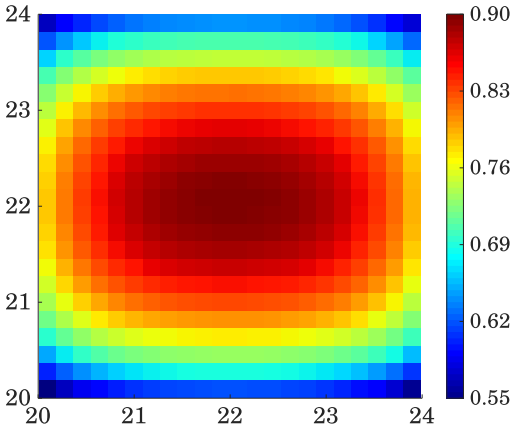
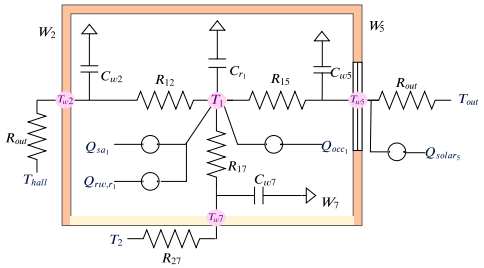


Figure 5.6: Partition of the safe set for $\mathbf{M}_{s'}$, along with the optimal safety probability for each partition set when using IMDP.

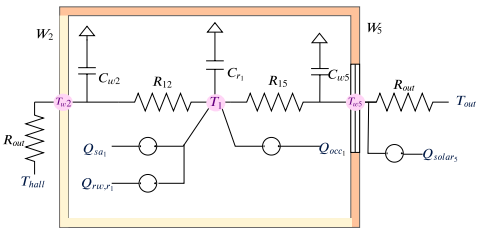
5.4.2 Two zone heating setup with large number of continuous variables

In this second case study, we have the task of synthesising the optimal strategy that ensures the zone temperature does not deviate from the desired temperature set-point, using the model described in Section 4.3.2. This is composed of one discrete mode and seven continuous variables that evolve according to (4.12). This seventh-order model often cannot be used due to associated computational complexity to compute control actions. However, we can note that the modelling structure can be transformed into the form of (5.4) and thus, we can employ policy synthesis via (ϵ, δ)-simulation relations (see Section 5.1.2). In this section, we show how the classical MDP-based abstraction methods can be applied to perform synthesis over a larger number of continuous variables.

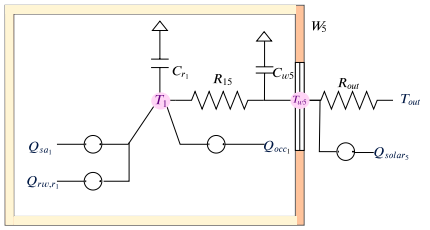
First, we need to construct a set of simpler models derived from this seventh-order model. We construct four lower-order models by performing model reduction based on assumptions made on the underlying physical quantities. Models could, in general, be obtained via numerical model-order reduction techniques which reduce the dimensionality of models with minimal loss in input-output accuracy [23], [78]. However key here, is that they maintain correspondences via the underlying physical variables. We depict the resulting set of models using Figure 5.7.



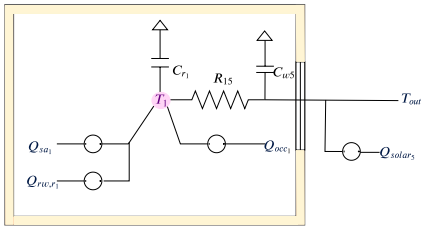
(a) 4^{th} order model ($\mathbf{M}_{s,4}$) representing dynamics of zone 1. Measurement T_{z_2} is now considered as a stochastic disturbance.



(b) 3^{rd} order model ($\mathbf{M}_{s,3}$) representing dynamics of zone 1. Assuming negligible temperature difference between T_{z_2} and T_{z_1}.



(c) 2^{nd} order model ($\mathbf{M}_{s,2}$) representing dynamics of zone 1. Assuming negligible temperature difference between T_{z_1}, the surrounding walls and T_{hall}.



(d) 1^{st} order model ($\mathbf{M}_{s,1}$) representing dynamics of zone 1. Assuming steady-state wall dynamics and only considers heat exchange with the outside air. Here, $C_r = C_{z_1} + C_{w_2}$.

Figure 5.7: Set of thermal models and the corresponding analogous RC circuits. The models states are highlighted using pink nodes and the stochastic disturbances are in blue. Here $Q_g = Q_{rw,r_1} + Q_{sa}$.

The fourth-order model of the temperature in zone 1 ($\mathbf{M}_{s,4}$), neglects the dynamics of zone 2 and replaces its state by a stochastic signal T_{z_2} modelled as an i.i.d. Gaussian distribution having a mean of 22 $°C$ and unit variance. Namely, independent realisations $T_{z_2}[k] \sim \mathcal{N}(20, 1)$, as can be seen in Figure 5.7a. The model comprises variables $x_{s,4} = (T_{z_1}, T_{w_5}, T_{w_2}, T_{w_7})$, which represent the inner and outer wall temperatures T_{w_5}, T_{w_2} of the zone, the temperature of the wall which separates the two neighbouring zones T_{w_7} and the indoor zone temperature T_{z_1}. Similarly to the original model it is affected by several stochastic sources represented using the variables $w_{s,4} = (T_{out}, T_{hall}, CO_{2_{z_1}}, T_{rw,r_1}, T_{z_2})$. We describe the model by

$$\mathbf{M}_{s,4} : \begin{cases} x_{s,4}[k+1] & = A_{s,4}x_{s,4}[k] + B_{s,4}u_{s,4}[k] + G_{s,4}w_{s,4}[k] + Q_{s,4}, \\ y[k] & = C_{s,4}x_{s,4}[k]. \end{cases} \tag{5.41}$$

The third-order model of the temperature in zone 1 ($\mathbf{M}_{s,3}$), shown in Figure 5.7b, is constructed by assuming negligible temperature difference between the two neighbouring zones $T_{z_2} = T_{z_1}$. $\mathbf{M}_3$ has $x_{s,3} \in \mathbb{R}^3, x_{s,3} = (T_{z_1}, T_{w_5}, T_{w_2})$ and four major sources of stochastic disturbance $w_{s,3} = (T_{out}, T_{hall}, CO_{2_{z_1}}, T_{rw,r_1})$. We represent the model by

$$\mathbf{M}_{s,3} : \begin{cases} x_{s,3}[k+1] & = A_{s,3}x_{s,3}[k] + B_{s,3}u_{s,3}[k] + G_{s,3}w[k] + Q_{s,3}, \\ y[k] & = C_{s,3}x_{s,3}[k]. \end{cases} \tag{5.42}$$

The second-order model of the temperature in zone 1 ($\mathbf{M}_{s,2}$), depicted in Figure 5.7c, assumes that the inner walls have a similar temperature to the zone temperature and $T_{hall} = T_{z_1} = T_{w_2}$. The model includes variables $x_{s,2} \in \mathbb{R}^2, x_{s,2} = (T_{z_1}, T_{w_5})$ and considers three major sources of stochastic disturbances $w_{s,2} = (T_{out}, CO_2, T_{rw,r_1})$

$$\mathbf{M}_{s,2} : \begin{cases} x_{s,2}[k+1] & = A_{s,2}x_{s,2}[k] + B_{s,2}u_{s,2}[k] + G_{s,2}w_{s,2}[k] + Q_{s,2} \\ y[k] & = C_{s,2}x_{s,2}[k]. \end{cases} \tag{5.43}$$

Lastly, the first-order model representing the temperature in zone 1 ($\mathbf{M}_{s,1}$) no longer considers dynamics of any of the walls and only takes into account the heat exchange with the outside air. The analogous RC circuit is shown in Figure 5.7d and it is described by

$$\mathbf{M}_{s,1} : \begin{cases} x_{s,1}[k+1] & = A_{s,1}x_{s,1}[k] + B_{s,1}u_{s,1}[k] + G_{s,1}w_{s,1}[k] + Q_{s,1}, \\ y[k] & = C_{s,1}x_{s,1}[k]. \end{cases} \tag{5.44}$$

The model comprises $x_{s,1} = T_{z_1}$ and considers three major sources of stochastic disturbance to the system $w_{s,1} = (T_{out}, CO_{2_{z_1}}, T_{rw,r_1})$, which act as an additional source of heat gain in the zone. The internal heat storage capacity of the zone encompasses both the heat stored by the walls and the zone itself and, thus, the new capacitance is $C_r = C_{z_1} + C_{w_2}$ [91]. Note, all the parametrised matrices $A_{s,i}, B_{s,i}, C_{s,i}, G_{s,i}, Q_{s,i}$, $i \in \{4, 3, 2, 1\}$ are detailed in the Appendix A.2. In all the given model descriptions, the output space of the model represents the temperature T_{z_1}. We introduce a norm on the output space y. For a given set $\mathbb{X}$ a metric or distance function $\mathbf{d}_{\mathbb{X}}$ is a function $\mathbf{d}_{\mathbb{X}} : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_{\geq 0}$. So the output space $y \in \mathbb{Y} \subset \mathbb{R}$ is then endowed with the Euclidean norm $\mathbf{d}_{\mathbb{Y}} = \|\cdot\|$.

To apply the (ϵ, δ)-simulation relations framework, we need to transform (4.12) into the form of

$$\mathbf{M} : \begin{cases} x[k+1] & = Ax[k] + Bu[k] + Gw[k], \\ y[k] & = Cx[k], \end{cases} \tag{5.45}$$

Consequently, for all the models $\mathbf{M}_{s,i}$, $i \in \{7, 4, \ldots, 1\}$, we split the states of the model into a steady state (x_{ss}) and a transient (x_{tr}) component, as follows

$$\mathbf{M}_{s,i} : \begin{cases} x_{ss,i} & = A_{s,i}x_{ss,i} + B_{s,i}u_{ss,i} + G_{s,i}w_{m,i} + Q_{s,i}, \\ x_{tr,i}[k+1] & = A_{s,i}x_{tr,i}[k] + B_{s,i}u_{tr,i}[k] + G_{s,i}\Sigma_{w,i}^{0.5}e[k], \\ y[k] & = C_{s,i}x_{ss,i} + C_{s,i}x_{tr,i}[k]. \end{cases} \tag{5.46}$$

Here, $u_{ss,i}$ represents the nominal input required to achieve the desired steady-state temperature of $T_{z_1} = T_{sp} = 22$ [$^\circ C$], $u_{tr,i}$ is the additional input required from the nominal one to reach a new desired temperature, $w_{m,i}$ represents the mean noise signal, e is white noise described by $e[k] \sim \mathcal{N}(0, \mathbf{I})$, $y_{ss,i}$ is the steady-state output, and $y_{tr,i}[k]$ is the output of the transient state. In order to avoid ill-conditioning, the input signal u is rescaled such that a full heating input $T_{sa} = 30$ [$^\circ C$] corresponds to value 1. We further restrict the input set with an upper bound $c_w = \frac{1}{30}$ that ensures an absolute deviation of 1 [$^\circ C$].

We consider pairs of concrete and reduced order models: ($\mathbf{M}_{s,7}$, $\mathbf{M}_{s,4}$), ($\mathbf{M}_{s,7}$, $\mathbf{M}_{s,3}$), ($\mathbf{M}_{s,7}$, $\mathbf{M}_{s,2}$), ($\mathbf{M}_{s,7}$, $\mathbf{M}_{s,1}$) and we will index all the model pairs using $i = \{4, \ldots, 1\}$ respectively (corresponding to the model order of the simpler model in the pair). We repeat the process of computing the (ϵ, δ)-relations for each pair of models, and obtain the model pair that provides the best (ϵ, δ) trade-off. In other words, we identify the abstract model that simulates the seventh-order concrete model with the best guarantees. For the identified model pair, we then synthesise and refine policies such that we attain the maximal probability of satisfying

$$\varphi_2 := \mathrm{P}_{\geq p}\left(\mathbf{G}^{\leq k=9}\left[|T_{sp} - T_{z_1}| \leq 1\right]\right) = \mathrm{P}_{\geq p}\left(\mathbf{G}^{\leq k=9}[|y| \leq 1]\right).$$

Computation of (ϵ, δ) relation for pairs of models The (ϵ, δ)-approximate simulation relations are computed over a range of logarithmically spaced points $\delta \in [0.001, 1]$ for which the corresponding ϵ values are computed. For succinctness,

the reader is referred to [71] for the computational algorithm's required to compute the (ϵ, δ) pair. This process is repeated for all the model pairs and we list the resulting trade-off for parameters (ϵ, δ) in the simulation relation in Table 5.4. Note that with decreasing model order, ϵ increases: this is due to the assumptions taken to construct the different levels of abstractions. For instance, recall that to construct the third-order model from the second-order model, we assume that the inner walls between the two zones have a similar temperature. This assumption reduces the model complexity but consequently increases the output deviation, as witnessed from the obtained ϵ values. Furthermore, for increasing values of δ, ϵ decreases to a positive lower bound which is a function of the bounded input set.

δ	1	$10^{\frac{-1}{3}}$	$10^{\frac{-2}{3}}$	10^{-1}	$10^{\frac{-4}{3}}$	$10^{\frac{-5}{3}}$	10^{-2}	$10^{\frac{-7}{3}}$	$10^{\frac{-8}{3}}$	10^{-3}
$\mathbf{M}_{s,4}$	0.0043	0.0674	0.0809	0.0909	0.0993	0.1066	0.1132	0.1192	0.1248	0.1301
$\mathbf{M}_{s,3}$	0.0387	0.1063	0.1206	0.1315	0.1404	0.1482	0.1553	0.1617	0.1678	0.1734
$\mathbf{M}_{s,2}$	0.0613	0.1230	0.1363	0.1461	0.1542	0.1613	<u>0.1678</u>	0.1737	0.1792	0.1843
$\mathbf{M}_{s,1}$	0.0184	0.1426	0.1690	0.1886	0.2049	0.2192	0.2320	0.2438	0.2548	0.2651

Table 5.4: Trade-off between (ϵ, δ) for concrete $(\mathbf{M}_{s,7})$ and abstract model pairs $(\mathbf{M}_{s,i}, i = 4, \dots 1)$.

We are interested in selecting a low-order model that introduces the least errors in the specification and is computationally feasible. Based on these outcomes we proceed with the pair $(\mathbf{M}_{s,7}, \mathbf{M}_{s,2})$ and choose the parameters pair $(\epsilon, \delta) = (0.1678, 10^{-2})$.

Controller synthesis for $\mathbf{M}_{s,2}$ We solve the safety property to φ_2 for the concrete model. For the abstract model we modify it as follows

$$\varphi 2_{(\epsilon,\delta)} := \mathrm{P}_{\geq p+\gamma}\left(\mathbf{G}^{\leq K=9}|y| < 1-\epsilon\right) = \mathrm{P}_{\geq p+0.09}\left(\mathbf{G}^{\leq K=9}|y| \leq 0.8322\right).$$

Here γ gives the accumulation of the error in probability over the time horizon of interest and is obtained from $1-\gamma = (1-\delta)^9$ then $\gamma \leq 9\delta$. Note that y is replaced with the computation of $y_{tr,2}$ by using (5.46). This is done since we are only interested in computing the deviations in fluctuations from the nominal zone temperature $y_{ss,2}$. Policy synthesis over the simpler model $\mathbf{M}_{s,2}$ is performed by employing heavily optimised algorithms in FAUST^2 [147]. Note, we optimised the

algorithms via the use of sparse matrices and by splitting the abstraction into deterministic and stochastic parts via decoupling of the noise. Subsequently, a stochastic dynamic programming scheme is set up such that the final value function provides the maximal probability p for the specification $\varphi_{2_{(\epsilon,\delta)}}$. This grid-based computation of the safety probability over the 9 times steps of the formula leads to a MDP with $|\mathcal{S}| = 2445$ states and $|\mathcal{A}| = 2035$ discrete actions, an overall accuracy of 0.13 and a total computation time of 200.8 $[s]$. Over this approximation, we compute the optimal policy $\pi^*_{(\epsilon,\delta)}$. This results in $\varphi_{2_{(\epsilon,\delta)}}$ being satisfied with a probability of at least $p^{\pi^*}_{\varphi_{2_{(\epsilon,\delta)}}} = p^{\pi^*}_{(\epsilon,\delta)} - \varepsilon_{\max} = 0.9997 - 0.13 = 0.8697$ for the reduced order model $\mathbf{M}_{s,2}$ initialised at origin (zero fluctuation in air temperature in the zone). The 0.13 error is being introduced due to the gridding in FAUST^2.

Controller refinement for $\mathbf{M}_{s,7}$ The resulting policy $\pi^*_{(\epsilon,\delta)}$ is refined back to π^* via the (ϵ, δ) simulation relations. The refinement procedure follows algorithm in [71] and refines the original policy into a new policy which ensures that events defined over the output space have equal probability. This results in the safety probability for $\mathbf{M}_{s,7}$, initialised at the origin, being satisfied with a lower bounded probability of $p^*_{\varphi_2} = p^{\pi^*}_{\varphi_{2_{(\epsilon,\delta)}}} - 9\delta = 0.8697 - 0.09 = 0.7797$ with the inner air temperature fluctuations remaining in the bound between $[-1, 1]$.

Experiments

We show the advantages of employing (ϵ, δ)-simulation relations over the current state of the art where certified policies for complex models are synthesised directly. We make use of the optimised algorithms from FAUST^2 to synthesise the optimal policy that satisfies the specification φ_2 directly on all the models $(\mathbf{M}_{s,7}, \mathbf{M}_{s,4}, \mathbf{M}_{s,3}, \mathbf{M}_{s,2})$, while we make use of the (ϵ, δ)-simulation relations framework to synthesise and refine the policies for the same set of models. For our framework, we compute the policies using the following pair of models $(\mathbf{M}_{s,7}, \mathbf{M}_{s,2}), (\mathbf{M}_{s,4}, \mathbf{M}_{s,2})$, $(\mathbf{M}_{s,3}, \mathbf{M}_{s,2}), (\mathbf{M}_{s,2}, \mathbf{M}_{s,1})$ and the refined policies must

satisfy the following specification

$$\varphi_2 := \mathrm{P}_{\geq p+9\delta-0.13}\left(\mathbf{G}^{\leq K=9}|y| \leq 1\right).$$

The policies for the abstract models are computed using algorithms from FAUST^2. In both cases, the abstraction procedure performed within FAUST^2 introduces an error of 0.13 in the overall accuracy of the achieved probability of satisfying φ_2. We record the total time taken to compute the policies, in seconds, the amount of memory required to store the gridded state-space and the probability of satisfiability of φ_2 when using the directly synthesised policy or the refined policy. The results are shown in Table 5.5. One should note that when using FAUST^2 directly, one needs to tighten the bounds of the state-space such that the gridding of the underlying model is of sufficiently small size which allows the synthesis procedure to be performed. This, however, comes at the expense of the overall probability of satisfying the specification φ_2 (cf. Table 5.5). For $\mathbf{M}_{s,7}$ the policy could not be computed directly due to large memory usage required: this highlights the advantages of using the (ϵ, δ)-simulation relations framework where policies for higher complex order models can be achieved with good certificates.

Model Order of $\mathbf{M}_{s,i}$	**Method**	**Time taken** $[s]$	**Memory** $[\lvert\mathcal{S}\rvert \times \lvert\mathcal{A}\rvert]$	$\mathbf{p}^{\pi^*}_{\varphi_2}$
7	Refinement, $\mathbf{M}_{s,2}, (\epsilon, \delta) = (0.1678, 0.01)$	200.8	4975575	0.7797
4	Refinement, $\mathbf{M}_{s,2}, (\epsilon, \delta) = (0.1210, 0.01)$	192.7	4384236	0.7320
3	Refinement, $\mathbf{M}_{s,2}(\epsilon, \delta) = (0.0641, 0.01)$	125.9	3313750	0.7466
2	Refinement, $\mathbf{M}_{s,1}, (\epsilon, \delta) = (0.1698, 0.01)$	108.2	3218436	0.6796
7	Directly	-	-	-
4	Directly	974.8	9148370	0.5050
3	Directly	390.3	5753100	0.6022
2	Directly	97.3	3203460	0.7212

Table 5.5: Comparison between computing the policies directly using the high order model vs computing the refined policies using the (ϵ, δ)-simulation relations framework.

5.4.3 Air quality model capturing CO_2 and temperature dynamics in zone

We are interested in synthesising a switching strategy that maximises the probability p of satisfying

$$\varphi_3 := \mathrm{P}_{\geq p}\left(\neg X_{safe}\ \mathbf{U}^{\leq K=32}\ X_{target}\right).$$

where $K = 8 \times \Delta = 32$ steps, $X_{unsafe} = [450,\ 460] \times [20,\ 24]$ and $X_{target} = [400,\ 450] \times [20,\ 24]$, for the SHS described using (4.17). To recapitulate, the SHS consists of four discrete modes,

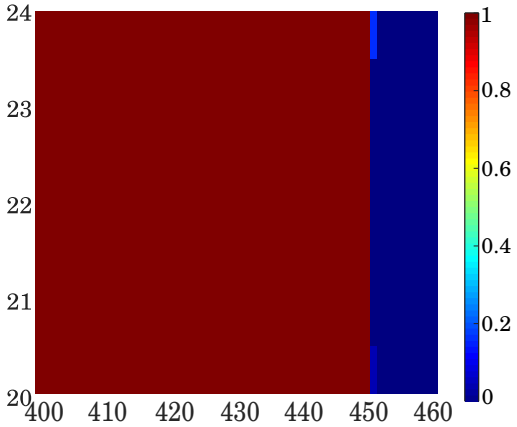
$$\{q_0 = (\neg F_{en}, \neg W_{en}), q_1 = (F_{en}, W_{en}), q_2 = (\neg F_{en}, W_{en}), q_3 = (F_{en}, \neg W_{en})\}$$

each of which effect the evolution of two continuous variables corresponding to the CO_2 and zone temperature and whose dynamics are described using (4.16).

To solve this synthesis problem, we opt to use formal abstractions via IMDPs and employ the framework described in Section 5.2.5. This is in view of the computational complexity that would be required if abstractions via MDP are employed. We employ a fixed grid with $\Delta x^T = [0.5\ 1]^T$ partition for each dimension, to generate the abstraction. Over the abstraction, we run the synthesis algorithm to obtain the robust strategy $\pi^*_{\varphi_3}$ with the corresponding lower probability bounds.

The total time to compute the abstraction and to generate $\pi^*_{\varphi_3}$ is 951.6 $[s]$ with abstraction error $\varepsilon_{\max} = 0.13$. The generated IMDP has $|\mathcal{S}| = 1920$ states with $|\mathcal{S}^{q_i}| = 480, \forall i = \{0, \ldots, 3\}$ and one s_{u}. Figure 5.8a shows the discretisation of mode q_0 together with the lower probability bounds of satisfying φ_3. Similar, results are obtained for the remaining modes. One can note a high probability of remaining within the safe set, while avoiding the target set over the fixed time horizon of 32 time steps. Table 5.8b summarises the IMDP abstraction results.

The resulting strategy takes the form of a look-up table for each cell within the gridded hybrid state space and is also a function of the time horizon. We depict the resulting look-up tables for mode q_0 at time instances $k = 1$ and $k = 32$ within Figure 5.9. Similar look-up tables are obtained for the remaining modes. The optimal policy has an intuitive translation: for lower zone temperature, the



(a)

Method	$\lvert\mathcal{S}\rvert$ [states]	$\lvert\mathcal{S}^{q_i}\rvert$ $i = \{0, \ldots, 3\}$	Time taken [s]	Error ε_{max}
IMDP	1920	480	951.6	0.13

(b)

Figure 5.8: Solving the synthesis problem using formal abstractions with IMDP: (a) lower bound probabilities of satisfying φ_3 for mode q_0 and (b) the quantitative results.

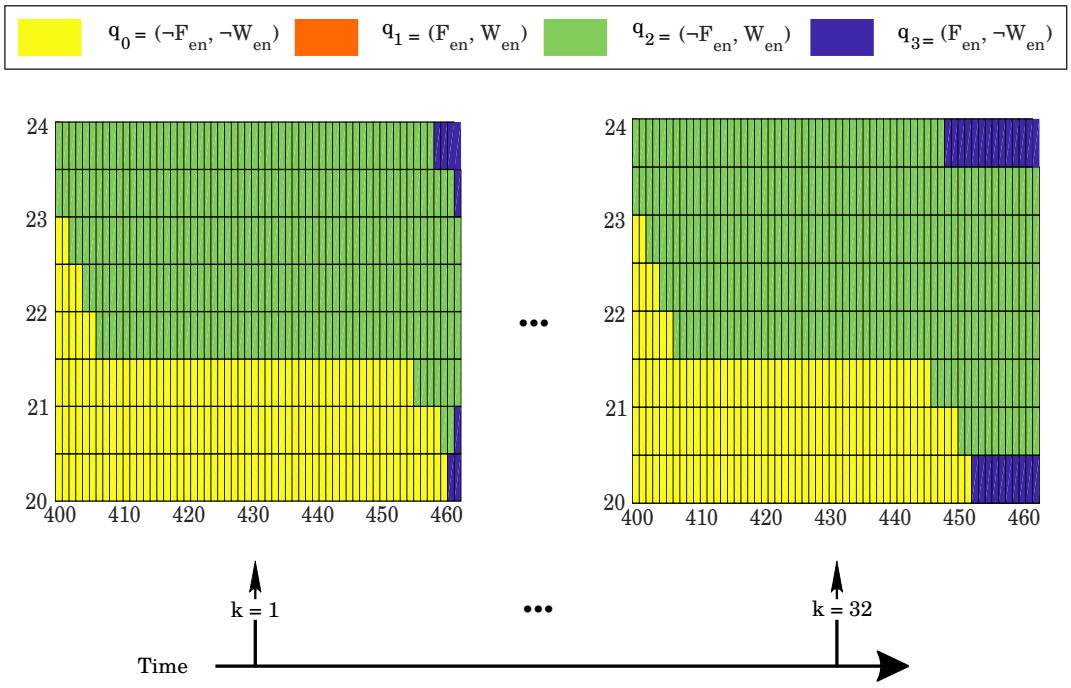


Figure 5.9: The synthesised control policy taking the form of a look-up table, where for each point in time and cell in the state-space the optimal action is stated. Here, we show the look-up table for mode q_0. We have similar look-up tables for q_i, $i \in \{1, \ldots, 3\}$.

optimal action is when both zone fan and windows and doors are closed. This is a consequence of the gradual exchange of temperature and CO_2 to the outside and neighbouring zones. However, as the zone temperature increases the accumulation of the CO_2 levels increases at a rate greater than the exchange with the outside.

This is due to the coupling between the two variables via the temperature heat gain $Q_{occ_1}(t) = \mu CO_{2_{z_1}}(t) + \beta_1$ in (4.15). Consequently, the optimal action is to open the windows and doors. Finally, when both the CO_2 levels in the zone are outside the target set (or conversely within the unsafe set) and the rate of air-flow from the window is not sufficient in reducing the CO_2 levels, the optimal action is to turn on the fan which has a faster rate of airflow and hence will be able to reduce the accumulation of CO_2 faster.

5.5 Conclusion

This chapter has put forward the techniques for performing verification and strategy synthesis of SHS via formal abstractions. The focus of this chapter has been on the study of formal abstractions taking the form of MDPs or IMDPs. More specifically, for formal abstractions taking the shape of MDPs, we show how strategy synthesis can be extended for models with larger number of continuous variables. While for abstractions via IMDPs, we present novel algorithms for computing abstractions which are both scalable and efficient. This chapter further compares the two abstraction methods and solves the verification and synthesis goals for the case studies presented in Chapter 4. One should note that all three case studies employed reachability-like properties within the analysis, in order to align with the low-level performance metrics of interest. However, performing abstractions via IMDPs can be extended to cater for reward-based properties. Such properties can be translated into the equivalent DFA [35] and thus can be used within the abstraction procedure directly.

6 Implementation of StocHy

Contents

This chapter presents a novel software tool called StocHy, for the quantitative analysis of discrete-time *stochastic hybrid systems* (SHS). It integrates the framework for stochastic processes presented in Chapter 4, together with the abstraction algorithms delineated in Chapter 5, with the aim of simplifying the modelling and analysis of SHS and thus make them attractive to a wider audience.

StocHy accepts a high-level description of stochastic models and constructs an equivalent SHS model. The tool enables users to (i) simulate the SHS evolution over a given time horizon; and to (ii) automatically construct formal abstractions of the SHS. Abstractions are then employed for formal verification or strategy (policy, control) synthesis. StocHy allows for modular modelling, and has separate simulation, verification and synthesis engines, which are implemented as independent libraries. The tool is implemented in C++ and employs manipulations based on vector calculus, sparse matrices, symbolic construction of probabilistic kernels, and multi-threading. Experiments show StocHy's marked improved performance when compared to existing abstraction-based approaches: in particular, StocHy beats state-of-the-art tools in terms of precision (abstraction error) and computational effort, and finally attains scalability to large-sized models (13 continuous dimensions). StocHy participated in the annual Applied Verification for Continuous and Hybrid Systems (ARCH) competition [1] within the *Stochastic Modelling* group, further highlighting its high perfomance and scalability. StocHy is available at

`www.gitlab.com/natchi92/StocHy`.

This chapter is structured as follows: Section 6.1 crisply delineates related software tools that can handle classes of SHS. Next, we introduce software features in Section 6.2 and provide an overview of the implementation of StocHy in Section 6.3. We highlight the use of StocHy by a set of experimental evaluations in Section 6.4: we provide four different case studies that emphasise the applicability, ease of use, and scalability of StocHy. Details on executing all the case studies are detailed in this chapter and within a Wiki page that accompanies the StocHy distribution. We further provide details on how to execute all the case studies (see Section 4.3) presented in this thesis within the Wiki page.

6.1 Related tools

In the following, we identify the (few) existing tools that can handle (classes of) SHS. The tools stress the need for a new software that allows for (i) straightforward

and general SHS modelling construction and (ii) scalable automated analysis.

FAUST^2 [147] Of much inspiration for StocHy, the tool *Formal Abstractions of Uncountable-STate STochastic processes* (FAUST^2) generates formal abstractions for uncountable-state discrete-time stochastic processes and performs verification of reachability-like properties and corresponding synthesis of policies. It natively supports SHS models with a single discrete mode and finite actions; and is naïvely implemented in MATLAB, resulting in a lack of scalability to large models.

SReachTools [161] An open-source MATLAB toolbox that handles discrete-time continuous state linear dynamical systems with additive stochastic disturbance. It performs verification (and analogously synthesis) for stochastic reachability via statistical methods and approaches drawn from convex optimisation, Fourier transforms and computational geometry. Unlike, StocHy it does not handle SHS with multiple modes; but it handles dynamics and safety constraints that can be time-varying.

Modest Toolset [79] The MODEST TOOLSET supports the modelling and analysis of continuous-time SHS. It is composed of a modular framework centred around the stochastic hybrid automata formalism (PHA) [74], which combine probabilistic discrete-transitions with the deterministic evolution of continuous variables; and supports the JANI specification [34] which provides a variety of input languages and analysis back ends.

HYPEG [130] The Java-based library HYPEG is a simulator for hybrid Petri nets with general transitions [65] which combine discrete and continuous components with a possibly large number of random variables, whose stochastic behaviour follows arbitrary probability distributions. HYPEG uses time-bounded discrete-event simulation and well-known SMC techniques to verify complex properties, including time-bounded reachability [130].

6.2 Features

In this section, we highlight the key novelties and strengths of StocHy's implementation. To perform formal verification and strategy synthesis we embed the techniques presented within Chapter 5.

6.2.1 Formal abstractions into Markov decision processes

StocHy improves the classical abstraction methods (see Section 5.1) by:

1. employing sparse matrix representation to manipulate the transition probabilities, in order to attain faster generation of the abstraction and use in formal verification or strategy synthesis.
2. simplifying the input model description, a high-level state-space description is all that is need.
3. exploiting the use of symbolic kernels.
4. automatically performing verification of SHS with multiple modes.

6.2.2 Formal abstraction into Interval Markov decision processes

The finite state-space of IMDP is based on the procedures presented within Section 5.2.

StocHy :

1. provides a novel abstraction algorithm allowing for efficient computation of the abstract model
2. generates abstraction models with significantly smaller errors and to verify models with up to 13 continuous variables.
3. exploits the use of sparse matrices for the manipulation of the upper and lower bound probabilities.
4. automates the synthesis algorithm with the abstraction procedure and the temporal property of interest.
5. makes use of adaptive and sequential refining of the underlying abstraction.

6.2.3 Analysis via Monte Carlo simulations

In this work, we analyse a SHS model using Monte Carlo simulations.

StocHy :

1. generates Monte Carlo simulation trajectories for a fixed time horizon.
2. automatically generates statistics from the simulations in the form of histograms, visualising the evolution of both the continuous random variables and the discrete modes.

6.3 Overview of StocHy

6.3.1 Installation

StocHy is set up using the provided GET_DEP file found within the distribution package, which will automatically install all the required dependencies. The executable RUN.SH builds and runs StocHy. We also provide a docker container version [30] such that StocHy can be easily used on other operating systems.

6.3.2 Modularity

StocHy comprises independent libraries for different tasks, namely (i) MDP, (ii) IMDP, and (iii) simulator. This allows for seamless extensions of individual sub-modules with new or existing tools and methods. The function `performTask` acts as a multiplexer for calling any of the libraries depending on the input model and task specification.

6.3.3 Data structures

StocHy makes use of multiple techniques to minimise computational overhead. It employs vector algebra for efficient handling of linear operations, and whenever possible it stores and manipulates matrices as sparse structures. It uses the linear algebra library Armadillo [140], [141], which applies multi-threading and a sophisticated expression evaluator that has been shown to speed up matrix

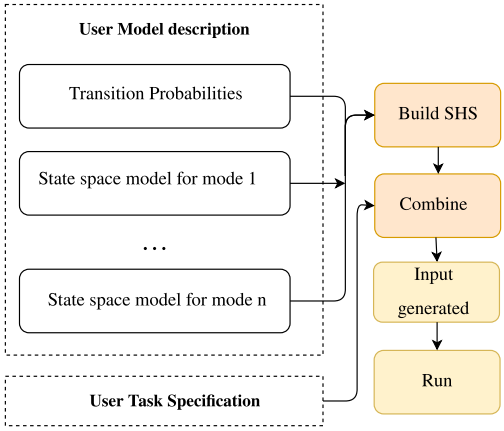
manipulations in C++ when compared to other libraries. MDP based abstractions define the underlying kernel functions symbolically using the library GiNaC [22], for easy evaluation of the stochastic kernels.

6.3.4 Input interface

The user interacts with StocHy via the MAIN file and must specify (i) a high-level description of the model dynamics and (ii) the task to be performed. The description of model dynamics can take the form of a list of the transition probabilities between the discrete modes, and of the state-space models for the continuous variables in each mode; alternatively, a description can be obtained by specifying a path to a MATLAB file containing the model description in state space form together with the transition probability matrix. Tasks can be of three kinds (each admitting specific parameters): simulation, verification, or synthesis. The general structure of the input interface is illustrated via an example in Listing 6.1: here the user is interested in simulating a SHS with two discrete modes $\mathcal{Q} = \{q_0, q_1\}$ and two continuous variables ($x \in \mathbb{R}^2$) evolve according to the stochastic difference equation, $x[k+1] = A(q_i)x[k] + G(q_i)w[k],\ i \in \{1, 2\}$ where $A(q_i),\ G(q_i)$ are properly sized matrices for each discrete mode and $w \sim \mathcal{N}(0, \mathbf{I})$. The model is autonomous and has no control actions.

The relationship between the discrete modes is defined by a fixed transition probability (line 1). The evolution of the continuous dynamics are defined in lines 2-12. The initial condition for both the discrete modes and the continuous variables are set in lines 13-18 (this is needed for simulation tasks). The equivalent SHS model is then set up by instantiating an object of type `shs_t<arma::mat,int>` (line 20). Next, the task is defined in line 25 (simulation with a time horizon $K = 32$, as specified in line 22 and using the simulator library, as set in line 24). We combine the model and task specification together in line 27. Finally, StocHy carries out the simulation using the function `performTask` (line 29).

```
arma::mat Tq = { {0.4, 0.6},{0.7,0.3}};     // Transition probabilities
// Evolution of the continuous variables for each discrete mode
// First model
arma::mat Aq0 = {{0.5, 0.4},{0.2,0.6}};
arma::mat Gq0 = {{0.4,0},{0.3, 0.3}};
ssmodels_t modelq0(Aq0, Gq0);
// Second model
arma::mat Aq1 = {{0.6, 0.3},{0.1,0.7}};
arma::mat Gq1 = {{0.2,0},{0.1, 0}};
ssmodels_t modelq1(Aq1, Gq1);
std::vector<ssmodels_t> models =
{modelq1,modelq2};
// Set initial conditions
// Initial state q_0
arma::mat q_init = arma::zeros<arma::mat>(1,1);
// Initial continuous variables
arma::mat x1_init = arma::ones<arma:mat>(2,1);
exdata_t data(x1_init,q_init);
// Build shs
shs_t<arma::mat,int> mySHS(Tq,models,data);
// Time horizon
int K = 32;
// Task definition  (simulator, mdp, imdp)
int lb = simulator;
taskSpec_t mySpec(lb,K);
// Combine
inputSpec_t<arma::mat,int> myInput(mySHS,mySpec);
// Perform task
performTask(myInput);
```



Listing 6.1: Description of MAIN file for simulating a SHS consisting of two discrete modes and two continuous variables evolving according to (5.5).

6.3.5 Output interface

We provide outputs as data files for all three libraries, which are stored within the RESULTS folder. We also provide additional PYTHON scripts for generating plots as needed. For abstractions based on MDP, the user has the additional option to export the generated MDP or DTMC to PRISM format, to interface with the popular model checkers [48], [76], [100] (StocHy prompts the user this option following the completion of the verification or synthesis task). As a future extension, we plan to export the generated abstraction models to the modelling format JANI [34].

6.4 Experimental evaluation

We apply StocHy on four different case studies highlighting different models and tasks to be performed. All the experiments are run on a standard laptop, with an Intel Core i7-8550U CPU at 1.80GHz × 8 and with 8 GB of RAM.

6.4.1 Case Study 1: Formal verification

We consider the following SHS which takes the form of (4.1), and has one discrete mode and two continuous variables $x = (CO_{2_{z_1}}, T_{z_1})$ representing the level of CO_2 and the ambient temperature respectively. The continuous variables evolve according to

$$CO_{2_{z_1}}[k+1] = CO_{2_{z_1}}[k] + \frac{\Delta}{V}\left(-m_{sa}CO_{2_{z_1}}[k] + \varrho_c(CO_{2,out} - CO_{2_{z_1}}[k])\right) + \sigma_1 w_1[k], \tag{6.1}$$

$$T_{z_1}[k+1] = T_{z_1}[k] + \frac{\Delta}{C_z}\left(m_{sa}C_{pa}(T_{sp} - T_{z_1}[k]) + \frac{\varrho_c}{R}(T_{out} - T_{z_1}[k])\right) + \sigma_2 w_2[k],$$

where Δ the sampling time $[min]$, V is the volume of the zone $[m^3]$, m_{sa} is the fixed mass air flow pumped inside the room $[m^3/min]$, ϱ_c is the natural drift air flow $[m^3/min]$, $CO_{2,out}$ is the outside CO_2 level $[ppm/min]$, T_{sp} is the desired temperature $[^oC]$, T_{out} is the outside temperature $[\ ^\circ C/min]$, C_z is the zone capacitance $[Jm^3/\ ^\circ C]$, C_{pa} is the specific heat capacity of air $[J/\ ^\circ C]$, R is the resistance to heat transfer $[\ ^\circ C/J]$, and $\sigma_{(\cdot)}$ is a variance term associated to the noise $w_{(\cdot)} \sim \mathcal{N}(0,1)$.

We are interested in verifying whether the continuous variables remain within the safe set $X_{safe} = [405, 540] \times [18, 24]$ over 45 minutes ($K = 3$). This property can be encoded as a BLTL property, $\varphi_1 := \mathrm{P}_{=?}\left(\mathbf{G}^{\leq K} X_{safe}\right)$. When tackled with the method based on MDP abstraction that hinges on the computation of Lipschitz constants, this verification task is numerically tricky, in view of difference in dimensionality of the range of $CO_{2_{z_1}}$ and T_{z_1} within the safe set X_{safe} and the variance associated with each dimension $G_{q_0} = \left[\begin{smallmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{smallmatrix}\right] = \left[\begin{smallmatrix}40.096 & 0\\ 0 & 0.511\end{smallmatrix}\right]$. The MDP is generated based on computation of the global Lipschitz constant, where the state space is partitioned along the longest end in order to try and obtain rectangular partitions. Given

Case study 1: Listings explaining task specification for (a) MDP and (b) IMDP.

```
1  // Dynamics definition
2  shs_t<arma::mat,int>
       myShs('../CS1.mat');
3  // Specification for FAUST^2
4  // safe set
5  arma::mat safe =
       {{405,540},{18,24}};
6  // max error
7  double eps = 1;
8  // grid type
9  // (uniform, adaptive)
10 int gridType = uniform;
11 // time horizon
12 int K = 3;
13 // task and property type
14 // (verify_safety ,
       verify_reach_avoid,
15 // synthesis_safety,
       synthesis_reach_avoid )
16 int p = verify_safety;
17 // library ( simulator, mdp,
       imdp)
18 int lb = mdp;
19 // task specification
20 taskSpec_t
       mySpec(lb,K,p,safe,eps,gridType);
```

Listing 6.2: (a) MDP

```
// Dynamics definition
shs_t<arma::mat,int>
    myShs('../CS1.mat');
// Specification for IMDP
// safe set
arma::mat safe
    {{405,540},{18,24}};
// grid size for each dimension
arma::mat grid = {{5.5,0.25}};
// relative tolerance
// for each dimension
arma::mat reft = {{1,1}};
// time horizon
int K = 3;
// task and property type
// (verify_safety ,
    verify_reach_avoid,
// synthesis_safety,
    synthesis_reach_avoid )
int p = verify_safety;
// library ( simulator, mdp,
    imdp)
int lb = imdp;
// task specification
taskSpec_t
    mySpec(lb,K,p,safe,grid,reft);
```

Listing 6.3: (b) IMDP

the difference in magnitude of the relative variance G_{q_0} along each dimension, the gridding procedure generates rectangular cells, where the dynamics of the shortest dimension (T_{z_1}) are lost since you have significantly fewer partitions along this dimension [146]. In order to mitigate this, StocHy automatically rescales the state space so all the dynamics evolve in a comparable range.

Implementation StocHy provides two verification methods, one based on MDP and the second based on IMDP. We parse the model from file CS1.MAT (see line 2 of Listings 6.2(a) and 6.3(b), corresponding to the two methods). CS1.MAT sets parameter values to (6.1) and uses a $\Delta = 15$ $[min]$. As anticipated, we employ both techniques over the same model description:

- for MDP we specify the safe set (X_{safe}), the maximum allowable error, the grid type (whether uniform or adaptive grid), the time horizon, together with the type of property of interest (safety or reach-avoid). This is carried out in lines 5-20 in Listing 6.2(a).

- for the IMDP method, we define the safe set (X_{safe}), the grid size, the relative tolerance, the time horizon and the property type. This can be done by defining the task specification using lines 5-20 in Listing 6.3 (b).

Finally, to run either of the methods on the defined input model, we combine the model and the task specification using `inputSpec_t<arma::mat,int>myInput (myShs,mySpec)`, then run the command `performTask(myInput)`. StocHy provides a summary of the generated results as the primary source of output within the command-line interface. The summary quotes the total number of states in the generated abstraction, the computational time taken and the abstraction error. This is followed by a prompt to store the verification results for both methods within the RESULTS directory:

- for MDP, StocHy generates four data files within the RESULTS folder: REPRESENTATIVE_POINTS.TXT contains the partitioned state space; TRANSITION_MATRIX.TXT consists of the transition probabilities of the generated abstract DTMC; PROBLEM_SOLUTION.TXT contains the sat probability for each state of the DTMC; and E.TXT stores the global maximum abstraction error. StocHy also prompts the user to (i) export the generated abstraction to PRISM or STORM and (ii) generate a figure showing the probability of satisfaction within the safe set.

- for IMDP, StocHy generates three data files in the same folder: STEPSMIN.TXT stores $\check{T}_s$ of the abstract IMDP; STEPSMAX.TXT stores $\hat{T}_s$; and SOLUTION.TXT contains the sat probability and the errors ϵ_s for each abstract state s. StocHy also prompts the user to generate a figure showing the lower-bound probability of satisfaction within the safe set.

Outcomes We perform the verification task using both MDP and IMDP abstractions generated via uniform gridding with different grid sizes. We further compare the outcomes of StocHy against the tool FAUST^2, which is implemented in MATLAB [147]. Note that the IMDP consists of $|\mathcal{S}|+1$ states, where the additional state is the sink state $s_u = \mathcal{D}\backslash X_{safe}$. The results are shown in Table 5.1. We saturate (conservative) errors output that are greater than 1 to this value and show the probability of satisfying the formula obtained from IMDP for a grid size of 3481 states in Figure 6.1 – similar probabilities are obtained for the remaining grid sizes.

Tool Method	**Impl. Platform**	$\lvert\mathcal{S}\rvert$ [states]	**Time** [s]	**Error** $\varepsilon_{\max}$
FAUST^2	MATLAB	576	186.7	1
MDP	C++	576	51.4	1
IMDP	C++	576	87.4	0.236
FAUST^2	MATLAB	1089	629.0	1
MDP	C++	1089	78.1	1
IMDP	C++	1089	387.9	0.174
FAUST^2	MATLAB	2304	2633.2	1
MDP	C++	2304	165.8	1
IMDP	C++	2304	1552.9	0.121
FAUST^2	MATLAB	3481	7523.7	1
MDP	C++	3481	946.3	1
IMDP	C++	3481	3623.1	0.098
FAUST^2	MATLAB	4225	10022.9	0.900
MDP	C++	4225	3313.9	0.900
IMDP	C++	4225	4854.6	0.089

Table 6.1: Case study 1: Comparison of verification results for φ_1 when using MDP vs IMDP abstractions.

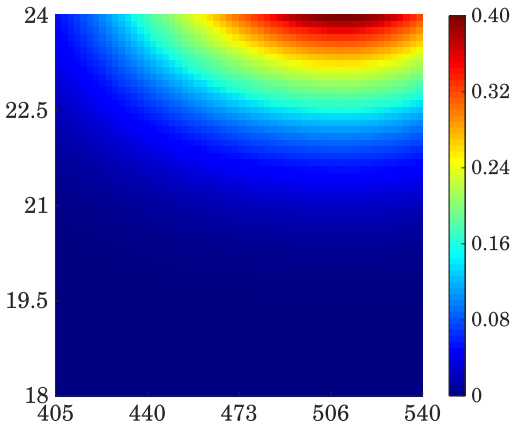


Figure 6.1: Case study 1: Lower bound probability of satisfying φ_1 generated using IMDP with 3481 states.

As evident from Table 5.1, the new IMDP method outperforms the approach using FAUST^2 in terms of the maximum error associated to the abstraction (FAUST^2 generates an abstraction error < 1 only with 4225 states). Comparing the MDP within StocHy and the original FAUST^2 implementation (running in MATLAB), StocHy offers computational speed-up for the same grid size. This is due to the faster computation of the transition probabilities, through StocHy's use of matrix manipulations. StocHy also simplifies the input of the dynamical model description: in the original FAUST^2 implementation, the user is asked to manually input the stochastic kernel in the form of symbolic equations in a MATLAB script. This is not

required when using StocHy as it automatically generates the underlying symbolic kernels from the input state-space model descriptions.

6.4.2 Case Study 2: Strategy synthesis

We consider a stochastic process with two modes $\mathcal{Q} = \{q_0, q_1\}$, which for q_0 continuously evolves according to

$$\begin{matrix} x_1[k+1] \\ x_2[k+1] \end{matrix} = \begin{bmatrix} 0.43 & 0.52 \\ 0.65 & 0.12 \end{bmatrix} \begin{matrix} x_1[k] \\ x_2[k] \end{matrix} + \begin{bmatrix} 1 & 0.1 \\ 0 & 0.1 \end{bmatrix} \begin{matrix} w_1[k] \\ w_2[k] \end{matrix},$$

and for q_1 according to

$$\begin{matrix} x_1[k+1] \\ x_2[k+1] \end{matrix} = \begin{bmatrix} 0.65 & 0.12 \\ 0.52 & 0.43 \end{bmatrix} \begin{matrix} x_1[k] \\ x_2[k] \end{matrix} + \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix} \begin{matrix} w_1[k] \\ w_2[k] \end{matrix}.$$

Consider the continuous domain shown in Figure 6.2a over both discrete locations. We plan to synthesise the optimal switching strategy $\pi^\star$ that maximises the probability of reaching the *green* region, whilst avoiding the *purple* one, over an unbounded time horizon, given any initial condition within the domain. This can be expressed with the CSLTL formula $\varphi_2 := (\neg \mathit{purple})\ \mathbf{U}\ \mathit{green}$, where the atomic propositions $\Theta = \{\mathit{purple}, \mathit{green}\}$ denote regions within the set $X = [-1.5, 1.5]^2$ (see Figure 6.2a).

Implementation We define the model dynamics following lines 3-14 in Listing 6.1, while we use Listing 6.3 to specify the synthesis task and together with its associated parameters. The CSLTL property φ_2 is over an unbounded time horizon, which leads to employing the IMDP method for synthesis (recall that the MDP implementation can only handle time-bounded properties, and its abstraction error monotonically increases with the time horizon of the formula). In order to encode the task we set the variable `safe` to correspond to X, the grid size to 0.12 and the relative tolerance to 0.06 along both dimensions (cf. lines 5-10 in Listing 6.3). We set the time horizon `K = -1` to represent an unbounded time horizon, let `p = synthesis_reach_avoid` to trigger the synthesis engine over the given specification and make `lb = imdp` to use IMDP method (cf. lines 12-19 in Listing 6.3). This task specification partitions the set X into the underlying IMDP via uniform

gridding. Alternatively, the user has the option to make use of the adaptive-sequential algorithm by defining a new variable `eps_max` which characterise the maximum allowable abstraction error and then specify the task using `taskSpec_t mySpec(lb,K,p,boundary,eps_max,grid,rtol);` Next, we define two files (PHI1.TXT and PHI2.TXT) containing the coordinates within the gridded domain (see Figure 6.2a) associated with the atomic propositions *purple* and *green*, respectively. This allows for automatic labelling of the state-space over which synthesis is to be performed. Running the main file, StocHy performs the abstraction and solves the synthesis problem against φ_2. On completion, it provides a summary of the generated results and quotes the total number of states in the generated abstraction, the computational time taken and the abstraction error. This is followed by a prompt to store the generated results within the data file called SOLUTION.TXT file found in the RESULTS folder. The data file contains the synthesised $\pi^\star$ policy, the lower bound for the probabilities of satisfying φ_2, and the local errors ϵ_s for any region s.

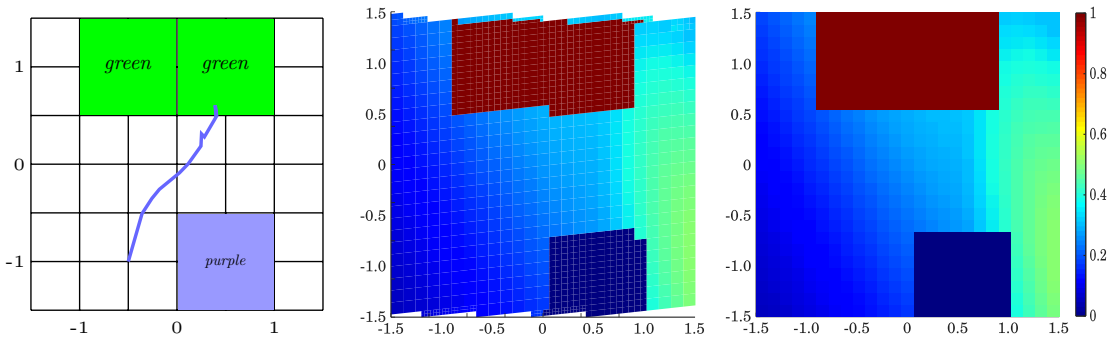


Figure 6.2: Case study 2: (a) Gridded domain together with a superimposed simulation of trajectory initialised at $(-0.5, -1)$ within q_0, under the synthesised switched strategy π^*, and the lower probabilities of satisfying φ_2 for mode q_0 (b) and for mode q_1 (c), as computed by StocHy.

Outcomes The case study generates an abstraction with a total of 2410 states, a maximum probability of 1, a maximum abstraction error of 0.20, and it requires a total time of 1639.3 $[s]$. Figure 6.2b and 6.2c depict the resulting lower-bound probabilities for satisfying φ_2 for mode q_0 and q_1, respectively. In this case, we

witness a slightly larger abstraction error via the IMDP method then in the previous case study. This is due to the non-diagonal covariance matrix within mode q_0, which introduces a rotation in X. When labelling the states associated with the regions *purple* and *green*, an additional error is introduced due to the over- and under-approximation of states associated with each of the two regions. We further show the simulation of a trajectory under $\pi^\star$ with a starting point of $(-0.5, -1)$ in q_0, within Figure 6.2a. The trajectory satisfies φ_2 and is automatically generated by StocHy.

6.4.3 Case Study 3: Scaling in continuous dimension of model

Verification via MDP abstractions has been shown to scale for analysis of SHS with up to four continuous variables [146]. In this subsection, we focus on pushing this limit and show how using the IMDP method we are able to scale to perform analysis of SHS models with a larger number of continuous variables. With this goal, focus is on the continuous dynamics by considering a stochastic process with $\mathcal{Q} = \{q_0\}$ (single mode) and dynamics evolving according to

$$X_n[k+1] = 0.8\mathbf{I}_d X_n[k] + 0.2\mathbf{I}_n W_n[k], \tag{6.2}$$

where n corresponds to the number of continuous variables. We are interested in checking the BLTL safety specification, evaluated over a probabilistic operator P

$$\varphi_3 := \mathrm{P}_{=?}\left(\mathbf{G}^{\leq 50} X_{safe}\right),$$

where $X_{safe} = [-1, 1]^n$, as the continuous dimension d of the model varies. In view of the focus on scalability for this Case Study 3, we disregard discussing the computed probabilities, which we instead covered in Section 6.4.1.

Implementation Similar to Case Study 2, we follow lines 3-12 in Listing 6.1 to define the model dynamics, while we use Listing 6.3 to specify the verification task using the IMDP method. For this example, we employ a uniform grid having a grid size of 1 and relative tolerance of 1 for each dimension (cf. lines 5-10 in

Listing 6.3). We set `K = 50` to represent the time horizon, `p = verify_safety` to perform verification over a safety property and `lb = imdp` to use the IMDP method (cf. lines 12-20 in Listing 6.3). In Table 6.2 we list the number of states required for each dimension, the total computational time, and the maximum error associated with each abstraction.

Outcomes From Table 6.2 we can deduce that by employing the IMDP method within StocHy, the generated abstract models have manageable state spaces, thanks to the tight error bounds that are obtained. Notice that since the number of cells per dimension is increased with the dimension n of the model, the associated abstraction error ε_{max} is decreased. The small error is also due to the underlying contractive dynamics of the process. This is a key fact leading to scalability over the continuous dimension n of the model: StocHy displays a significant improvement in scalability over the state of the art [147] and allows abstracting stochastic models with relevant dimensionality.

Dimensions [n]	2	3	4	5	6	7	8	9	10	11	12	13
$\lvert\mathcal{S}\rvert$ [states]	4	8	16	32	64	128	256	512	1024	2048	4096	8192
Time taken [s]	0.02	0.06	0.1	0.5	1.5	6.5	31.8	136.3	535.9	2065.1	9804.0	21122.2
Error (ε_{max})	0.08	0.02	6.0e-3	1.0e-3	4.0e-4	8.2e-5	5.7e-6	1.1e-6	1.6e-7	2.2e-8	1.1e-8	4.89e-9

Table 6.2: Case study 3: Verification results of the IMDP-based approach over φ_3, for varying dimension n of the stochastic process.

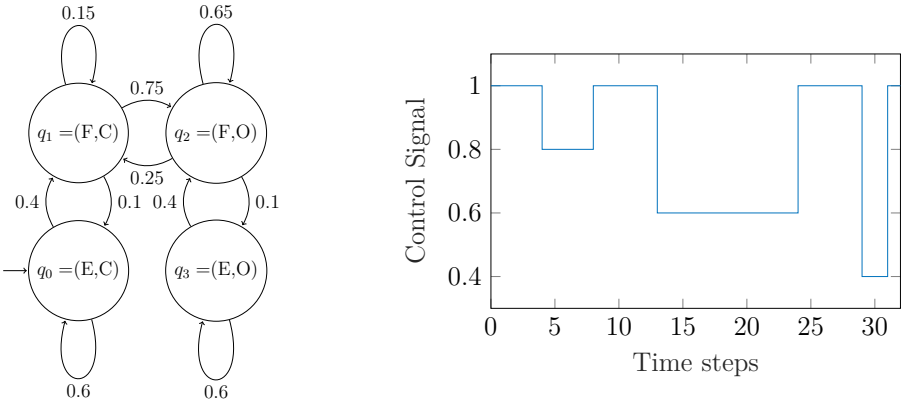
6.4.4 Case Study 4: Simulations

For this last case study, we refer to the CO_2 model described in Case Study 1 (Section 6.4.1). We extend the CO_2 model to capture (i) the effect of occupants leaving or entering the zone within a time step (ii) the opening or closing of the windows in the zone. m_{sa} is now a control input and is an exogenous signal. This can be described as a SHS comprising two-dimensional dynamics, over discrete modes in the set $\{q_0 = (E, C), q_1 = (F, C), q_2 = (F, O), q_3 = (E, O)\}$ describing possible configurations of the room (empty (E) or full (F), and with windows open

(O) or closed (C)). A DTMC representing the discrete modes and their dynamics is in Figure 6.3a. The continuous variables evolve according to (6.1), which now captures the effect of switching between discrete modes, as

$$\begin{aligned} CO_{2_{z_1}}[k+1] &= CO_{2_{z_1}}[k] + \frac{\Delta}{V}\left(-m_{sa}CO_{2_{z_1}}[k] + \varrho(t)(CO_{2,out} - CO_{2_{z_1}}[k])\right) \\ &\quad + \mathbf{1}_F CO_{2,occ}[k] + \sigma_1 w_1[k], \\ T_{z_1}[k+1] &= T_{z_1}[k] + \frac{\Delta}{C_z}(m_{sa}C_{pa}\left(T_{sp} - T_{z_1}[k]) + \frac{\varrho(t)}{R}(T_{out} - T_{z_1}[k])\right) \\ &\quad + \mathbf{1}_F Q_{occ}[k] + \sigma_2 w_2[k], \end{aligned} \tag{6.3}$$

where the additional terms are: ϱ is the natural drift air flow that changes depending whether the window is open (ϱ_o) or closed (ϱ_c) $[m^3/min]$; $CO_{2,occ}$ is the generated CO_2 level when the zone is occupied (it is multiplied by the indicator function $\mathbf{1}_F$) $[ppm/min]$; Q_{occ} is the generated heat due to occupants $[\,^\circ C/min]$, which couples the dynamics in (6.3) as $Q_{occ,k} = \mu CO_{2_{z_1}}[k] + \beta$.



(a) **(b)**

Figure 6.3: Case study 4: (a) DTMC for the discrete modes of the CO_2 model and (b) the input control signal.

Implementation The provided file CS4.MAT sets the values of the parameters in (6.3) and contains the transition probability matrix representing the relationships between discrete modes. We select a sampling time $\Delta = 15$ $[min]$ and simulate the evolution of this dynamical model over a fixed time horizon $K = 8$ $[hours]$ (i.e. 32 steps) with an initial CO_2 level $CO_{2_{z_1}} \sim \mathcal{N}(450, 25)$ $[ppm]$ and a temperature level

```
1  // Number of simulations
2  int monte = 5000;
3  // Initial continuous variables
4  arma::mat x_init = arma::zeros<arma::mat>(2,monte);
5  // Initialise random generators
6  std::random_device rand_dev;
7  std::mt19937 generator(rand_dev());
8  // Define distributions
9  // CO_2 levels
10 std::normal_distribution<double> d1{450,25};
11 // Temperature values
12 std::normal_distribution<double> d2{17,2};
13 // Populate initial values
14 for(size_t i = 0; i < monte; ++i)
15 {
16 x_init(0,i) = d1(generator);
17 x_init(1,i) = d2(generator);
18 }
19 // Initial discrete mode q_0 = (E,C)
20 arma::mat q_init = arma::zeros<arma::mat>(1,monte);
21 // Definition of control signal
22 // Read from .txt/.mat file or define here
23 arma::mat u =readInputSignal("../u.txt");
24 //Combining
25 exdata_t data(x_init,u,q_init);
```

Listing 6.4: Case study 4: definition of intial conditions for simulation

of $T_{z_1} \sim \mathcal{N}(17, 2)$ [$^{\circ}C$]. We define the initial conditions using Listing 6.4. Line 2 defines the number of Monte Carlo simulations using by the variable `monte` and sets this to 5000. We instantiate the initial values of the continuous variables using the term `x_init`, while we set the initial discrete mode using the variable `q_init`. This is done using lines 4-20 which defines independent normal distribution for each of the continuous variable from which we sample 5000 points for each of the continuous variables and defines the initial discrete mode to $q_0 = (E, C)$. We define the control signal m_{sa} in line 23, by parsing the `u.txt` which contains discrete values of m_{sa} for each time step (see Figure 6.3b). Once the model is defined, we follow Listing 6.1 to perform the simulation. The simulation engine also generates a PYTHON script, `simPlots.py`, which gives the option to visualise the simulation outcomes offline.

Outcomes The generated simulation plots are shown in Figure 6.4, which depicts: (i) a sample trace for each continuous variable (the evolution of $CO_{2_{z_1}}$ is shown in Figure 6.4a, T_{z_1} in Figure 6.4b) and for the discrete modes in Figure 6.4c; (ii) histograms portraying the range of values the continuous variables can be in during each time step and the associated count (Figure 6.4d for $CO_{2_{z_1}}$, Figure 6.4e for T_{z_1}); and a histogram showing the likelihood of being in a discrete mode within each time step (see Figure 6.4f). The total time taken to generate the plots is 48.6 $[s]$.

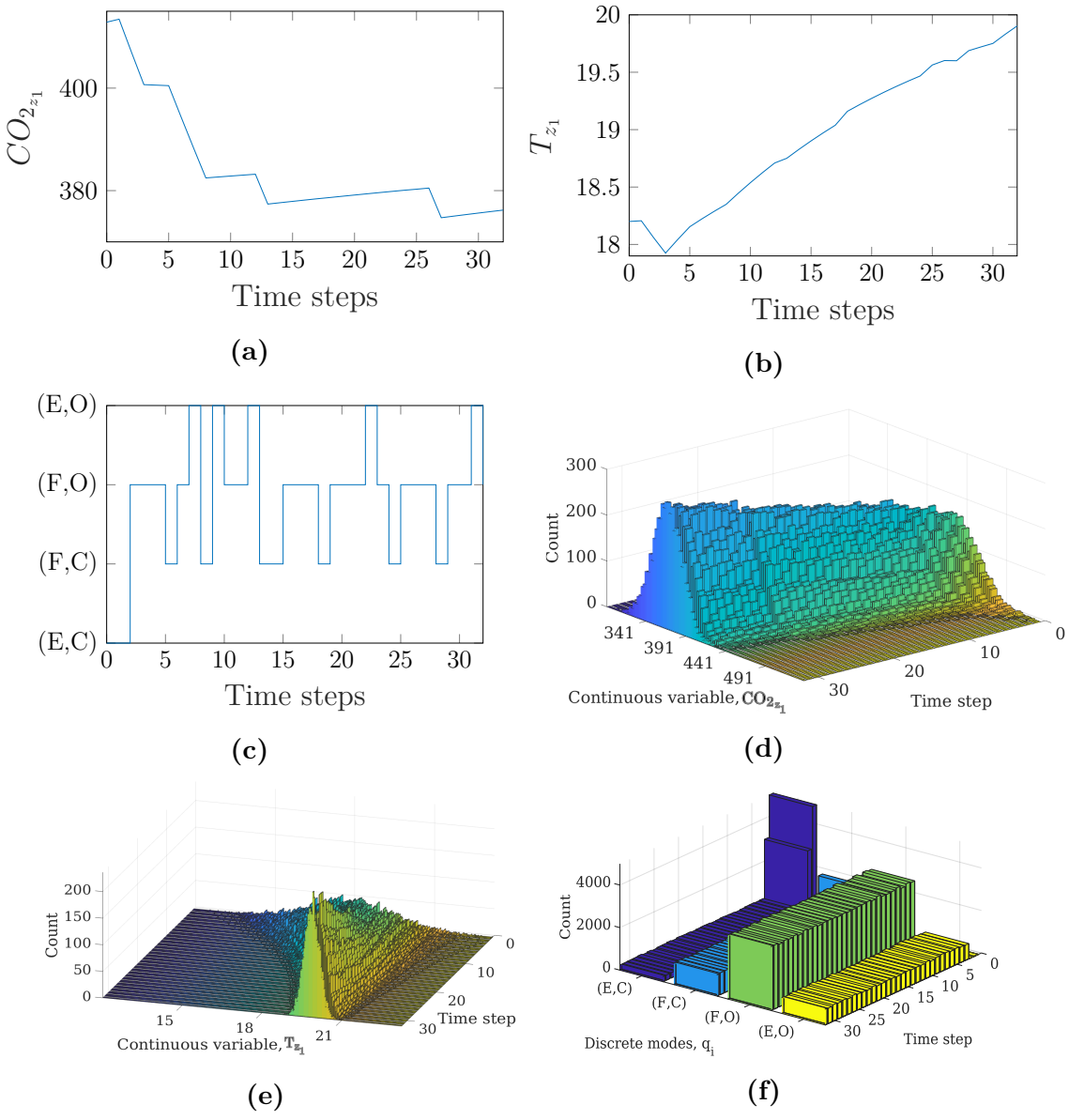


Figure 6.4: Case study 4: Simulation single traces for continuous variables (a) $CO_{2_{z_1}}$, (b) T_{z_1} and discrete modes (c) q. Histogram plots with respect to time step for (d) $CO_{2_{z_1}}$, (e) T_{z_1} and (f) q_i, $i \in \{0, \ldots, 3\}$.

6.5 Conclusion

We have presented StocHy, a new software tool for the quantitative analysis of stochastic hybrid systems. There is a plethora of enticing extensions that we are planning to explore. In the short term, we intend to: (i) interface with other model checking tools such as MODEST TOOLSET [79]; (ii) embed algorithms for policy refinement, so we can generate policies for models having numerous continuous input variables using formal abstractions via MDP [71]. In the longer term, we plan to extend StocHy such that it can (i) employ a graphical user-interface; (ii) allow analysis of continuous-time SHS; and (iii) make use of data structures such as multi-terminal binary decision diagrams [50], [59] to reduce the memory requirements during the construction of the abstract MDP or IMDP.

7 Conclusion and Future Research

Contents

This chapter summaries the thesis by discussing key contributions in the manuscript and delineates some directions for research that are presently pursued by the author.

7.1 Conclusion

The rapid growth and development of the building industry poses the problem of certification to guarantee their performance. The fundamental challenge in achieving such a goal is the diversity of modelling methods and frameworks available, which are typically decoupled from the certification requirements. Buildings are also exposed to a high level of uncertainty and are effected by external dynamics (such as the weather) that increase complexity during the modelling stage. In this work we devise a framework for constructing and analysing models for BAS according to a certification goal.

We classify the term certification into two separate classes: high-level (overall performance) and low-level certification (behaviour of subsystems). This allows us to reason at the appropriate level of complexity depending on the goal at hand. In Chapter 2 we formally define these two classes and identify key metrics that are used to quantify the performance of the system at each certification level. Following this chapter, the thesis is split into methods for reasoning over either of the certification levels.

For high-level certification we focus on the task of building maintenance. This needs to be carried out over the whole building system in order to ensure efficient and cost-effective operation. With this goal in mind, in Chapter 3 we present the *fault maintenance trees (*FMT*) framework* as a means of *performing preventative maintenance for a* BAS *setup.* FMTs model the different components and their degradation within the BAS together with the relationship between the behaviour of individual components with respect to fault propagation. This makes it ideal for reasoning about the selection of maintenance actions. In this thesis we apply the framework for BAS, give semantics to the qualitative FMT description in the form of CTMCs and devise a PMC *architecture* for computing KPI metrics such that optimal maintenance strategies are selected. We further compare our analysis against techniques based on SMC. While PMC is more computationally efficient, it requires *state-space reduction* techniques in order to be able to analyse the whole FMT structure.

Low-level certification requests the need for more detailed modelling structures. Consequently, in Chapter 4 we introduce the mathematical modelling framework of SHS that captures both uncertainty (taking the form of noise) and the relationship between discrete (e.g. software control algorithms) and continuous (physical structures) components. We provide a *library of models* for BAS, show how we can build different SHS models of varying complexities, and translate low-level certification goals into formal specifications using temporal logic. Chapter 5 builds on the SHS framework and presents two formal abstraction methods over which certification goals can be analysed. First, we delineate the classical approach of

abstracting SHS into discrete-time Markov processes (DTMC, or MDP) and then computing verification and synthesis goals. We also show how (ϵ, δ)-*simulation relations* can be *applied* for performing strategy synthesis via MDP for models with a large number of continuous variables. Next, we introduce a novel method for *scalable* and *efficient abstractions* via *Interval Markov decision processes* (IMDP). A comparison between the two abstraction techniques (MDP vs IMDP based) highlights the strengths of the novel IMDP method in terms of the abstraction error generated, scalability and the ability to reason over unbounded time horizons. Finally, we solve the BAS case studies presented in Chapter 4.

The algorithms and case studies presented in Chapter 4 and 5 are realised in a *state-of-the-art software tool called* StocHy. This tool is aimed at making SHS more attractive to general end-users and easier to use. StocHy allows for (i) simulation, (ii) verification and (iii) synthesis of SHS. It is written in C++ for portability and is both modular and easily extendable in structure.

7.2 Recommendations for future research

There are many possible areas in which the work in this thesis can be developed further.

Analysis of continuous-time stochastic hybrid systems It is certainly of interest to extend the analysis of models evolving over continuous time and space. This requires the extension of our SHS model to include the description of the evolution of the continuous variables via stochastic differential equations. Unfortunately, the computation of the conditional stochastic kernel comes with a multitude of complexities [105]. One of the options is to first discretise the continuous space and then perform any of the formal abstraction techniques delineated in this thesis. However, this comes at a caveat of requiring a huge number of states to represent the model for a sufficient abstraction error.

Formal modelling language for stochastic hybrid systems In the current implementation of StocHy, users need to provide a description of the relationship between discrete modes in the form of discrete transition probabilities, while the evolution of the continuous dynamics is described using state-space equations. It would be desirable to have a higher level (possibly, graphical) description of the SHS. This will allow for the inclusion of more versatile dynamics (e.g. transition probabilities described by a variety of probability distributions, non-linear evolution of continuous variables or probabilistic resets), together with providing an easier means of connecting with other tools. It was further noted in [1] that there is a lack of a common formal language for describing SHS: benchmarks portraying different SHS models had different definitions and hence making it hard to compare techniques and methodologies. A reputable attempt to provide such a language is the HMODEST language [74] and the language exchange protocol JANI [34]. However, the HMODEST language is limited to continuous-time SHS where the continuous variables are described using ordinary differential equations (i.e. neglect stochasticity in the evolution of the continuous variables).

Encoding of the transition probabilities within the abstractions The key element in the large memory requirements needed for formal abstractions is the storage of the conditional transition probabilities over the gridded state space. In StocHy, we have employed the use of sparse matrices to reduce this memory requirement. However in the worst case, this corresponds to the full size of the conditional transition probabilities for all the possible combinations making up the states space. A possible means of reduction is to employ a more efficient data structure that makes use of encodings such as multi-terminal binary decision diagram [50], [59]. This however, comes with its own qualms: most probabilities in the transition probability matrix are unique values. To mitigate this one can truncate the underlying probability distribution to a certain significant decimal value and then quantify the additional abstraction error being introduced.

Predictive maintenance via fault maintenance trees In Chapter 3, we present a preventative maintenance method for BAS using FMT. We can rephrase the problem into an optimisation scheme that computes the best maintenance actions via a performance-based cost function. One approach is to transform the degradation models within the FMT into dynamical models which depend on the maintenance action [3], [43], [55]. Consequently, we can describe the FMT as SHS and employ the strategy synthesis algorithms found in StocHy to compute the optimal maintenance strategies. Note, that due to the large number of degrading components present in the FMT, special care needs to be taken to mitigate state-explosion as a result of performing parallel composition of all the individual abstract representations of the components. However, this also alleviates the need to approximate time steps via the Erlang distribution.

Connect the library of models for BAS with the Brick schema The BRICK [18] meta-data schema defines a concrete ontology network describing the structure of the building based on the connected sensors and the relationship between them. This allows for generation of a high-level description on the structure of different types of buildings. By connecting the library of BAS models and extending it to contain models of other sub-components which are not currently included (e.g. the chiller), we can construct realistic and diverse models over which certification and analysis can be performed. Further bridging the gap between the actual building structure, modelling and analysis.

Appendices

A
Case studies

Contents

A.1 Two zone heating setup with stochastic dynamics

We present the corresponding system matrices of the model $\mathbf{M}_s$ described in Section 4.3.1. $\mathbf{M}_s$ is given by (4.10) and is characterised by the following system matrices

$$A_s = \begin{bmatrix} 0.6250 & 0.00024 & 0.00065 & 0 \\ 1.3737e-05 & 0.7696 & 0 & 0.00029 \\ 0.0134 & 0 & 0.9513 & 0 \\ 0 & 0.0134 & 0 & 0.9513 \end{bmatrix}, \; B_s = \begin{bmatrix} 0.3739 \\ 0.2300 \\ 0 \\ 0 \end{bmatrix},$$

$$Q_s = \begin{bmatrix} 0.0037 \\ -1.6528e-04 \\ 2.6511 \\ 2.6511 \end{bmatrix}, \; \Sigma_s = \begin{bmatrix} 0.3096 & 0 & 0 & 0 \\ 0 & 0.3096 & 0 & 0 \\ 0 & 0 & 1.5488 & 0 \\ 0 & 0 & 0 & 1.5488 \end{bmatrix}.$$

$\mathbf{M}_{s'}$ is shaped by the following system matrices

$$A_{s'} = \begin{bmatrix} 0.6250 & 0.00024 \\ 1.3737e-05 & 0.7696 \end{bmatrix},\ B_{s'} = \begin{bmatrix} 0.3739 \\ 0.2300 \end{bmatrix},$$

$$Q_{s'} = \begin{bmatrix} 0.0037 \\ -1.6528e-04 \end{bmatrix},\ \Sigma_{s'} = \begin{bmatrix} 0.3096 & 0 \\ 0 & 0.3096 \end{bmatrix}.$$

A.2 Two zone heating setup with large number of continuous variables

The model $\mathbf{M}_{s,7}$ is described by the following system matrices

$$A_{s,7} = \begin{bmatrix} 0.9678 & 0 & 0.0035 & 0 & 0.0036 & 0 & 0.0036 \\ 0 & 0.9682 & 0 & 0.0034 & 0 & 0.0034 & 0.0034 \\ 0.0106 & 0 & 0.9494 & 0 & 0 & 0 & 0 \\ 0 & 0.0097 & 0 & 0.9523 & 0 & 0 & 0 \\ 0.01057 & 0 & 0 & 0 & 0.9494 & 0 & 0 \\ 0 & 0.0097 & 0 & 0 & 0 & 0.9523 & 0 \\ 0.0106 & 0.0097 & 0 & 0 & 0 & 0 & 0.9794 \end{bmatrix},$$

$$B_{s,7} = \begin{bmatrix} 0.0195 \\ 0.0199 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},\ Q_{s,7} = \begin{bmatrix} 0.0164 \\ -0.0018 \\ 0.0387 \\ 0.0189 \\ 0.0110 \\ 0.0108 \\ 0.0109 \end{bmatrix},$$

$$G_{s,7} = \begin{bmatrix} 0 & 0 & 1.9380e-05 & 0 & 0.0019 & 0 \\ 0 & 0 & 0 & 5.7769e-07 & 0 & 0.0015 \\ 0.0458 & 0 & 0 & 0 & 0 & 0 \\ 0.0425 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0397 & 0 & 0 & 0 & 0 \\ 0 & 0.0377 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In Section 5.4.2, we perform model-order reduction on $\mathbf{M}_{s,7}$, to obtain four simpler models $\mathbf{M}_{s,i}, i \in \{4, \ldots, 1\}$. The model $\mathbf{M}_{s,4}$ is described using

$$A_{s,4} = \begin{bmatrix} 0.9678 & 0.0036 & 0.0036 & 0.0036 \\ 0.0106 & 0.9494 & 0 & 0 \\ 0.0106 & 0 & 0.9494 & 0 \\ 0.0106 & 0 & 0 & 0.9794 \end{bmatrix},\ B_{s,4} = \begin{bmatrix} 0.0195 \\ 0 \\ 0 \\ 0 \end{bmatrix},\ Q_{s,4} = \begin{bmatrix} 0.0164 \\ 0.0386 \\ 0.0110 \\ 0.0110 \end{bmatrix},$$

$$G_{s,4} = \begin{bmatrix} 0 & 0 & 1.9381e-05 & 0.0019 & 0 \\ 0.0458 & 0 & 0 & 0 & 0 \\ 0 & 0.0397 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0097 \end{bmatrix}.$$

$\mathbf{M}_{s,3}$ is given by

$$A_{s,3} = \begin{bmatrix} 0.9714 & 0.0036 & 0.0036 \\ 0.0106 & 0.9494 & 0 \\ 0.0106 & 0 & 0.9494 \end{bmatrix},\ B_{s,3} = \begin{bmatrix} 0.0195 \\ 0 \\ 0 \end{bmatrix},\ Q_{s,3} = \begin{bmatrix} 0.0164 \\ 0.0387 \\ 0.0110 \end{bmatrix},$$

$$G_{s,3} = \begin{bmatrix} 0 & 0 & 1.9381e-05 & 0.0019 \\ 0.0459 & 0 & 0 & 0 \\ 0 & 0.0397 & 0 & 0 \end{bmatrix}.$$

The model $\mathbf{M}_{s,2}$ is characterised by

$$A_{s,2} = \begin{bmatrix} 0.9750 & 0.0036 \\ 0.0106 & 0.9388 \end{bmatrix},\ B_{s,2} = \begin{bmatrix} 0.0195 \\ 0 \end{bmatrix},\ Q_{s,2} = \begin{bmatrix} 0.0164 \\ 0.0387 \end{bmatrix},$$

$$G_{s,2} = \begin{bmatrix} 0 & 1.9381e-05 & 0.0019 \\ 0.0459 & 0 & 0 \end{bmatrix}.$$

Last, $\mathbf{M}_{s,1}$ is delineated using

$$A_{s,1} = \begin{bmatrix} 0.9177 \end{bmatrix},\ B_{s,1} = \begin{bmatrix} 0.0575 \end{bmatrix},\ Q_{s,1} = \begin{bmatrix} 0.0872 \end{bmatrix},\ G_{s,1} = \begin{bmatrix} 0 & 1.9381e-05 & 0.0057 \end{bmatrix}.$$

A.3 Air quality model capturing CO_2 and temperature dynamics in zone

We provide the state-space models describing the continuous evolution using (4.16) for each discrete mode. When the current discrete state corresponds to $q_0 = (\neg F_{en}, \neg W_{en})$, the continuous dynamics evolve according to

$$A_{s,q_0} = \begin{bmatrix} 0.9995 & 0 \\ 0.0030 & 0.9998 \end{bmatrix},\ Q_{s,q_0} = \begin{bmatrix} 1.4748 \\ 0.0007 \end{bmatrix},$$

$$G_{s,q_0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \Sigma_{s,q_0} = \begin{bmatrix} 9.7227 & 0 \\ 0 & 2.3241 \end{bmatrix}.$$

For $q_1 = (F_{en}, W_{en})$ we have

$$A_{s,q_1} = \begin{bmatrix} 0.9830 & 0 \\ 0.4693 & 0.9935 \end{bmatrix},\ Q_{s,q_1} = \begin{bmatrix} 2.0050 \\ 0.1378 \end{bmatrix},$$

$$G_{s,q_1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \Sigma_{s,q_1} = \begin{bmatrix} 14.5641 & 0 \\ 0 & 4.6482 \end{bmatrix}.$$

Similarly for $q_2 = (\neg F_{en}, W_{en})$

$$A_{s,q_2} = \begin{bmatrix} 0.9913 & 0 \\ 6.4603e-06 & 0.9998 \end{bmatrix},\ Q_{s,q_2} = \begin{bmatrix} 2.0050 \\ 0.0007 \end{bmatrix},$$

$$G_{s,q_2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \Sigma_{s,q_2} = \begin{bmatrix} 14.5641 & 0 \\ 0 & 2.3241 \end{bmatrix}.$$

Last, we have that when the current state is in $q_3 = (F_{en}, \neg W_{en})$

$$A_{s,q_3} = \begin{bmatrix} 0.9911 & 0 \\ 6.4603e-06 & 0.9935 \end{bmatrix},\ Q_{s,q_3} = \begin{bmatrix} 1.4748 \\ 0.1378 \end{bmatrix},$$

$$G_{s,q_3} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \Sigma_{s,q_3} = \begin{bmatrix} 9.7227 & 0 \\ 0 & 4.6482 \end{bmatrix}.$$

These model are constructed using the library of BAS models, where the following variables are assumed as external disturbance sampled from i.i.d. Gaussian distributions as follows

$$CO_{2,occ} \sim \mathcal{N}(975, 0.5) \qquad CO_{2,out} \sim \mathcal{N}(415, 0.5),$$
$$T_{w_{jn}} \sim \mathcal{N}(18, 2), \qquad T_{sa_1} \sim \mathcal{N}(22, 1).$$

We additionally set $\sigma_{2_{z_i}} = 0.83$ and $\sigma_{z_i} = 0.2$ for $i = \{1, 2\}$.

B

Proofs associated with formal abstractions into IMDP

Contents

B.1 Proof of Proposition 1

Proof. For a fixed $a \in \mathcal{A}$, recall that

$$T(s \mid x, a) = \int_s \mathcal{N}(t \mid F(a,x), \Sigma_x(a))\, dt,$$

where $F(a,x) = F(a)x + Q(a)$ and $\Sigma_x(a) = G(a)\Sigma_w G(a)^T$. By applying a whitening through the transformation matrix $\mathcal{T}_a = \Lambda_a^{-\frac{1}{2}} V_a^T$, we obtain that $\mathcal{T}_a Cov_x(a) \mathcal{T}_a^T = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Thus, by working in the transformed space induced by $\mathcal{T}_a$, we obtain

$$T_x(s \mid x, a) = \int_{Post(s, \mathcal{T}_a)} \mathcal{N}\Big(t \mid \mathcal{T}_a\, F(a,x), \mathbf{I}\Big)\, dt.$$

Under the assumption that $Post(s, \mathcal{T}_a)$ is a hyper-rectangle, the above multidimensional integral can be separated and expressed as a product of n integrals of uni-dimensional normal distributions

$$\begin{aligned} T_x(s \mid x, a) &= \int_{Post(s,\mathcal{T}_a)} \mathcal{N}\Big(t \mid \mathcal{T}_a\, F(a,x), \mathbf{I}\Big)\, dt \\ &= \int_{v_l^{(1)}}^{v_u^{(1)}} \cdots \int_{v_l^{(n)}}^{v_u^{(n)}} \mathcal{N}\Big(t_1 \mid y^{(1)}, 1\Big) \cdots \mathcal{N}\Big(t_n \mid \\ &\qquad\qquad y^{(n)}, 1\Big)\, dt_1 \cdots dt_n \\ &= \prod_{i=1}^{n} \int_{v_l^{(i)}}^{v_u^{(i)}} \mathcal{N}\Big(t_i \mid y^{(i)}, 1\Big)\, dt_i \\ &= \prod_{i=1}^{n} \frac{1}{2}\Big(\mathrm{erf}(\frac{y^{(i)} - v_l^{(i)}}{\sqrt{2}}) - \mathrm{erf}(\frac{y^{(i)} - v_u^{(i)}}{\sqrt{2}})\Big), \end{aligned}$$

where $y = \mathcal{T}_a\, F(a, x)$. □

B.2 Proof of Theorem 5.2.1

Proof. We first consider the maximum case and then discuss the minimum case. The KKT conditions guarantee that if $y \in Post(s'_i, \mathcal{T}_a)$ is a local maximum for f, then there must exist a vector of constants $\eta = (\eta_1, \ldots, \eta_k)$ such that $\nabla f(y) = H^T \eta$, $\eta_i \geq 0$ for all $i \in \{1, ..., k\}$, and $\eta_i(\sum_{j=1}^{m} H^{(i,j)} y^{(j)} - b_i) = 0$, where $H^{(i,j)}$ is the component in the i-th row and j-th column of matrix H. Note that we have a constant η_i, $i \in \{1, \ldots, k\}$, for each of the half-spaces defining $Post(s'_i, \mathcal{T}_a)$. Thus, there are three possible cases:

- **Case 1: x^* is not in the boundary of $Post(s'_i, \mathcal{T}_a)$.** In this case the KKT conditions imply that y is a maximum only if $\nabla f(y) = 0$. For a normal distribution with identity covariance, this point is exactly $y = \Big(\frac{v_u^{(i)} + v_l^{(1)}}{2}, ..., \frac{v_u^{(n)} + v_l^{(n)}}{2}\Big)$. If $y \in Post(s'_i, \mathcal{T}_a)$, then this is the global maximum, because it is the global maximum of the unconstrained problem.

- **Case 2: x^* is a vertex of $Post(s'_i, \mathcal{T}_a)$.** We call a vertex an intersection of n half-spaces. As a consequence, we have that the KKT conditions are satisfied in y, vertex of $Post(s'_i, \mathcal{T}_a)$, if and only if $\nabla f(y) = \bar{H}^T \eta$, where $\bar{H}$ is

the sub matrix that contains only the n rows of H representing the half-spaces interesting at y, and vector η contains only the n corresponding constants. Thus, we have a system of n equations and n variables that have a solution for $\eta_i \in \mathbb{R}$. However, since the set of vertices is finite, it is generally faster to just include all the vertices as possible candidate solutions instead of solving the system of equations.

- **Case 3: y is in the boundary of $Post(s'_i, \mathcal{T}_a)$, but is not a vertex.** In this case only $r < n$ of the half-spaces in H intersect at y. Thus, if y is a maximum then $\nabla f(y) = \bar{H}^T \eta$, where $\bar{H}$ is the sub matrix of H containing the $r < n$ half-spaces intersecting at y, and η contains only the r corresponding constants. Note that this is a system with more equations than variables. Therefore, only when some of constraints become linearly dependent, there may be a solution for $y \in Post(s'_i, \mathcal{T}_a)$, if at all.

The minimum case is identical except that condition $\nabla f(y) = H^T \eta$ is replaced with $\nabla f(y) = -H^T \eta$. □

B.3 Proof of Proposition 2

Proof. By Definition we have

$$f(y) = \prod_{i=1}^{n} \bar{f}(y^{(i)} \mid v_l^{(i)}, v_u^{(i)}),$$

where

$$\bar{f}(y^{(i)} \mid v_l^{(i)}, v_u^{(i)}) = \frac{1}{2}\Big(\text{erf}(\frac{y^{(i)} - v_l^{(i)}}{\sqrt{2}}) - \text{erf}(\frac{y^{(i)} - v_u^{(i)}}{\sqrt{2}})\Big)$$

with $v_u^{(i)} > v_l^{(i)}$. Now, since a product of log-concave functions is a log-concave function itself, to show that $f(y)$ is log-concave, it is enough to show that $\bar{f}(y^{(i)} \mid v_l^{(i)}, v_u^{(i)})$ is log-concave for $i \in \{1, ..., n\}$. In order to do that we first need to observe that

$$\bar{f}(y^{(i)} \mid v_l^{(i)}, v_u^{(i)}) = \int_{y^{(i)} - v_u^{(i)}}^{y^{(i)} - v_l^{(i)}} \mathcal{N}(t \mid 0, 1) dt.$$

That is, $\bar{f}$ induces a standard Gaussian probability measure $\bar{P}$. We denote with $\bar{P}([y^{(i)}-v_u^{(i)}, y^{(i)}-v_l^{(i)}])$ the resulting probability for convex Borel set $[y^{(i)}-v_u^{(i)}, y^{(i)}-v_l^{(i)}]$. By rearranging terms, for $\lambda \in [0,1], y_1, y_2 \in \mathbb{R}$, we finally obtain

$$\begin{aligned}
&\bar{f}(\lambda y_1 + (1-\lambda)y_2 \mid v_l^{(i)}, v_u^{(i)}) = \\
&\bar{P}(\lambda[y_1 - v_u^{(i)}, y_1 - v_l^{(i)}] + (1-\lambda)[y_2 - v_u^{(i)}, y_2 - v_l^{(i)}]) \geq \\
&\bar{P}([y_1 - v_u^{(i)}, y_1 - v_l^{(i)}])^{\lambda} \bar{P}[y_2 - v_u^{(i)}, y_2 - v_l^{(i)}])^{1-\lambda} = \\
&\bar{f}(y_1 \mid v_l^{(i)}, v_u^{(i)})^{\lambda} \bar{f}(y_2 \mid v_l^{(i)}, v_u^{(i)}))^{(1-\lambda)},
\end{aligned}$$

where the above inequality is due to Theorem 2 in [131]. □

B.4 Proof of Proposition 3

Proof. For the upper bound, we have that for $s_i \in \mathcal{S}_{\text{safe}}$ and $a \in \mathcal{A}$,

$$\begin{aligned}
\max_{x \in s_i} T_x(X \mid x, a) &\leq \max_{x \in s_i} \int_X \mathcal{N}(z \mid F(a,x), \Sigma_x(a))\, dz \\
&= \max_{y \in Post(s_i', \mathcal{T}_a)} \int_{Post(X, \mathcal{T}_a)} \mathcal{N}(z \mid y, \mathbf{I})\, dz \\
&\leq \max_{y \in Post(s_i', \mathcal{T}_a)} \sum_{s \in \bar{\mathcal{S}}^q} \int_{Post(s, \mathcal{T}_a)} \mathcal{N}(z \mid y, \mathbf{I})\, dz \\
&= \max_{y \in Post(s_i', \mathcal{T}_a)} \sum_{s \in \bar{\mathcal{S}}^q} f(y, s).
\end{aligned}$$

For the lower bound, similarly to the upper bound, we have that

$$\min_{x \in s_i} T_x(X \mid x, a) \geq \min_{y \in Post(s_i', \mathcal{T}_a)} \sum_{s \in \mathcal{S}^q} f(y, s).$$

□

B.5 Proof of Theorem 5.2.2

For each φ, let $\mathcal{A}_{\bar{\varphi}} = (\mathcal{Z}, 2^{\bar{\Theta}}, \tau, z_0, \mathcal{Z}_{\text{ac}})$ be the DFA correspondent to φ with initial state z_0. Then, $P(\varphi \mid x, X, \pi_{\mathcal{H}}^*)$ can be computed on the product stochastic hybrid system $\mathcal{H}_{\varphi} = \mathcal{H} \times \mathcal{A}_{\bar{\varphi}} = (A \times \mathcal{Z}, F_{\varphi}, G_{\varphi}, \Theta, \mathcal{L}_{\varphi})$, where $\mathcal{L}_{\varphi}(x, (a,z)) = \mathcal{L}((a,x))$, $F_{\varphi}(a,z) = F(a,x)$ and $G_{\varphi}(a,z) = G(a)$. We define the set of accepting

states of $\mathcal{H}_\varphi$ as $X_{ac} = X \times \mathcal{A} \times \mathcal{Z}_{\text{ac}}$. It is possible to show that $P(\varphi \mid x, X, \pi^*_{\mathcal{H}})$ can be computed as the solution of the following Bellman equation

$$V(z_0, x, X, \pi^*_{\mathcal{H}}) = \begin{cases} 1 & \text{if } (x, \pi^*_{\mathcal{H}}(x), z_0) \in X_{ac} \\ 0 & \text{if } x \notin X \\ \int_X f(x'|x, \pi^*_{\mathcal{H}}(x_0)) V(\tau(z_0, L(x, \pi^*_{\mathcal{H}}(x)), x', X, \pi^*_{\mathcal{H}}) dx \end{cases} \tag{B.1}$$

where $f(x'|x,\ \pi^*_{\mathcal{H}}(x_0))$ the density function of transition kernel T and, with an abuse of notation, we call $\pi^*_{\mathcal{H}}(x_0)$ the action resulting from the application of the (stationary) strategy $\pi^*_{\mathcal{H}}$ in x_0. For $s \in \mathcal{S}$ call

$$\breve{V}^{\pi^*_{\mathcal{H}}}(z, s, X, \pi^*_{\mathcal{H}}) = \min_{x \in s} V(z, x, X, \pi^*_{\mathcal{H}}).$$

Then, it follows that

$$\breve{V}^{\pi^*_{\mathcal{H}}}(z_0, s, X, \pi^*_{\mathcal{H}}) = \begin{cases} 1 & \text{if there exists } x \in s \text{ s.t. } (x, \pi^*_{\mathcal{H}}(x), z_0) \in X_{ac} \\ 0 & \text{if } x \notin X \\ \min_{x \in s} \int_X f(x'|x, \pi^*_{\mathcal{H}}(x)) V(\tau(z_0, L(x, \pi^*_{\mathcal{H}}(x)), x', X, \pi^*_{\mathcal{H}}) dx' \end{cases}$$

Then, because for each $x_1, x_2 \in s$ it holds that $\pi^*_{\mathcal{H}}(x_1) = \pi^*_{\mathcal{H}}(x_2)$ and $\mathcal{S}_\varphi$ is a discretization of X that respects the propositional regions, we obtain

$$\breve{V}^{\pi^*_{\mathcal{H}}}(z_0, s, X, \pi^*_{\mathcal{H}}) \leq \begin{cases} 1 & \text{if there exists } x \in s \text{ s.t. } (x, \pi^*_{\mathcal{H}}(x), z_0) \in X_{ac} \\ 0 & \text{if } x \notin X \\ \min_{x \in s} \sum_{s \in \mathcal{S}_\varphi} T(s|x, \pi^*_{\mathcal{H}}(x)) \breve{V}^{\pi^*_{\mathcal{H}}}(\tau(z_0, L(x, \pi^*_{\mathcal{H}}(x)), x', X, \pi^*_{\mathcal{H}}) \end{cases}$$

The latter expression is exactly (5.33) for a fixed strategy $\pi^*_{\mathcal{H}}$. Similar approach can be used to prove that the solution of (B.1) is upper bounded by (5.34).

B.6 Proof of Lemma 1

$\mathcal{S}_\varphi$ is a discretization of X that does not respect the propositional regions R, and the labelling function L of $\mathcal{I}$ introduces an under approximation of those regions. Similar to the proof of Theorem 5.2.2, a product SHS $\mathcal{H}_\varphi$ can be constructed. By replacing

the discretization $\mathcal{S}_\varphi$ in the Bellman equation and noting that L under-approximates R, it holds that $\check{V}^{\pi^*_{\mathcal{H}}}(z_0, s, X, \pi^*_{\mathcal{H}})$ is an under-approximation of $P(\varphi \mid d_0, X, \pi^*_{\mathcal{H}})$.

For the upper bound, note that the labelling function $\mathcal{L}'$ over-approximates the labels of each region. With the same derivation as above but using L' instead of $\mathcal{L}$, it follows that

$$\hat{V}^{\pi^*_{\mathcal{H}}}(z, s, X, \pi^*_{\mathcal{H}}) \geq P(\varphi \mid d_0, X, \pi^*_{\mathcal{H}}),$$

where

$$\hat{V}^{\pi^*_{\mathcal{H}}}(z, s, X, \pi^*_{\mathcal{H}}) = \max_{x \in q} V(z, x, X, \pi^*_{\mathcal{H}}),$$

and $V(z, x, X, \pi^*_{\mathcal{H}})$ is defined in (B.1).

Bibliography

[1] A. Abate, H. Blom, N. Cauchi, K. Degiorgio, M. Franzle, E. M. Hahn, S. Haesaert, H. Ma, M. Oishi, C. Pilch, A. Remke, M. Salamati, S. Soudjani, B. van Huijgevoort, and A. P. Vinod, "ARCH-COMP19 Category Report: Stochastic Modelling", *EPiC Series in Computing*, vol. 61, G. Frehse, Ed., pp. 62–102, 2019.

[2] A. Abate, H. Blom, N. Cauchi, S. Haesaert, A. Hartmanns, K. Lesser, M. Oishi, V. Sivaramakrishnan, S. Soudjani, C.-I. Vasile, *et al.*, "ARCH-COMP18 Category Report: Stochastic Modelling", in *ARCH18. Fifth International Workshop on Applied Verification of Continuous and Hybrid Systems, ARCH@ ADHS 2018, Oxford, UK, July 13, 2018*, 2018, pp. 71–103.

[3] A. Abate, C. E. Budde, N. Cauchi, A. van Harmelen, K. A. Hoque, and M. Stoelinga, "Modelling Smart Buildings Using Fault Maintenance Trees", in *European Workshop on Performance Engineering (EPEW)*, Springer, 2018, pp. 110–125.

[4] A. Abate, C. E. Budde, N. Cauchi, K. A. Hoque, and M. Stoelinga, "Assessment of Maintenance Policies for Smart Buildings: Application of Formal Methods to Fault Maintenance Trees", in *Proceedings of the European Conference of the PHM Society*, PHM society, vol. 4, 2018.

[5] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandin, "Approximate Model Checking of Stochastic Hybrid Systems", *European Journal of Control*, vol. 16, no. 6, pp. 624–641, 2010, ISSN: 0947-3580.

[6] A. Abate, F. Redig, and I. Tkachev, "On the Effect of Perturbation of Conditional Probabilities in Total Variation", *Statistics and Probability Letters*, vol. 88, pp. 1–8, 2014.

[7] Z. Afroz, G. Shafiullah, T. Urmee, and G. Higgins, "Modeling Techniques used in Building HVAC Control Systems: A Review", *Renewable and Sustainable Energy Reviews*, vol. 83, pp. 64–84, 2018.

[8] W. Akram and M. A. Niazi, "A Formal Specification Framework for Smart Grid Components", *Complex Adaptive Systems Modeling*, vol. 6, no. 1, p. 5, 2018.

[9] S. Alaswad and Y. Xiang, "A Review on Condition-Based Maintenance Optimization Models for Stochastically Deteriorating System", *Reliability Engineering and System Safety*, vol. 157, pp. 54–63, 2017, ISSN: 0951-8320.

[10] ASHRAE, "HVAC Systems and Equipment", *American Society of Heating, Refrigerating, and Air Conditioning Engineers, Atlanta, GA*, 1996.

[11] ——, "Standard 55-2010 - thermal Environmental Conditions for Human Occupancy", *American Society of Heating, Refrigerating and Air-Conditioning Engineering, Atlanta, GA*, 2004.

[12] F. Auffenberg, S. Snow, S. Stein, and A. Rogers, "A Comfort-Based Approach to Smart Heating and Air Conditioning", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 3, 28:1–28:20, 2017, ISSN: 2157-6904.

[13] "Maintenance Optimization", in *Stochastic Models in Reliability*, T. Aven and U. Jensen, Eds., vol. 41, Springer New York, 1999, pp. 169–211, ISBN: 978-0-387-22593-7.

[14] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Verifying Continuous Time Markov Chains", in *International Conference on Computer Aided Verification (CAV)*, Springer, vol. 1102, 1996, pp. 269–276.

[15] P. Bacher and H. Madsen, "Identifying Suitable Models for the Heat Dynamics of Buildings", *Energy and Buildings*, vol. 43, no. 7, pp. 1511–1522, 2011, ISSN: 0378-7788.

[16] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski, "Controller Synthesis for Probabilistic Systems (Extended Abstract)", in *Exploring New Frontiers of Theoretical Informatics*, J.-J. Levy, E. W. Mayr, and J. C. Mitchell, Eds., Boston, MA: Springer US, 2004, pp. 493–506, ISBN: 978-1-4020-8141-5.

[17] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[18] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, *et al.*, "Brick : Metadata Schema for Portable Smart Building Applications", *Applied Energy*, vol. 226, pp. 1273–1292, 2018, ISSN: 0306-2619.

[19] O. Balci, "Verification, Validation, and Certification of Modeling and Simulation Applications", in *Proceedings of the 35th Conference on Winter Simulation: Driving Innovation*, ser. WSC '03, New Orleans, Louisiana: Winter Simulation Conference, 2003, pp. 150–158, ISBN: 0-7803-8132-7.

[20] S. Baldi, F. Zhang, T. L. Quang, P. Endel, and O. Holub, "Passive versus Active Learning in Operation and Adaptive Maintenance of Heating, Ventilation, and Air Conditioning", *Applied Energy*, vol. 252, p. 113 478, 2019, ISSN: 0306-2619.

[21] R. E. Barlow and F. Proschan, "Mathematical Theory of Reliability", *Science*, vol. 148, no. 3674, pp. 1208–1209, 1965, ISSN: 0036-8075.

[22] C. Bauer, A. Frink, and R. Kreckel, "Introduction to the GiNaC Framework for Symbolic Computation within the C++ Programming Language", *Journal of Symbolic Computation*, vol. 33, no. 1, pp. 1–12, 2002.

[23] U. Baur, P. Benner, and L. Feng, "Model Order Reduction for Linear and Nonlinear Systems: A System-Theoretic Perspective", *Archives of Computational Methods in Engineering*, vol. 21, no. 4, pp. 331–358, 2014, ISSN: 1886-1784.

[24] G. Behrmann, K. G. Larsen, and J. I. Rasmussen, "Priced Timed Automata: Algorithms and Applications", in *International Symposium on Formal Methods for Components and Objects*, Springer, 2004, pp. 162–182.

[25] H. Belkhouane, J. Hensen, and S. Attia, "Thermal Comfort Models for Net Zero Energy Buildings in Hot Climates", in *Second International Conference on Energy and Indoor Environment for Hot Climates*, Doha, 2017.

[26] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic Planning and Control of Robot Motion [Grand Challenges of Robotics]", *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

[27] D. M. Berry and E. Kamsties, "Ambiguity in Requirements Specification", in *Perspectives on software requirements*, Springer, 2004, pp. 7–44.

[28] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 2014.

[29] H. Blom and J. Lygeros (Eds.), *Stochastic Hybrid Systems: Theory and Safety Critical Applications*, ser. Lecture Notes in Control and Information Sciences 337. Springer Verlag, Berlin Heidelberg, 2006.

[30] C. Boettiger, "An Introduction to Docker for Reproducible Research", *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.

[31] M. Bouissou, H. Elmqvist, M. Otter, and A. Benveniste, "Efficient Monte Carlo Simulation of Stochastic Hybrid Systems", in *Proceedings of the tenth International Modelica Conference; March 10-12; 2014; Lund; Sweden*, Linköping University Electronic Press, 2014, pp. 715–725.

[32] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[33] T. Brázdil, K. Chatterjee, M. Chmelik, V. Forejt, J. Kretínský, M. Z. Kwiatkowska, D. Parker, and M. Ujma, "Verification of Markov Decision Processes Using Learning Algorithms", in *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings*, 2014, pp. 98–114.

[34] C. E. Budde, C. Dehnert, E. M. Hahn, A. Hartmanns, S. Junges, and A. Turrini, "JANI: Quantitative Model and Tool Interaction", in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Springer, 2017, pp. 151–168.

[35] A. Camacho, O. Chen, S. Sanner, and S. A. McIlraith, "Decision-Making with Non-Markovian Rewards: From LTL to Automata-Based Reward Shaping", in *Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2017, pp. 279–283.

[36] L. Cardelli, M. Kwiatkowska, and L. Laurenti, "A Stochastic Hybrid Approximation for Chemical Kinetics Based on the Linear Noise Approximation", in *Computational Methods in Systems Biology*, E. Bartocci, P. Lio, and N. Paoletti, Eds., Cham: Springer International Publishing, 2016, pp. 147–167, ISBN: 978-3-319-45177-0.

[37] N. Cauchi and A. Abate, "Benchmarks for Cyber-Physical Systems: A Modular Model Library for Building Automation Systems", in *Proceedings of the Sixth IFAC Conference on Analysis and Design of Hybrid Systems (ADHS)*, vol. 51, 2018, pp. 49–54.

[38] ——, "StocHy: Automated Verification and Synthesis of Stochastic Processes", in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, T. Vojnar and L. Zhang, Eds., Cham: Springer International Publishing, 2019, pp. 247–264, ISBN: 978-3-030-17465-1.

[39] N. Cauchi, K. A. Hoque, A. Abate, and M. Stoelinga, "Efficient Probabilistic Model Checking of Smart Building Maintenance using Fault Maintenance Trees", in *Proceedings of the fourth ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys)*, ACM, 2017, p. 24.

[40] N. Cauchi, K. A. Hoque, M. Stoelinga, and A. Abate, "Maintenance of Smart Buildings Using Fault Trees", *ACM Transactions on Sensor Networks (TOSN)*, vol. 14, no. 3-4, 28:1–28:25, 2018, ISSN: 1550-4859.

[41] N. Cauchi, L. Laurenti, M. Lahijanian, A. Abate, M. Kwiatkowska, and L. Cardelli, "Efficiency Through Uncertainty: Scalable Formal Synthesis for Stochastic Hybrid Systems", in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, New York, NY, USA: ACM, 2019, pp. 240–251, ISBN: 978-1-4503-6282-5.

[42] N. Cauchi, K. Macek, and A. Abate, "Model-Based Predictive Maintenance in Building Automation Systems with User Discomfort", *Energy*, vol. 138, pp. 306–315, 2017.

[43] M. E. Cholette, H. Yu, P. Borghesani, L. Ma, and G. Kent, "Degradation Modeling and Condition-Based Maintenance of Boiler Heat Exchangers using Gamma Processes", *Reliability Engineering & System Safety*, vol. 183, pp. 184–196, 2019.

[44] D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte, *et al.*, "EnergyPlus: Creating a New-Generation Building Energy Simulation Program", *Energy and Buildings*, vol. 33, no. 4, pp. 319–331, 2001.

[45] I. Cupeiro Figueroa, J. Drgoňa, M. Abdollahpouri, D. Picard, and L. Helsen, "State Observer for Optimal Control using White-box Building Models", in *Fifth International High Performance Buildings Conference, Purdue, USA, July 9-12, 2018*, 2018.

[46] F. Dannenberg, M. Kwiatkowska, C. Thachuk, and A. J. Turberfield, "DNA Walker Circuits: Computational Potential, Design, and Verification", in *International Workshop on DNA-Based Computers*, Springer, 2013, pp. 31–45.

[47] A. David and S. Larry, "The Least Variable Phase Type Distribution is Erlang", *Stochastic Models*, vol. 3, no. 3, pp. 467–473, 1987.

[48] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A Storm is Coming: A Modern Probabilistic Model Checker", in *Computer Aided Verification (CAV)*, R. Majumdar and V. Kunčak, Eds., Cham: Springer International Publishing, 2017, pp. 592–600, ISBN: 978-3-319-63390-9.

[49] J. Deng, R. Yao, W. Yu, Q. Zhang, and B. Li, "Effectiveness of the Thermal Mass of External Walls on Residential Buildings for Part-Time Part-Space Heating and Cooling using the State-Space Method", *Energy and Buildings*, 2019, ISSN: 0378-7788.

[50] T. van Dijk and J. van de Pol, "Sylvan: Multi-Core Framework for Decision Diagrams", *International Journal on Software Tools for Technology Transfer*, vol. 19, no. 6, pp. 675–696, 2017.

[51] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate Model Predictive Building Control via Machine Learning", *Applied Energy*, vol. 218, pp. 199–216, 2018.

[52] T. H. Durkin, "Boiler System Efficiency", *ASHRAE Journal*, vol. 48, no. 7, p. 51, 2006.

[53] M. H. Everdij and H. A. Blom, "Hybrid Petri Nets with Diffusion that have Into-Mappings with Generalised Stochastic Hybrid Processes", in *Stochastic Hybrid Systems*, Springer, 2006, pp. 31–63.

[54] I. K. Faisal and M. H. Mahmoud, "Risk-Based Maintenance (RBM): a Quantitative Approach for Maintenance or Inspection Scheduling and Planning", *Journal of Loss Prevention in the Process Industries*, vol. 16, no. 6, pp. 561–573, 2003, ISSN: 0950-4230.

[55] M. Fan, Z. Zeng, E. Zio, R. Kang, and Y. Chen, "A Stochastic Hybrid Systems Model of Common-Cause Failures of Degrading Components", vol. 172, Elsevier, 2018, pp. 159–170.

[56] A. Fatima, A. Kodjo, A. Cardenas, D. Yves, and K. Sousso, "Comparison and simulation of building thermal models for effective energy management", *Smart Grid and Renewable Energy*, vol. 6, no. 4, pp. 95–112, 2015.

[57] L. Feng, C. Wiltsche, L. Humphrey, and U. Topcu, "Controller Synthesis for Autonomous Systems Interacting with Human Operators", in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, ACM, 2015, pp. 70–79.

[58] A. Foucquier, S. Robert, F. Suard, L. Stéphan, and A. Jay, "State of the Art in Building Modelling and Energy Performances Prediction: A Review", *Renewable and Sustainable Energy Reviews*, vol. 23, pp. 272–288, 2013.

[59] M. Fujita, P. C. McGeer, and J.-Y. Yang, "Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation", *Formal Methods in System design*, vol. 10, no. 2-3, pp. 149–169, 1997.

[60] M. Gao, K. Wang, and L. He, "Probabilistic Model Checking and Scheduling Implementation of an Energy Router System in Energy Internet for Green Cities", *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1501–1510, 2018.

[61] R. Givan, S. Leach, and T. Dean, "Bounded-Parameter Markov Decision Processes", *Artificial Intelligence*, vol. 122, no. 1-2, pp. 71–109, 2000.

[62] T. T. Gorecki, F. A. Qureshi, and C. N. Jones, "OpenBuild: an Integrated Simulation Environment for Building Control", in *2015 IEEE Conference on Control Applications (CCA)*, IEEE, 2015, pp. 1522–1527.

[63] S. Goyal, H. A. Ingley, and P. Barooah, "Zone-Level Control Algorithms Based on Occupancy Information for Energy Efficient Buildings", in *American Control Conference (ACC), 2012*, IEEE, 2012, pp. 3063–3068.

[64] M. Grant and S. Boyd, *CVX: Matlab Software for Disciplined Convex Programming, version 2.1*, `http://cvxr.com/cvx`, 2014.

[65] M. Gribaudo and A. Remke, "Hybrid Petri Nets with General One-Shot Transitions", *Performance Evaluation*, vol. 105, pp. 22–50, 2016.

[66] B. Grünbaum, V. Klee, M. A. Perles, and G. C. Shephard, *Convex Polytopes*. Springer, 1967, vol. 16.

[67] D. Guck, T. Han, J.-P. Katoen, and M. R. Neuhäußer, "Quantitative Timed Analysis of Interactive Markov Chains", in *NASA Formal Methods Symposium*, Springer, 2012, pp. 8–23.

[68] D. Guck, H. Hatefi, H. Hermanns, J.-P. Katoen, and M. Timmer, "Modelling, Reduction and Analysis of Markov Automata", in *Quantitative Evaluation of Systems (QEST), 2013 Tenth International Conference on*, Springer, 2013, pp. 55–71.

[69] D. Guck, J. Spel, and M. Stoelinga, "DFTCalc: Reliability Centered Maintenance via Fault Tree Analysis (Tool Paper)", in *International Conference on Formal Engineering Methods*, Springer, 2015, pp. 304–311.

[70] M. Gustin, R. S. McLeod, and K. J. Lomas, "Forecasting Indoor Temperatures During Heatwaves using Time Series Models", *Building and Environment*, vol. 143, pp. 727–739, 2018.

[71] S. Haesaert, N. Cauchi, and A. Abate, "Certified Policy Synthesis for General Markov Decision Processes: An Application in Building Automation Systems", *Performance Evaluation*, vol. 117, pp. 75–103, 2017.

[72] S. Haesaert and S. Soudjani, "Achievements in Correct-by-design Control for Stochastic Systems", in *Proceedings of the Fifth International Workshop on Symbolic-Numeric Methods for Reasoning About CPS and IoT*, ser. SNR '19, New York, NY, USA: ACM, 2019, pp. 12–15, ISBN: 978-1-4503-6697-7.

[73] S. Haesaert, S. E. Z. Soudjani, and A. Abate, "Verification of General Markov Decision Processes by Approximate Similarity Relations and Policy Refinement", *SIAM Journal on Control and Optimization*, vol. 55, no. 4, pp. 2333–2367, 2017.

[74] E. M. Hahn, A. Hartmanns, H. Hermanns, and J.-P. Katoen, "A Compositional Modelling and Analysis Framework for Stochastic Hybrid Systems", *Formal Methods in System Design*, vol. 43, no. 2, pp. 191–232, 2013.

[75] E. M. Hahn, V. Hashemi, H. Hermanns, and A. Turrini, "Exploiting Robust Optimization for Interval Probabilistic Bisimulation", in *Quantitative Evaluation of Systems*, G. Agha and B. Van Houdt, Eds., Cham: Springer International Publishing, 2016, pp. 55–71, ISBN: 978-3-319-43425-4.

[76] E. M. Hahn, Y. Li, S. Schewe, A. Turrini, and L. Zhang, "IscasMc: A Web-Based Probabilistic Model Checker", in *FM 2014: Formal Methods*, C. Jones, P. Pihlajasaari, and J. Sun, Eds., Cham: Springer International Publishing, 2014, pp. 312–317, ISBN: 978-3-319-06410-9.

[77] Z. Han, R. X. Gao, and Z. Fan, "Occupancy and Indoor Environment Quality Sensing for Smart Buildings", in *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, 2012, pp. 882–887.

[78] H. Harb, N. Boyanov, L. Hernandez, R. Streblow, and D. Müller, "Development and Validation of Grey-Box Models for Forecasting the Thermal Response of Occupied Buildings", *Energy and Buildings*, vol. 117, pp. 199–207, 2016.

[79] A. Hartmanns and H. Hermanns, "The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification", in *Tools and Algorithms for the Construction and Analysis of Systems: 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, E. Ábrahám and K. Havelund, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 593–598.

[80] M. Hasanbeig, A. Abate, and D. Kroening, "Certified Reinforcement Learning with Logic Guidance", *CoRR*, vol. abs/1902.00778, 2019. arXiv: 1902.00778.

[81] M. Hekmatnejad and G. Fainekos, "Optimal Multi-Valued LTL Planning for Systems with Access Right Levels", in *2018 Annual American Control Conf. (ACC)*, IEEE, 2018, pp. 2363–2370.

[82] H. Hermanns and L. Zhang, "From Concurrency Models to Numbers", in *Nato Science for Peace and Security Series*, IOS Press, 2011.

[83] K. A. Hoque, O. A. Mohamed, and Y. Savaria, "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, 2015, pp. 1635–1640.

[84] J. Hu, J. Lygeros, and S. Sastry, "Towards a Theory of Stochastic Hybrid Systems", in *International Workshop on Hybrid Systems: Computation and Control (HSCC)*, Springer, 2000, pp. 160–173.

[85] International Energy Agency, *Market Report Series: Energy Efficiency 2018, Analysis and outlooks to 2040*, ed. by F. Birol, France: IEA Publications, 2018, 174 pp.

[86] A. K. Jardine, D. Lin, and D. Banjevic, "A Review on Machinery Diagnostics and Prognostics Implementing Condition-Based Maintenance", *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006.

[87] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A Bayesian Approach to Model Checking Biological Systems", in *The Seventh Conference on Computational Methods in Systems Biology (CMSB 2009)*, Springer, 2009, pp. 218–234.

[88] S. G. Johnson, *The NLopt nonlinear-optimization Package*, 2014.

[89] Y. Kabalci, "Communication Methods for Smart Buildings and Nearly Zero-Energy Buildings", in *Energy Harvesting and Energy Efficiency*, Springer, 2017, pp. 459–489.

[90] K. Kailash C. and P. Michael, *Reliability Engineering*. John Wiley & Sons, Inc., 2014, ISBN: 978-1-118-14067-3.

[91] K. Kalsi, M. Elizondo, J. Fuller, S. Lu, and D. Chassin, "Development and Validation of Aggregated Models for Thermostatic Controlled Loads with Demand Response", in *45th Hawaii International Conference on System Science (HICSS)*, IEEE, 2012, pp. 1959–1966.

[92] N. R. Kristensen, H. Madsen, and S. B. Jørgensen, "Parameter Estimation in Stochastic Grey-box Models", *Automatica*, vol. 40, no. 2, pp. 225–237, 2004, ISSN: 0005-1098.

[93] J. Krystul and H. A. Blom, "Sequential Monte Carlo Simulation of Rare Event Probability in Stochastic Hybrid Systems", *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 176–181, 2005.

[94] S. Kubba, *Handbook of Green Building Design and Construction: LEED, BREEAM, and Green Globes*. Butterworth-Heinemann, 2012.

[95] O. Kupferman and M. Y. Vardi, "Model Checking of Safety Properties", *Formal Methods in System Design*, vol. 19, pp. 291–314, 3 2001.

[96] M. Kurt and J. P. Kharoufeh, "Monotone Optimal Replacement Policies for a Markovian Deteriorating System in a Controllable Environment", *Operations Research Letters*, vol. 38, no. 4, pp. 273–279, 2010.

[97] M. Kwiatkowska, "Quantitative Verification: Models, Techniques and Tools", in *The sixth Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering: Companion Papers*, ser. ESEC-FSE companion '07, New York, NY, USA: ACM, 2007, pp. 449–458, ISBN: 978-1-59593-812-1.

[98] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic Model Checking", in *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, Springer, 2007, pp. 220–270.

[99] ——, "Quantitative Verification Techniques for Biological Processes", in *Algorithmic Bioprocesses*, Springer, 2009, pp. 391–409.

[100] ——, "PRISM 4.0: Verification of Probabilistic Real-time Systems", in *Proceedings 23rd International Conference on Computer Aided Verification (CAV)*, G. Gopalakrishnan and S. Qadeer, Eds., ser. LNCS, vol. 6806, Springer, 2011, pp. 585–591.

[101] ——, "Probabilistic Model Checking: Advances and Applications", in *Formal System Verification*, Springer, 2018, pp. 73–121.

[102] M. Lahijanian, S. B. Andersson, and C. Belta, "Formal Verification and Synthesis for Discrete-Time Stochastic Systems", *IEEE Transactions on Automatic Control*, vol. 60, no. 8, pp. 2031–2045, 2015.

[103] K. P. Lam, M. Höynck, B. Dong, B. Andrews, Y.-S. Chiou, R. Zhang, D. Benitez, J. Choi, *et al.*, "Occupancy Detection through an Extensive Environmental Sensor Network in an Open-Plan Office Building", *IBPSA Building Simulation*, vol. 145, pp. 1452–1459, 2009.

[104] K. G. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a Nutshell", *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 134–152, 1997.

[105] L. Laurenti, A. Abate, L. Bortolussi, L. Cardelli, M. Ceska, and M. Kwiatkowska, "Reachability Computation for Switching Diffusions: Finite Abstractions with Certifiable and Tuneable Precision", in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, ACM, 2017, pp. 55–64.

[106] A. Legay, B. Delahaye, and S. Bensalem, "Statistical Model Checking: An Overview.", *International Conference on Runtime Verification*, vol. 10, pp. 122–135, 2010.

[107] K. Lesser and M. Oishi, "Finite State Approximation for Verification of Partially Observable Stochastic Hybrid Systems", in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC 2015, New York, NY, USA: ACM, 2015, pp. 159–168, ISBN: 978-1-4503-3433-4.

[108] Y. Li and Z. O'Neill, "A Critical Review of Fault Modeling of HVAC Systems in Buildings", in *Building Simulation*, Springer, 2018, pp. 1–23.

[109] Z. Li, Y. Ren, L. Liu, and Z. Wang, "Parallel Algorithm for Finding Modules of Large-Scale Coherent Fault Trees", *Microelectronics Reliability*, vol. 55, no. 10, pp. 1400–1403, 2015, Proceedings of the 26$^{\text{th}}$ European Symposium on Reliability of Electron Devices, Failure Physics and AnalysisSI: Proceedings of ESREF 2015, ISSN: 0026-2714.

[110] T. Lu and M. Viljanen, "Prediction of Indoor Temperature and Relative Humidity using Neural Network Models: Model Comparison", English, *Neural Computing and Applications*, vol. 18, no. 4, pp. 345–357, 2009, ISSN: 0941-0643.

[111] J. Lygeros and M. Prandini, "Stochastic Hybrid Systems: a Powerful Framework for Complex, Large Scale Applications", *European Journal of Control*, vol. 16, no. 6, pp. 583–594, 2010.

[112] Y. Ma, A. Kelman, A. Daly, and F. Borrelli, "Predictive Control for Energy Efficient Buildings with Thermal Storage: Modeling, Stimulation, and Experiments", *Control Systems, IEEE*, vol. 32, no. 1, pp. 44–64, 2012, ISSN: 1066-033X.

[113] M. Maasoumy, A. Sangiovanni-Vincentelli, *et al.*, "Smart Connected Buildings Design Automation: Foundations and Trends", *Foundations and Trends® in Electronic Design Automation*, vol. 10, no. 1-2, pp. 1–143, 2016.

[114] K. Macek, P. Endel, N. Cauchi, and A. Abate, "Long-Term Predictive Maintenance: A Study of Optimal Cleaning of Biomass Boilers", *Energy and Buildings*, vol. 150, pp. 111–117, 2017.

[115] A. Mahdavi and F. Tahmasebi, "Predicting People's Presence in Buildings: An Empirically Based Model Performance Analysis", *Energy and Buildings*, vol. 86, pp. 349–355, 2015, ISSN: 0378-7788.

[116] T. Mancini, F. Mari, I. Melatti, I. Salvo, E. Tronci, J. K. Gruber, B. Hayes, M. Prodanovic, and L. Elmegaard, "Parallel Statistical Model Checking for Safety Verification in Smart Grids", in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, IEEE, 2018, pp. 1–6.

[117] D. C. Matisoff, D. S. Noonan, and A. M. Mazzolini, "Performance or Marketing Benefits? The Case of LEED Certification", *Environmental Science and Technology*, vol. 48, no. 3, pp. 2001–2007, 2014.

[118] S. E. Mattsson, H. Elmqvist, and M. Otter, "Physical System Modeling with Modelica", *Control Engineering Practice*, vol. 6, no. 4, pp. 501–510, 1998.

[119] H. Maula, V. Hongisto, L. Östman, A. Haapakangas, H. Koskela, and J. Hyönä, "The Effect of Slightly Warm Temperature on Work Performance and Comfort in Open-Plan Offices – a Laboratory Study", *Indoor Air*, vol. 26, no. 2, pp. 286–297, 2016.

[120] I. Mezić and T. Runolfsson, "Uncertainty Propagation in Dynamical Systems", *Automatica*, vol. 44, no. 12, pp. 3003–3013, 2008.

[121] C. Miller, H. Hu, and L. Klesch, "Hybrid Calibration Methodology for Building Energy Models Coupling Sensor Data and Stochastic Modeling", in *Technologies for Sustainability (SusTech), 2013 1st IEEE Conference on*, 2013, pp. 37–43.

[122] J. Moubray, *Reliability-Centered maintenance*. Industrial Press Inc., 1997.

[123] A. Nagy, A. Bratukhin, and T. Sauter, "Efficient Thermal Modeling for Distributed Energy Management in Industrial Buildings", in *Systems Conference (SysCon), 2015 9th Annual IEEE International*, 2015, pp. 309–316.

[124] G. Norman and D. Parker, *Quantitative Verification: Formal Guarantees for Timeliness, Reliability and Performance*. London Mathematical Society and Smith Institute, 2014.

[125] C. Nzukam, A. Voisin, E. Levrat, D. Sauter, and B. Iung, "A Dynamic Maintenance Decision Approach Based on Maintenance Action Grouping for HVAC Maintenance Costs Savings in Non-Residential Buildings", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 722–13 727, 2017.

[126] R. Palin, D. Ward, I. Habli, and R. Rivett, "ISO 26262 Safety Cases: Compliance and Assurance", pp. 12–12, 2011.

[127] A. Parisio, M. Molinari, D. Varagnolo, and K. Johansson, "A Scenario-Based Predictive Control Approach to Building HVAC Management Systems", in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, 2013, pp. 428–435.

[128] Z. Peng, Y. Lu, A. Miller, T. Zhao, and C. Johnson, "Formal Specification and Quantitative Analysis of a Constellation of Navigation Satellites", *Quality and Reliability Engineering International*, vol. 32, no. 2, pp. 345–361, 2016.

[129] A. Persily and L. de Jonge, "Carbon Dioxide Generation Rates for Building Occupants", *Indoor Air*, vol. 27, no. 5, pp. 868–879, 2017.

[130] C. Pilch and A. Remke, "HYPEG: Statistical model checking for hybrid petri nets: Tool paper", in *Proceedings of the Eleventh EAI International Conference on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS 2017, Venice, Italy: ACM, 2017, pp. 186–191.

[131] A. Prékopa, "Logarithmic Concave Measures with Application to Stochastic Programming", *Acta Scientiarum Mathematicarum*, vol. 32, pp. 301–316, 1971.

[132] S. Privara, J. Cigler, Z. Vana, F. Oldewurtel, C. Sagerschnig, and E. Zacekova, "Building Modeling as a Crucial Part for Building Predictive Control", *Energy and Buildings*, vol. 56, pp. 8–22, 2013, ISSN: 0378-7788.

[133] A. Rae, P. Robert, and H.-L. Hausen, *Software Evaluation for Certification*. McGraw-Hill, Inc., 1994.

[134] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-Physical Systems: The Next Computing Revolution", in *Design Automation Conference*, 2010, pp. 731–736.

[135] Requirements and Technical Concepts for Aviation (RTCA), "DO-331: Model-Based Development and Verification Supplement to DO-178C and DO-278A", 2011.

[136] S. M. Ross, J. J. Kelly, R. J. Sullivan, W. J. Perry, D. Mercer, R. M. Davis, T. D. Washburn, E. V. Sager, J. B. Boyce, and V. L. Bristow, *Stochastic Processes*. Wiley New York, 1996, vol. 2.

[137] P. Roy, P. Tabuada, and R. Majumdar, "Pessoa 2.0: A Controller Synthesis Tool for Cyber-Physical Systems", in *Proceedings of the fourteenth international conference on Hybrid systems: computation and control*, ACM, 2011, pp. 315–316.

[138] E. Ruijters, D. Guck, P. Drolenga, M. Peters, and M. Stoelinga, "Maintenance Analysis and Optimization via Statistical Model Checking", in *International Conference on Quantitative Evaluation of Systems, 2016 Thirteenth International Conference on*, Springer, 2016, pp. 331–347.

[139] E. Ruijters and M. Stoelinga, "Fault Tree Analysis: A Survey of the State-Of-The-Art in Modeling, Analysis and Tools", *Computer Science Review*, vol. 15, pp. 29–62, 2015.

[140] C. Sanderson and R. Curtin, "Armadillo: A Template-Based C++ Library for Linear Algebra", *Journal of Open Source Software*, 2016.

[141] C. Sanderson and R. R. Curtin, "A User-Friendly Hybrid Sparse Matrix Class in C++", in *Mathematical Software - ICMS 2018 - Sixth International Conference, South Bend, IN, USA, July 24-27, 2018, Proceedings*, 2018, pp. 422–430.

[142] M. Schmidt, M. V. Moreno, A. Schülke, K. Macek, K. Mařík, and A. G. Pastor, "Optimizing Legacy Building Operation: The Evolution into Data-Driven Predictive Cyber-Physical Systems", *Energy and Buildings*, vol. 148, pp. 257–279, 2017, ISSN: 0378-7788.

[143] A. Sharma, G. Yadava, and S. Deshmukh, "A Literature Review and Future Perspectives on Maintenance Optimization", *Journal of Quality in Maintenance Engineering*, vol. 17, no. 1, pp. 5–25, 2011.

[144] F. Shmarov, N. Paoletti, E. Bartocci, S. Lin, S. A. Smolka, and P. Zuliani, "SMT-based Synthesis of Safe and Robust PID controllers for Stochastic Hybrid Systems", in *Haifa Verification Conference*, Springer, 2017, pp. 131–146.

[145] D. Škulj, "Discrete Time Markov Chains with Interval Probabilities", *International Journal of Approximate Reasoning*, vol. 50, no. 8, pp. 1314–1329, 2009.

[146] S. E. Z. Soudjani, "Formal Abstractions for Automated Verification and Synthesis of Stochastic Systems", PhD thesis, TU Delft, 2014.

[147] S. E. Z. Soudjani, C. Gevaerts, and A. Abate, "FAUST^2: Formal abstractions of uncountable-STate STochastic processes.", in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2015, pp. 272–286.

[148] J. Souyris, V. Wiels, D. Delmas, and H. Delseny, "Formal Verification of Avionics Software Products", in *International Symposium on Formal Methods*, Springer, 2009, pp. 532–546.

[149] Star, Energy, "Energy star®", *Program Requirements for Residential*, 2010.

[150] M. Střelec, K. Macek, and A. Abate, "Modeling and Simulation of a Microgrid as a Stochastic Hybrid System", in *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, 2012, pp. 1–9.

[151] D. Sturzenegger, D. Gyalistras, V. Semeraro, M. Morari, and R. Smith, "BRCM Matlab Toolbox: Model Generation for Model Predictive Building Control", in *American Control Conference (ACC), 2014*, 2014, pp. 1063–1069.

[152] D. Sturzenegger, D. Gyalistras, M. Morari, and R. S. Smith, "Model Predictive Climate Control of a Swiss Office Building: Implementation, Results, and Cost–benefit Analysis", *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 1–12, 2016.

[153] G. Sugumar, R. Selvamuthukumaran, M. Novak, and T. Dragicevic, "Supervisory Energy-Management Systems for Microgrids: Modeling and Formal Verification", *IEEE Industrial Electronics Magazine*, vol. 13, no. 1, pp. 26–37, 2019.

[154] S. Summers and J. Lygeros, "Verification of Discrete Rime Stochastic Hybrid Systems: A Stochastic Reach-Avoid Decision Problem", *Automatica*, vol. 46, no. 12, pp. 1951–1961, 2010, ISSN: 0005-1098.

[155] R. Suttell, "Preventive HVAC Maintenance is a Good Investment", *Buildings*, vol. 100, no. 7, pp. 50–52, 2006.

[156] P. Tabuada and G. J. Pappas, "Linear Time Logic Control of Discrete-Time Linear Systems", *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006, ISSN: 0018-9286.

[157] M. M. Tehrani, Y. Beauregard, M. Rioux, J. P. Kenne, and R. Ouellet, "A Predictive Preference Model for Maintenance of a Heating Ventilating and Air Conditioning System", *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 130–135, 2015.

[158] J. Trojanova, J. Vass, K. Macek, J. Rojíček, and P. Stluka, "Fault Diagnosis of Air Handling Units", *IFAC Proceedings Volumes*, vol. 42, no. 8, pp. 366–371, 2009.

[159] W. Vesely, F. Goldberg, N. Roberts, and D. Haasl, "Fault Tree Handbook", Nuclear Regulatory Commission Washington DC, Tech. Rep., 1981.

[160] W. Vesely, M. Stamatelatos, J. Dugan, J. Fragola, J. Minarick, and J. Railsback, "Fault Tree Handbook with Aerospace Applications version 1.1", *NASA Office of Safety and Mission Assurance, NASA HQ*, 2002.

[161] A. P. Vinod, J. D. Gleason, and M. M. K. Oishi, "SReachTools: A MATLAB Stochastic Reachability Toolbox", in *22nd ACM International Conference on Hybrid Systems: Computation and Control*, ACM, 2019, pp. 33–38.

[162] M. Volk, S. Junges, and J. P. Katoen, "Fast Dynamic Fault Tree Analysis by Model Checking Techniques", *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 370–379, 2018, ISSN: 1551-3203.

[163] R. Volk, J. Stengel, and F. Schultmann, "Building Information Modeling (BIM) for Existing Buildings — Literature Review and Future Needs", *Automation in Construction*, vol. 38, pp. 109–127, 2014, ISSN: 0926-5805.

[164] V. Vyatkin, "IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design", 2007.

[165] D. Wu and X. Koutsoukos, "Reachability Analysis of Uncertain Systems using Bounded-Parameter Markov Decision Processes", *Artificial Intelligence*, vol. 172, no. 8-9, pp. 945–954, 2008.

[166] S. Wu and J.-Q. Sun, "A Physics-Based Linear Parametric Model of Room Temperature in Office Buildings", *Building and Environment*, vol. 50, pp. 1–9, 2012, ISSN: 0360-1323.

[167] J. Yang, A. Pantazaras, K. A. Chaturvedi, A. K. Chandran, M. Santamouris, S. E. Lee, and K. W. Tham, "Comparison of Different Occupancy Counting Methods for Single System-Single Zone Applications", *Energy and Buildings*, vol. 172, pp. 221–234, 2018.

[168] H. L. Younes and R. G. Simmons, "Probabilistic Verification of Discrete Event Systems using Acceptance Sampling", in *International Conference on Computer Aided Verification*, Springer, 2002, pp. 223–235.

[169] B. Zhang and J. Baillieul, "Control and Communication Protocols Based on Packetized Direct Load Control in Smart Building Microgrids", *Proceedings of the IEEE*, vol. 104, no. 4, pp. 837–857, 2016.

[170] H. Zhang and D. W. R. Marsh, "Generic Bayesian Network Models for Making Maintenance Decisions from Available Data and Expert Knowledge", *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 232, no. 5, pp. 505–523, 2018.

[171] Q. Zhou, S. Wang, and Z. Ma, "A Model-Based Fault Detection and Diagnosis Strategy for HVAC Systems", *International Journal of Energy Research*, vol. 33, no. 10, pp. 903–918, 2009, ISSN: 1099-114X.

[172] Y. Zhu and L. Guo, "Sequential Preventive Maintenance Interval Determination based on Monte Carlo Method for Deteriorating Systems", in *2017 Second International Conference on Reliability Systems Engineering (ICRSE)*, 2017, pp. 1–5.