Probabilistic Verification of Satellite Systems for Mission Critical Applications

Yu Lu

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow

April 2016

Abstract

In this thesis, we present a quantitative approach using probabilistic verification techniques for the analysis of reliability, availability, maintainability, and safety (RAMS) properties of satellite systems. The subject of our research is satellites used in mission critical industrial applications. A strong case for using probabilistic model checking to support RAMS analysis of satellite systems is made by our verification results. This study is intended to build a foundation to help reliability engineers with a basic background in model checking to apply probabilistic model checking to small satellite systems.

We make two major contributions. One of these is the approach of RAMS analysis to satellite systems. In the past, RAMS analysis has been extensively applied to the field of electrical and electronics engineering. It allows system designers and reliability engineers to predict the likelihood of failures from the indication of historical or current operational data. There is a high potential for the application of RAMS analysis in the field of space science and engineering. However, there is a lack of standardisation and suitable procedures for the correct study of RAMS characteristics for satellite systems. This thesis considers the promising application of RAMS analysis to the case of satellite design, use, and maintenance, focusing on its system segments. Data collection and verification procedures are discussed, and a number of considerations are also presented on how to predict the probability of failure.

Our second contribution is leveraging the power of probabilistic model checking to analyse satellite systems. We present techniques for analysing satellite systems that differ from the more common quantitative approaches based on traditional simulation and testing. These techniques have not been applied in this context before. We present the use of probabilistic techniques via a suite of detailed examples, together with their analysis. Our presentation is done in an incremental manner: in terms of complexity of application domains and system models, and a detailed PRISM model of each scenario. We also provide results from practical work together with a discussion about future improvements.

Acknowledgements

First and foremost, I greatly appreciate to my first PhD supervisor Dr Alice A. Miller and my second PhD supervisor Dr Gethin Norman, for their invaluable guidance and patience.

I also would like to thank all my colleagues and collaborators, and additional thanks to Professor Christopher W. Johnson for his help.

Many thanks to my parents and my wife for their numerous support.

I am grateful for the full PhD Prize Studentship provided from the Scottish Informatics and Computer Science Alliance (SICSA).

Finally, I also would like to thank my PhD examiners, Dr Clare Dixon and Dr John O'Donnell. My thesis has been greatly improved based on their comments.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Yu Lu*)

To my family.

Table of Contents

List of Tables

List of Figures

Part I

Background and Preliminaries

Chapter 1

Introduction

1.1 Motivation

Satellite based systems appear in almost all aspects of our daily lives. In industry, satellites are already in operational use as part of commercial and non-critical applications such as fleet management, customer information, and selective door operation. They promise to form a core component for national and international critical infrastructures. Satellite constellations, such as GPS in the US, Galileo in Europe, and BeiDou in China, provide key information (e.g., location, timing, and pictures) for a variety of mission critical applications from guidance and navigation of unmanned vehicles, to space surveillance for disaster areas. The European Commission has recently certified Galileo-based extensions to GPS for railways, maritime, and aviation sectors. Therefore, a wide range of mission critical applications are completely dependent on satellite based infrastructures. However, satellites are vulnerable to physical and cyber attacks as well as accidental faults. System designers, engineers, and end users are typically not aware of these possible failures identified to satellite based infrastructures and this will affect the successfulness of the underlying missions.

Because of the mission critical nature of satellite systems, it is essential to guarantee not just qualitative correctness but also a variety of quantitative characteristics, such as reliability, availability, maintainability, and safety (RAMS), and to check if these systems meet the design requirements. Typically, RAMS properties are of paramount importance in the analysis of whether and how well satellite systems are capable of completing a particular mission. In general, reliability denotes a system's ability to continue to comply with its specification over its useful life; availability denotes the ability of the system to remain in that functioning state; maintainability is a design

property of the system and is determined by the ease at which the system can be repaired or maintained; finally Safety denotes the system does not cause harm to people, the environment, or any other assets during its life cycle - during normal use and also for foreseeable misuse. RAMS analysis has been indispensable in the design phase of satellites in order to achieve minimum failures or to increase mean time between failures (MTBF) and thus to plan maintainability strategies, optimise reliability and maximise availability.

There are a number of quantitative approaches for RAMS analysis of systems, including but not limited to failure modes and effects analysis (FMEA) [8, 101], reliability block diagram (RBD analysis [37, 102], fault tree analysis (FTA) [116, 102], and state space method [131, 124]. In particular, probabilistic model checking is a state space method, involving an exhaustive exploration of states and their transition of components of a repairable system, or interacting subsystems in the system. In general, this technique is based on the study of Markov models, and is appropriate to be applied to satellite systems and can be used to support some of other quantitative approaches. For probabilistic model checking, non-Markov models can also be modelled, but calculation is non-trivial unless a semi-Markov model can be established. A semi-Markov chain is a model in which state holding times are governed by general distributions, which is a natural extension of CTMCs. So, we have also considered the problem of an approximation to semi-Markov models using Markov models.

Verification is a process of analysing whether a system satisfies its specifications. Verification of RAMS requirements for computerised systems has been an active research area of Computer Science, Systems Engineering, and Software Engineering for decades. In the context of satellite systems, the verification problem appears to be difficult and not one that can be tackled completely by the current state of art verification techniques. This is due to the fact that modelling and reasoning about such systems involves a combination of multiple dimensions that must be formalised and verified simultaneously, and requires specifying and proving new and more complex system properties which are often too subtle to be expressed. Generally, verification techniques for space and satellite systems include testing, simulation, theorem proving, and model checking.

Testing is performed on an actual system, whereas simulation is performed on an abstract model of the system. Testing and simulation both involve checking whether the outputs are as expected based on certain inputs. These techniques are a cost-effective method to find system errors. But, checking all possible interactions, exe-

cutions, and faults is almost impossible due to the fact that no amount of testing and simulation is completely exhaustive. Thus, they can only show the presence of system errors, not their absence, and also only ensure that the system works for the given inputs.

Formal verification involves the use of mathematical techniques to ensure that a design conforms to some precisely expressed notion of functional correctness. It aims at providing a rigid and thorough means of evaluating the correctness of a system via techniques such as theorem proving, deductive reasoning, and model checking. Model checking is an automatic technique that can establish, via exhaustive analysis of the model of a system, whether its behaviour is correct with respect to a given specification. This involves exploring the underlying state space of the model, and specifying properties via some formal logic such as temporal logic. It has been successfully applied to numerous computer systems and their applications, including both software and hardware systems. Model checking is considered to be a powerful extension of traditional verification techniques. Satellite based systems raise numerous new research challenges, such as:

- The need for formal verification has to be initiated from design time for developing satellite systems. However, formal specification of such systems is more difficult than non-formal techniques such as simulation and fault trees.
- Satellite systems have a dynamically changing communication topology due to their physical mobility, which affects both energy usage and reliability of communication. Model checking techniques must accommodate these features.
- Satellites operate in an unpredictable, unreliable, and dynamic environment and sometimes need to respond quickly. Stochastic models must be developed to capture the impact of uncertainty and the probabilistic behaviour of underlying systems and external environment.
- Quantitative analysis techniques are required to predict the likelihood of failures as well as the energy cost of satellite operations over time and to select the best operation strategies given certain constraints.
- How can we ensure that our approaches are flexible and scalable to realistic space and satellite systems?

Some simple properties of a typical satellite have been verified using a single verification tool SPIN) [54]. However, a single formal approach cannot be used to specify

and verify more complex properties involving several dimensions. These dimensions include real-time aspects, environmental uncertainty, communication uncertainty, path planing, obstacle avoidance behaviours of small satellites, etc.

The correctness of a satellite system is fundamentally critical. Simulation is commonly considered to be the most successful verification technique for analysing the system?s behaviour. However, simulation alone is not sufficient to verify the RAMS requirements that are expressed in formal logic such as: “property P1 will be true for all experiments”, or “is property P2 true whenever property P3 is true”. The disadvantages of simulation are basically due to 2 reasons: (1) the analysis are not exhaustive, because only a part of the whole available cases will be dealt with by software simulations; (2) the outcome is basically simple information which is infeasible to verify complex cases.

This issue practically needs feasible approaches to the problem of analysing complex satellite system behaviour. Nowadays, model checking can be an extremely advantageous alternative technique to simulation. This is due to that model checking is not just complete in logic and rigorous in mathematics but flexible for the modelling and specification of complex behaviours. Besides, attempting to verifying RAMS properties of satellite systems has not been in a systematic manner. Simulation is unable to deal with the fast development in the design complexity of satellite systems alone, therefore we consider that model checking is the right tool and it is timely to apply model checking to this domain.

1.2 Aims and Benefits

Our aims are to apply formal methods in satellite for mission-critical applications for practical and complex scenarios so as to achieve decision support of design for satellite systems. This is done by verifying reliability, availability, maintainability, and safety properties relevant to a system’s specification. Model checking, which is a formal method and a tool for decision support, enables us check logical properties for all states that relate to the specification, and thus allows us to achieve our aims.

We also aim to help the space and reliability community by providing a more formal and rigorous assessment of these systems, in particular, by using probabilistic model checking to assess the likelihood and consequences of failures to their operations. Probabilistic model checking, also known as stochastic model checking, is a generalisation of model checking for verifying quantitative properties of systems that

exhibit stochastic behaviours. Models obtained by this technique are normally extensions or variants of Markov processes or timed automata, extended with costs and rewards that estimate resources and their usage during operation.

There several benefits of the work to users. First, it sought to represent RAMS properties using appropriate formal logic in order to understand what it is that makes each successful in predicting likelihood of failures, and how they might achieve minimum failures or to increase mean time between failures (MTBF) and thus to plan maintainability strategies, optimise reliability and maximise availability. Second, it provided novel RAMS analysis for satellite systems that nowadays form a core component for national and international critical infrastructures. Third, it developed efficient approaches to a representative configuration of satellite systems and the underlying practical environment, rather than expensive experimental simulation and testing. It was expected that the outcome would be close to actual scenario, and informative to system engineers.

Probabilistic model checking that has focused on computerised systems has not been conducted in practical satellite for mission-critical applications, and with such a complex behaviour. The techniques have been successfully applied to a variety of application domains, both to ensure correctness and to find optimal configurations of systems. Several popular areas of application include: safety-critical applications, performance analysis, and scheduling and optimisation. Through a number of industrial case studies, we demonstrate that it is possible to use the technique to mission critical applications. In this context, the effects of proposed changes to a satellite system can be first checked via a model, rather than via expensive prototypes. Reliability, availability, maintainability, and safety properties of satellite systems can be expressed in probabilistic temporal logic, and proved via probabilistic model checking.

1.3 Contributions

Current state-of-the-art verification techniques (e.g., testing, simulation, and qualitative model checking) appear to be unable to cope with the verification demand introduced by satellite based systems for mission critical applications, because reasoning about such systems requires combinations of multiple dimensions such as required for quantitative, continuous and stochastic behaviour to be considered, and requires proving properties which are subtle to express.

1.3.1 Thesis Statement

It is feasible, useful, and efficient to apply probabilistic verification, particularly probabilisitic model checking, during the design phase, to perform predictive analysis of reliability, availability, maintainability, and safety (RAMS) properties of mission critical systems that are reliant on satellites. Our main contribution is applying a range of probabilisitic verification techniques to various RAMS properties to satellite-based systems. These techniques have not been applied in this context before, and are demonstrated via a series of examples.

1.3.2 Considerations of Formal Models

In order to demonstrate this, we have investigated and dealt with several dimensions as follows:

- *Mobility*: Components of satellite systems are mobile. The formalism of process algebras allows mobility to be expressed by the transmission of names or terms. For instance, probabilisitic π-calculus is a good candidate, and can input directly into PRISM.

- *Concurrency*: all components of satellite systems exist and operate simultaneously. It is natural to model concurrency via parallel composition and nondeterminism in many models. We demonstrate this approach via reactive modules in PRISM.

- *Uncertainty*: satellites commonly have to work in uncertain circumstances, such as in a dynamic environment and with unreliable communication. Various probabilistic models exist to capture uncertain behaviours of systems and environments, and we use technically mature ones that are supported in PRISM.

- *Real time*: precise constraints on the timing of events are needed for the correct modelling satellite systems. (Probabilistic) timed automata allows us to specify delays that occur during transitions between states.

- *Cost of energy*: Models can be extended in PRISM with costs and rewards by associating real values with certain states or transitions, so allowing energy use to be measured and compared.

1.3.3 Results of the work

Formal verification of RAMS properties is difficult due to the complexity of the systems, and it is an useful alternative to, and a powerful extension of simulation and testing. There are two major contributions of the work. The first one is formalising and analysing RAMS properties of satellite systems for mission-critical applications, the other is analysing satellite systems via probabilistic model checking. We summarise the detailed results of our research as follows:

- The likelihood of failures from the indication of historical or current operational data have be predicted;
- System RAMS properties to satellite design, use, and maintenance, based on its system segments have been formalised and analysed;
- Mission RAMS properties considering different missions that are reliant on satellites have been analysed and verified;
- How such an approach can provide a useful tool for designers and reliability engineers of satellites has been demonstrated.

Overall, we have shown the ways in which probabilistic model checking can be applied to satellite systems. Previous work has also been reported in this thesis on analysing the reliability, availability, maintainability, and safety properties of satellite systems using non-formal techniques such as simulation and fault trees. We have illustrated our work by detailing several complete end-to-end design-time verification process including all models and specifications. In this thesis, each scenario is inspired by the real-world problem. By modelling existing scenarios, we have learnt to make abstraction of realistic satellite systems. Our verification results for our probabilistic models showed that the state space of our different models can be tractable and verification on them can be feasible.

The starting point of our probabilistic verification technique is a collection of models of satellite systems under consideration. Our system models obtained some expertise in finding appropriate abstractions to obtain smaller system models and to state RAMS properties in the logical formalism used. More specifically, our probabilistic models have been analysed and validated statically via peer review carried out by a couple of reliability and space professionals before being formally verified. Empirical studies indicate that peer review provides an effective technique for the validation.

Although peer review is almost complete manual, it is a rather useful technique. We believe that the results from the formal methods conform to reality.

Given the fact that verification using model-based techniques is only as good as the model of the system [12], we believe that it is difficult to achieve absolute guaranteed correctness for realistic systems. Our technique verifies a satellite system model, and not the actual system itself, therefore our approach is preferred to be used as a complementary technique. Despite the above limitations we conclude that our results can provide a significant level of insight into a system design in terms of reliability, availability, maintainability, and safety that are relevant to formalise the requirements for realistic satellite systems.

Specification validation has received little attention in verification in practice. To ensure our model checking results are meaningful, we have to ensure that both the model and specifications unambiguously fulfill the intentions of the system designers and reliability engineers. We have employed model and property validation techniques such as peer review. We carefully construct each model and property due to that there is no systematic approach. Nevertheless, it provided us with valuable insights in practice. Furthermore, we may consider other specification validation and debugging techniques in real applications for our future research.

1.4 Organisation of Thesis

This thesis is divided into two parts. The introductory part is Part I, which covers background information on satellite systems and verification techniques. Part II presents the main work done, and discussion and conclusions. In Figure 1.1, we present an overview of the chapters and their relational structure.

In Chapters 2 and 3 of this thesis, we provide a background on satellite based systems and probabilistic model checking respectively and highlight that no previous research has been conducted on the application of (probabilistic) model checking to explicitly specify and verify reliability, availability, maintainability, safety properties of systems, particularly space and satellite systems. In addition, we summarise the main issues and challenges that have arisen from the research in the area of modelling and verification of satellite systems for mission critical applications. A short review of the traditional and quantitative verification techniques and work done on space and satellite systems and their RAMS properties is also presented.

The space segment of a satellite system is the most important operational compo-

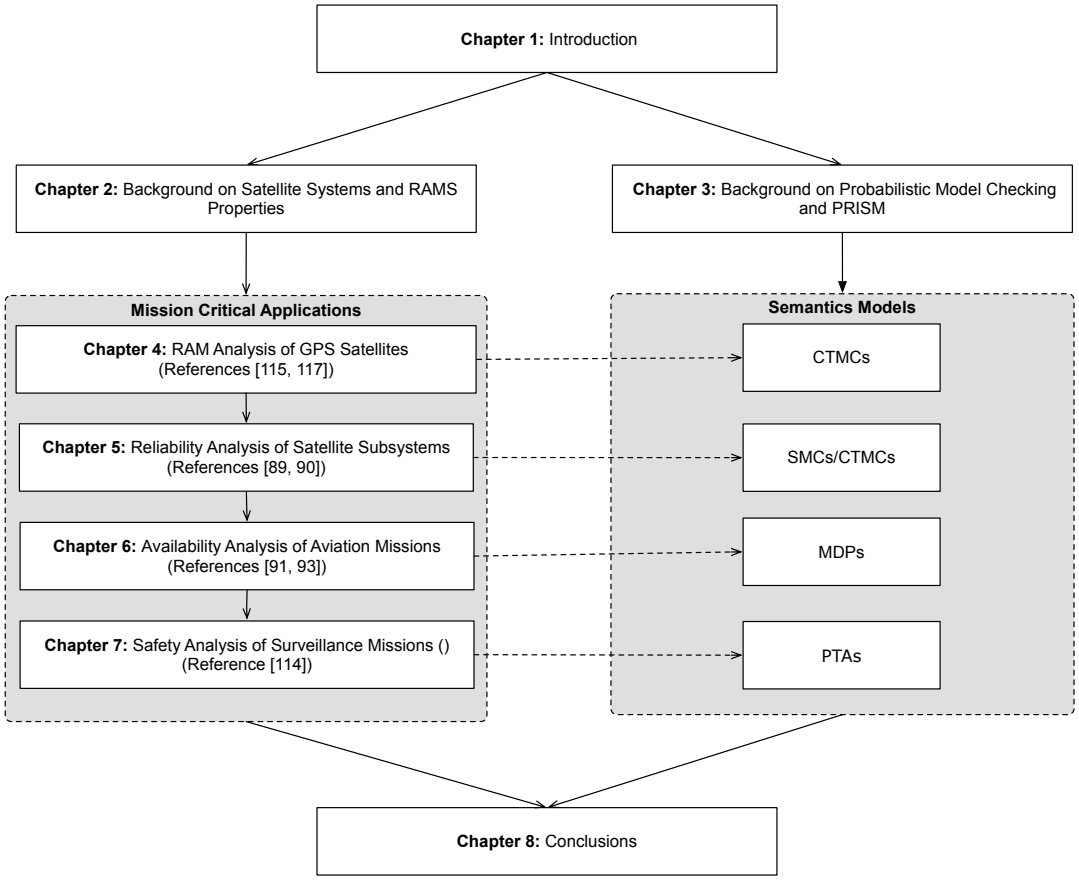


Figure 1.1: Structure of chapters of the thesis.

nent of an artificial satellite system. In Chapter 4, CTMC based formal models are constructed for two different kinds of space segment: a single satellite and a constellation of satellites, in an uncertain dynamic environment. Associated logical properties are developed, which allow us to generate time-dependent probability curves for the reliability, availability, maintainability, and safety characteristics in PRISM model checker.

Furthermore, the space segment is complex due to the fact that it consists of a large number of subsystems. Each subsystem may itself have complex and different failure modes. In addition, when analysing multi-state failure modes for satellite subsystems, the failure rates follow a general distribution such as Weibull distributions. Weibull distributions can be modelled as semi-Markov chains, which do not exhibit the exponentially distributed sojourn time. As a result, the CTMCs of Chapter 4 are not suitable for modelling systems that exhibit Weibull failures. In Chapter 5, we analyse multi-state failure modes for satellite subsystems. We define novel semi-Markov models that

characterise failure behaviours, based on Weibull failure modes inferred from realistic data sources. We approximate and encode these models using CTMCs in PRISM by using both hyper exponential distributions and Erlang distributions in order to answer meaningful questions about the reliability of these subsystems.

In Chapter 6, we consider not just the space segment but also the case of a particular mission that involves all three operational components, including a control segment and a user segment. The mission uses satellite based positioning systems in aviation sectors for aircraft guidance and navigation. We construct a formal model of the GNSS based positioning system for this application in the probabilistic π-calculus, a process algebra which supports modelling of concurrency, uncertainty, and mobility. We then encode our model using Markov decision processes (MDPs), and specify and analyse logical properties relating to system reliability and availability. We discuss how this approach can be successfully extended to the maritime and railway sectors for guiding and navigating ships and trains in an uncertain environment.

Another important and successful application of satellite systems is to provide surveillance information (e.g., images) in disaster areas. In Chapter 7, we demonstrate a case study involving path planning of a satellite within a limited time period for an emergency like the Wenchuan earthquake. The existing satellites have to be effectively and efficiently employed to rapidly and continuously cover and monitor the affected area during a short time period. In this chapter, we construct different probabilistic timed automata (PTAs) models of a single satellite in both cases of global and local path planning, and use PRISM to verify real-time and safety properties related to three essential parameters which have to be taken into consideration simultaneously for controlling the satellite. These parameters are: (1) the minimum time (2) the maximum observation coverage time, and (3) the minimum fuel cost, into the logical specification of the requirements.

In Chapter 8, we give a summary and our conclusions. We also emphasise our contributions, and outline the key challenges in the area, and suggest several ideas for related future work. Following the last chapter we present several appendices, which contain a list of abbreviations, all PRISM reactive modules of the formal models presented in the thesis.

1.5 List of Publications

Some of the material presented within this thesis has previously been published in the following joint authored papers. The majority of the technical content of these papers was carried out by myself.

1.5.1 Journal Articles

- Yu Lu, Alice A. Miller, and Christopher W. Johnson, "Towards the PRISM Encoding of Weibull Models for Satellite Subsystems, In Preparation. (Chapter 5)

- Yu Lu, Zhaoguang Peng, Alice A. Miller, Tingdi Zhao, and Christopher W. Johnson, "How reliable is satellite navigation for aviation? Checking availability properties with probabilistic verification", Reliability Engineering & System Safety, Volume 144, December 2015, pages 95-116, Elsevier. (Chapter 6)

- Zhaoguang Peng, Yu Lu, Alice A. Miller, Tingdi Zhao, and Christopher W. Johnson, "Formal Specification and Quantitative Analysis of a Constellation of Navigation Satellites", Quality and Reliability Engineering International, Volume 32, Issue 2, pages 345-361, March 2016, Wiley. (Chapter 4)

1.5.2 Conference Proceedings

- Yu Lu, Alice A. Miller, Ruth Hoffmann, and Christopher W. Johnson, "Towards the Automated Verification of Weibull Distributions for Systems Failure Rates", in Proceedings of the Joint 21st International Workshop on Formal Methods for Industrial Critical Systems and 16th International Workshop on Automated Verification of Critical Systems (FMICS-AVoCS 2016), Lecture Notes of Computer Science, Volume 9933, Springer, September 2016. (Chapter 5)

- Yu Lu, Zhaoguang Peng, and Alice Miller, "Uncertainty Analysis of Phased Mission Systems with Probabilistic Timed Automata", in Proceedings of the 7th IEEE International Conference on Prognostics and Health Management (PHM 2016), pages 1-8, IEEE, June 2016. (Chapter 7)

- Yu Lu, Alice A. Miller, Christopher W. Johnson, Zhaoguang Peng, and Tingdi Zhao, "Availability Analysis of Satellite Positioning Systems for Aviation using

the PRISM Model Checker", in Proceedings of the 17th IEEE International Conference on Computational Science and Engineering (CSE 2014), pages 704-713, IEEE, December 2014. (Chapter 6)

- Zhaoguang Peng, Yu Lu, Alice A. Miller, Christopher W. Johnson, and Tingdi Zhao, "A Probabilistic Model Checking Approach to Analysing Reliability, Availability, and Maintainability of a Single Satellite System", in Proceedings of the 7th European Modelling Symposium (EMS 2013), pages 611-616, IEEE, November 2013. (Chapter 4)

Chapter 2

Satellite Based Systems

Some materials in this chapter are included in the papers [93, 117]. We introduce some background material related to satellite based systems. Specifically, the components of target satellite systems that we model are described in detail, as well as the related desired systems characteristics that need to be specified and verified.

This chapter is organised as follows. In Section 2.1, we give an overview of the underlying satellite systems, especially the global navigation satellite systems (GNSS), which is one of the most important satellite systems. In Section 2.2, we explain different components of satellite systems, which we focus on in the thesis. In Section 2.3, the usage of satellite based infrastructures for industrial critical systems, especially mission critical applications, and its RAMS characteristics are covered. Finally, in Section 2.4 we discuss available verification techniques and analyse related work in the area of verification of satellite and space systems.

2.1 Introduction

With the emergence of efficient, high-performance, and low cost satellites, earth orbiting satellites are often deployed in satellite constellations and space systems to ensure reliable and dependable missions. These kinds of satellites have played an essential part in both civil and military contexts, and support a wide range of applications ranging from navigation of ground, air, and marine assets to surveillance of disaster areas. Most of these applications are both safety-critical and mission-critical, thus they heavily depend on these satellite based infrastructures. A group of artificial satellites which work in concert is known as a satellite constellation. A satellite constellation is a number of satellites with coordinated ground coverage, operating together under shared

control, synchronised so that they overlap in coverage and complement rather than interfere with other's satellites coverage.

A satellite based navigation system is a satellite constellation consisting of a number of artificial satellites that provide autonomous geospatial positioning or monitoring with global or regional coverage. They have been developed since the 1970s and are one of the most successful applications of satellites. In particular, a navigation satellite system with global coverage is referred to as a global navigation satellite system (GNSS). Leading international projects include the United States' Global Position System (GPS) and Russia's GLObal NAvigation Satellite System (GLONASS), both of which are fully operational. In addition, China is expanding its regional BeiDou navigation system into the global compass navigation system, and the European Union's Galileo positioning system is a GNSS in the initial deployment phase. Both of these systems are expected to be fully operational in the next decade. Other countries such as India, France, and Japan are in the process of developing their own regional navigation systems. See [61] for a good overview of navigation satellite systems.

A satellite is designed to a functional requirement and it is important that it satisfies this requirement. However it is also desirable that the satellite should be predictably available and this depends upon its reliability and availability. We aim to help the military or civil end users of satellites to assess the likelihood and consequences of fault or failure of satellite components in their operations. Reliability, availability, maintainability, and safety (RAMS) analysis has been indispensable in the design phase of navigation satellite systems. It is required to achieve minimum failures or to increase mean time between failures ($MTBF$) and thus to plan maintenance strategies, optimise reliability and maximise availability. The question of how to select optimal configurations and maintenance plans and underlying resources in order to satisfy requirements and improve efficiency is a key research question. This concern calls for effective solutions to the challenges of verifying large and complex navigation systems.

2.2 Components of Systems

Generally, a satellite based system consists of three major parts: the space segment, control segment and user segment. Recent theoretical research and standards have added a fourth segment to the satellite navigation system, which is the environment. The Galileo navigation system includes an environment segment in the composition of its navigation system. Although not an explicit part of the navigation system, the

environment segment is implied in the GPS system. To be conservative, the three traditional segments are used in this thesis as the components of the study, and the environmental segment is treated as an influencing factor on the system. Failure of any subsystem will lead to errors in final positioning of the user segment. Figure 2.1 is a schematic diagram of the segments of satellite systems.

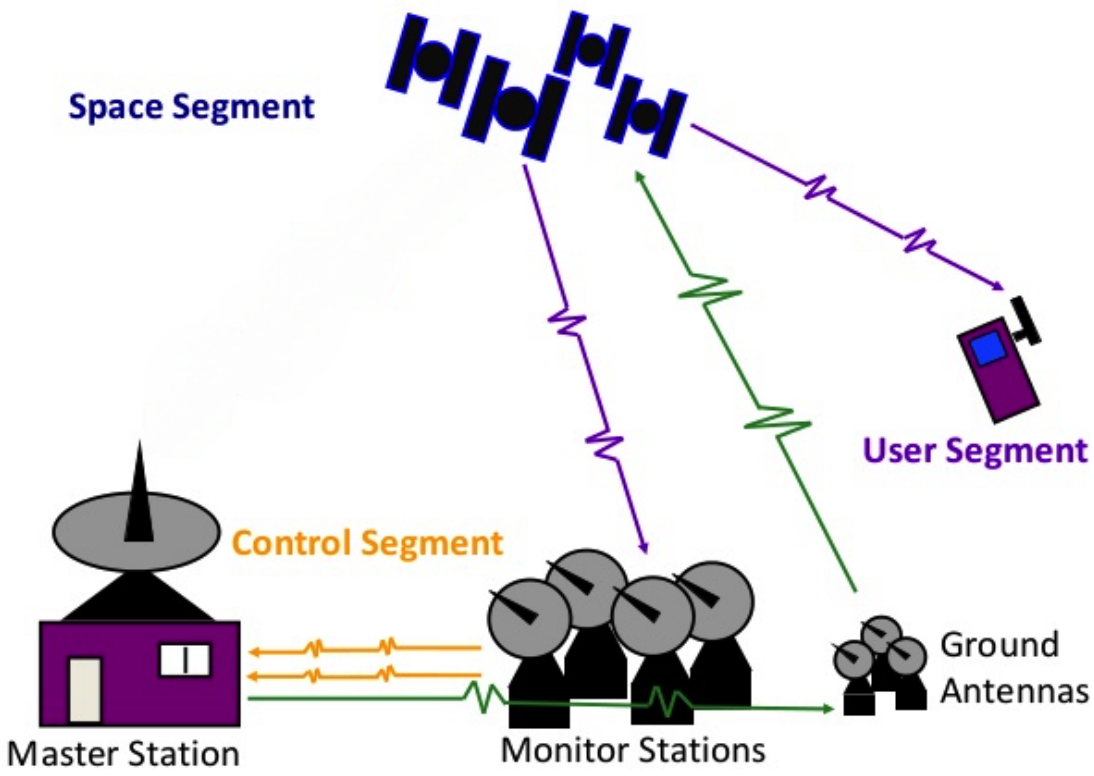


Figure 2.1: Three segments of a typical satellite system.

The monitor stations measure the pseudorange [1] of visible satellites every 6 seconds, correct them with ionospheric and meteorological data, smooth the measurement to generate data with a time interval of 15 seconds, perform smoothing again to generate data with a 15 minutes' time interval, and finally send the data to the master control station. The master control station is responsible for collecting and tracking data from each monitor station and calculating the satellite orbit and clock parameters using a Kalman estimator [52]. The results are transmitted to ground antennas and then to the satellite. Under the control of the master control station, the clock error, satellite

[1] pseudorange: the pseudo distance between a satellite and a navigation satellite receiver, which can be obtained by multiplying the speed of light by the time the signal has taken from the satellite to the receiver. There are accuracy errors in the time measured, thus the term "pseudorange" is used instead of "range"[133].

ephemeris, navigation data, etc., are calculated and then transmitted to the corresponding satellite, and at the same time, the information is verified. The satellites transmit data associated with their current states to the users. The users need to use the location information provided by the satellites for positioning during navigation. According to [84], in general, at least four satellites are required to determine the user's position.

The flow of information between segments of the system is shown in Figure 2.2. In this process, the accuracy of the information that each segment provides is critical and depends directly on the accuracy errors in the time measured. From the monitor station to the master control station, from the master control station to the ground antenna, from the ground antenna to the satellite, and from the satellite to the user, the entire process is implemented by information transmission. Errors may exist in the process of information transmission, and if these errors are passed on all the way to the user, the position provided by the navigation system is incorrect.

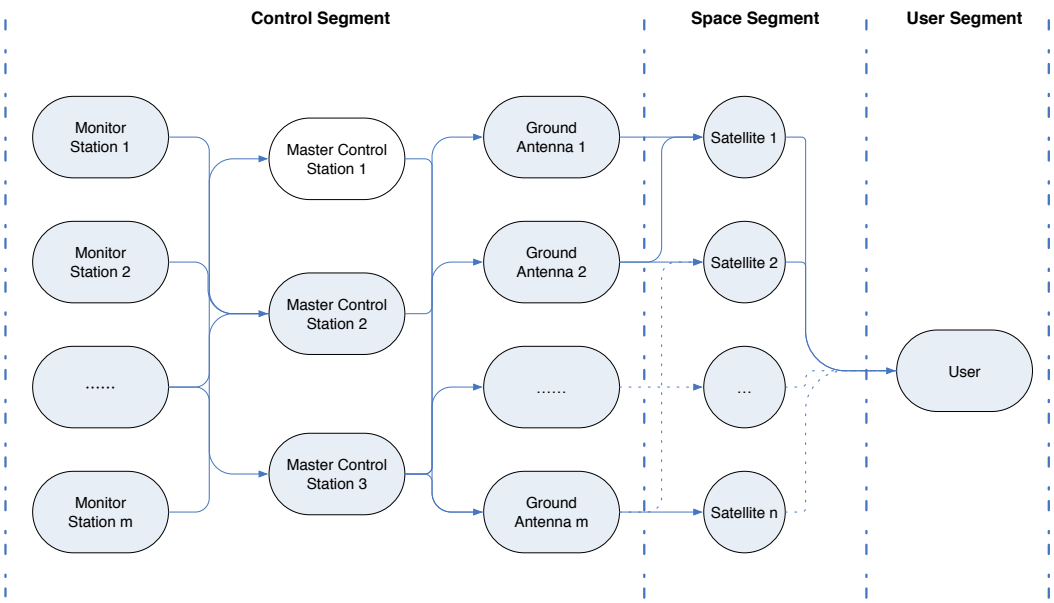


Figure 2.2: Flow of information between segments.

2.2.1 Space Segment

The space segment of a standard satellite based system is composed of a constellation of satellites, as shown in Figure 2.3 (e.g., GPS comprises 24 satellites). The arrangement of the satellite constellation can guarantee that four or more satellites can be

observed at the same time from any location on earth at any time and ensure that the propagation of the satellite signal will not be disturbed by the environment. Therefore, a GNSS-based navigation system should be a global and around-the-clock navigation system that continuously provides uninterrupted real-time navigation.

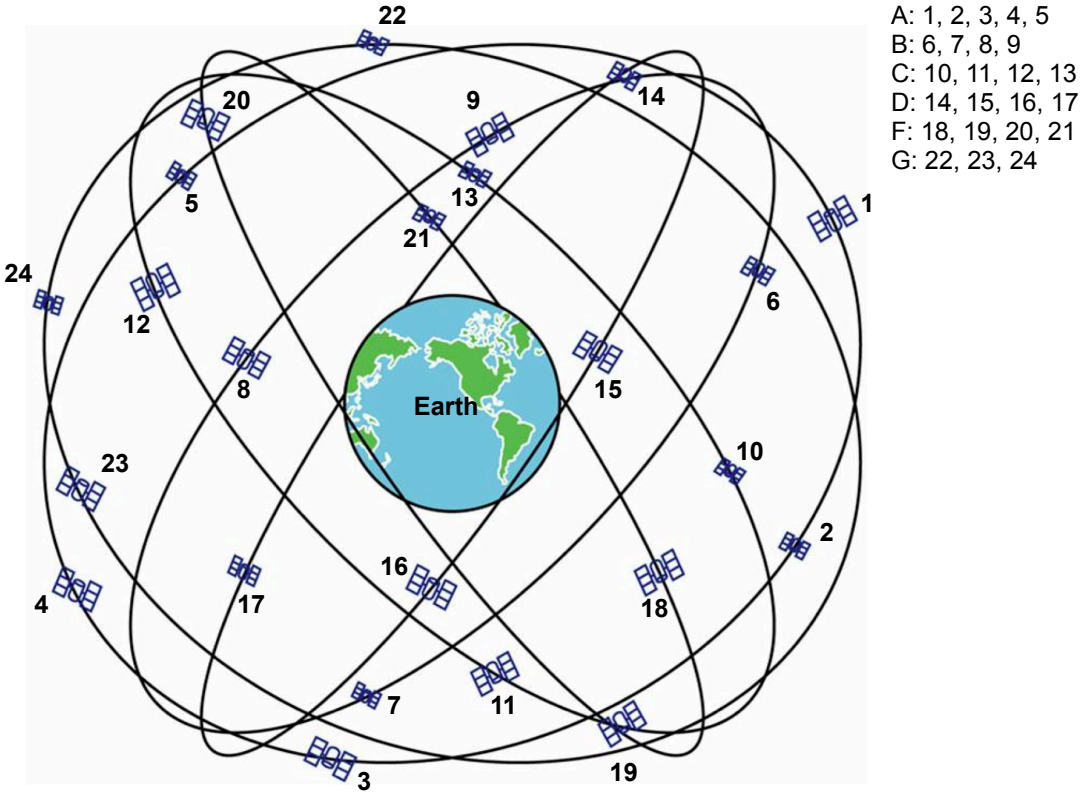


Figure 2.3: Space segment: a constellation of 24 satellites in GPS.

The role of the space segment is described by the functional specification as follows: (1) continuously transmit navigation signals to users worldwide with carrier radio waves at a specific band, including the pseudo-range for satellite navigation, the exact time for users, the distance measurement and a navigation message that contains the spatial location and current health status of the satellite; (2) receive messages, ephemeris and other related information from the ground antenna via a specific band when the satellite passes above a ground antenna; (3) transmit and receive satellite commands from the master control station through the ground antenna, including activating redundant satellites, correcting on-orbit satellite errors and adjusting the spatial attitude of satellites at the appropriate time; (4) adjust the direction of the pair of solar panels on both sides of the navigation satellite according to the position of the sun to ensure a stable power supply.

The satellite transmits signals at a specific frequency, providing a high-standard timing service to users worldwide. This function is implemented primarily by the atomic clock onboard the satellite.

2.2.2 Control Segment

The control segment consists of three parts: monitor stations, master control stations (MCS) and ground antennas. This segment is implemented in the form of a number of detecting and measuring systems distributed across various locations in the world. The control segment continuously monitors and tracks the satellites. The role of the control segment components includes: (1) monitor the satellite's operation and orbit states; (2) track and compute the orbit parameters of satellites and send them to the satellites to be retransmitted to the users via a navigation message; (3) synchronise the clocks of satellites; (4) perform scheduling for satellites when necessary. In the control segment of the satellite constellation, the monitor stations and ground antennas are unmanned, and the master control station is staffed. The unattended intelligent schema and information transmission among advanced communication networks is implemented through coordination between computers and atomic clocks.

2.2.2.1 Master Control Station

The master control station acts as the brain of the control segment. It is responsible for processing the information received by the receiving station and feeding the correct information to the ground antenna. The main functions of the master control station are summarised as follows.

First, it provides satellites with the accurate time. The atomic clocks onboard the satellites and the atomic clock of the monitor stations are synchronised by the master station, or, if a time difference between them is observed, the master control station will include it in the navigation data and send it to the ground antenna. Second, it corrects the environmental parameters of the atmosphere, satellite ephemeris [2], satellite clock correction, etc., by a calculation based on the data of satellites in the constellation system that are monitored by various monitor stations and then transmits this information to the ground antennas to update the satellites. Third, it controls and sends commands to the satellites, and coordinates backup satellites to replace failed satellites

[2] ephemeris: the positions of artificial satellites in the space at a given time or times

once satellites under normal operation fail to receive and transmit. Finally, it controls satellites that deviate from their orbit and returns them to their planned orbit.

2.2.2.2 Monitor Stations

Monitor stations are centres that measure and examine data under the control of the master control station. These stations consist of computers, high-precision atomic clocks, receivers, and some environmental data detection sensors. The receivers continuously monitor the status of the satellites, measure the on-orbit state of the satellites and collect environmental data to ensure the standards required for mission accuracy.

The environmental sensors acquire data on the local environment, and the atomic clock ensures an accurate time. The computer processes these measurements and stores the data, then transmits the data to the master control station for calculating the satellite orbit. The control segment of the GPS is made up of five Monitor Stations located at Hawaii, Kwajalein, Ascension Island, Diego Garcia, and Colorado Springs.

2.2.2.3 Ground Antennas

The functions of ground antennas under the control of the master control station are to receive clock errors, ephemeris, mission data and other commands, which are all calculated and determined by the master control station; transmit this information to the system; and check the correctness of the transmitted information. There are also three Ground Antennas located at Ascension Island, Diego Garcia, and Kwajalein. Ground antennas consist primarily of a computer, a transmitter at a specific band, and a transmitting antenna.

2.2.3 User Segment

The user segment of the satellite system consists of multiple parts, generally including system software, a system receiver, a computer and meteorological equipment. The receiver hardware mainly consists of several parts: the controller, host, power supply and antennae.

The main roles of the receiver are as follows: (1) Receive signals from satellites in the system, capture the signals selected by the satellite's cut-off angle, check the operating orbits of the satellites and calculate the user's position information and the measurements of the satellites; (2) Perform data conversion, expansion and calculation on the signals received from the system to calculate the transmission time of the

signal from the satellite to the receiver antennae; (3) Calculate the user's position, velocity and time (PVT) based on the data transmitted from the satellite; (4) Present the processed data to the user through the data display.

2.2.4 Environment Segment as an Influencing Factor

In Europe, the arrival of satellite systems based on Galileo, which is interoperable with the existing GPS, will provide localisation support to aircrafts, trains, and ships, etc. However, the reception of satellite signals can be sensitive, and depends on the environment around the antenna, in the air, and on the ground. Thus it may degrade the typical reliability and availability of the underlying satellite systems.

The environment segments for aircrafts, trains, and ships are very different. Regarding trains, the main environmental influence factor is mountain or tunnel. Whereas, the environmental influence factors for aircrafts are unknown external disturbances such as solar radiation and magnetic effect. Although environment segment is treated differently for various user segments, we will the same way that is to assign different probabilities for various levels of reception of satellite signal when we model the communication between space segment and user segment. In the following example, we use trains to illustrate the approach to assigning probabilities.

Satellite systems have the potential to greatly enhance the efficiency and performance of rail transport operations. A number of satellites are available as a function of the position of a train on its journey and the environment (e.g., obstacles, tunnels, mountains) that surround it. The immediate environment have different effect on the quality of the signal strength of the satellite measurements made along the different routes of trains. For instance, the satellite signal cannot be tracked reliably below the predefined signal levels.

Since the reception of a satellite signal is directly related to the environment around the antenna, we can distinguish 3 levels as illustrated in Figure 2.4. The best level of reception occurs when the satellite is directly visible. We call this level as "visible"(or Line of Sight (LOS) state). When the optical link between the receiver (user segment) and the satellite (space segment) is not available, the signal can be received by an alternate path if it can reflect on a surrounding obstacle, otherwise, it will be blocked and not received. As the train runs, since the environment is not uniform along the track, the satellite reception states will vary function of time from "visible", "alternate path", and "blocked".

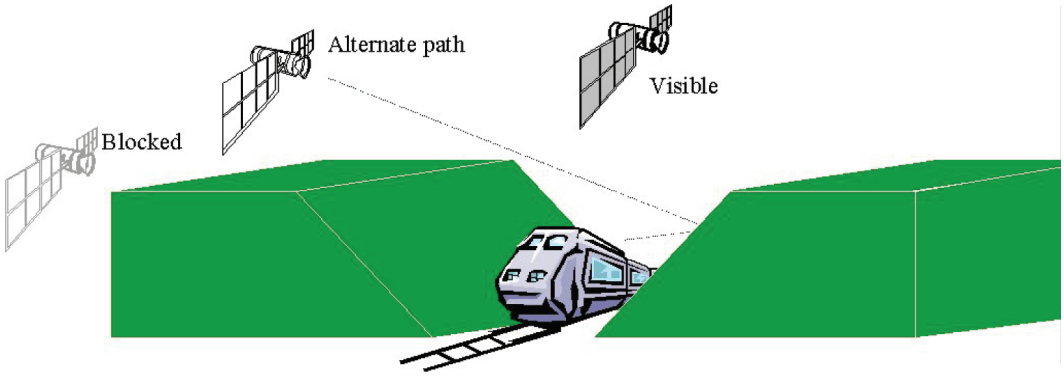


Figure 2.4: Satellite reception in a particular environment.

2.3 RAMS Requirements for Mission Critical Space Industry

Recently, there has been an increasing amount of research in the application of Reliability, Availability, Maintainability, and Safety (RAMS) techniques to space and satellite industry. The key focus of RAMS research is to look at interactions between reliability, availability, maintainability, and safety.

RAMS studies provide system designers and satellite infrastructure engineers with the metrics to assess the impact of design and maintenance decisions over particular periods of time. As a result, modelling and analysis are able to help to determine whether changes in the resources allocated to achieve particular reliability or availability requirements will have significant knock-on effects for other operating parameters.

2.3.1 Failure Characteristics

As shown in Figure 2.5, satellite operation constitutes a cycle of information transmission between components. A satellite transmits a signal to the monitor station, the monitor station transmits the signal to the master control station, the master control station then transmits the signal to the ground antenna, and finally, the ground antenna uploads the information to the satellite.

Due to various factors, the monitor station, master control station, or ground antenna may fail during the operation of the system, resulting in a temporary interruption of the operation, which will resume after repair. Similarly, the satellite can also fail during operation and not transmit signals properly. In this thesis, failures due to satel-

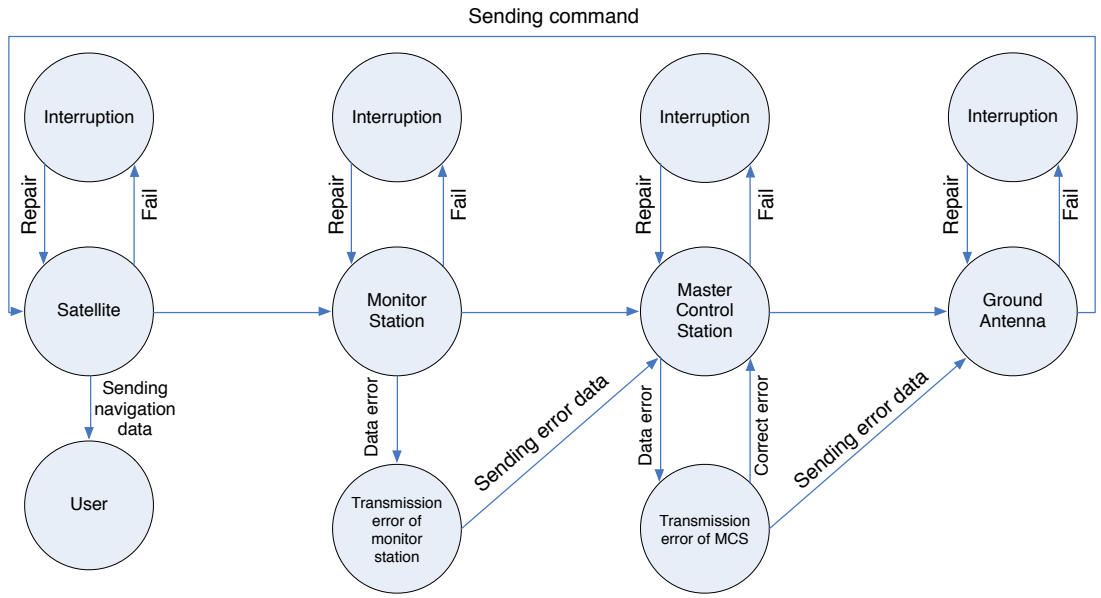


Figure 2.5: Signal transmission in satellite systems.

lite ageing are considered in the satellite analysis. Once failure occurs, new satellites must be launched to replace the failed satellites.

During signal transmission from the monitor station to the master control station and from the master control station to the ground antenna, abnormal signal transmission may occur, resulting in errors in information and corresponding anomalies in the subsequent update information for the satellites. This can affect the mission if the situation is severe. If anomalies occur in signal transmission, the master control station can correct the signal after a certain period of time.

Based on a preliminary investigation, it is assumed in our analysis that the information exchange among the satellites, monitor station and ground antenna does not itself generate information anomalies, but its reliability is a direct consequence of the reliability of the satellites and ground antenna. It is additionally assumed that information anomalies can only occur in the signal transmission between the master control station and the monitor station.

2.3.2 RAMS Requirements

Over the past decade, Europe's space and satellite industry is being challenged by increased competition and by the adjustments of EU rules and regulations to improve interoperability. The current situation forces developers to reduce costs, improve re-

liability, regularity, and maintain or improve operational safety. These developments brought an urgent need on the formal specification of RAMS requirements.

A host of standards and regulatory documents provide the background for the RAMS requirements in mission-critical applications. In Europe, they depend on the support and approval of the European Committee for Electrotechnical Standardisation (CENELEC). In the United States, American National Standards Institute (ANSI) brought into correspondence the European CENELEC's RAMS requirements by their own International Electrotechnical Commission (IEC) standards. Therefore, EN50126 has a counter-part IEC 62278 dealing with RAMS requirements while the specification requirements in EN50128 are parallelised in IEC 622279.

For instance, the standard EN 50126 defines RAMS in terms of long-term system characteristics as follows:

- Reliability: the probability that a system can perform a required function under given conditions for a given time interval.
- Availability: the ability of a system to be in a state to perform a required function under given conditions at a given instant of time or over a given time interval, assuming that the required external resources are provided.
- Maintainability: the probability that a given active maintenance action, for a system under given conditions of use, can be carried out within a stated time interval when the maintenance is performed under stated conditions and using stated procedures and resources.
- Safety: a system is said to have an adequate safety if does not cause harm to people, the environment, or any other assets during its life cycle - during normal use and also for foreseeable misuse.

In this thesis, we quantify these attributes and calculate them using probabilities. This integrated perspective is particularly important for satellite operations when, for example, reliability can typically be assured by maintaining the satellites but only at the cost of availability. Alternatively, pressure to increase availability through reduced maintenance cycles may reduce reliability and also undermine safety.

2.3.3 Relationship between RAMS Properties

Although the accuracy of satellite positioning in the aviation environment is in general sufficient, it is its availability that limits the system dependability and overall perfor-

mance. Availability properties relate to the reliability and maintainability of satellite systems. Traditionally, it is the probability that the system is operating at a satisfactory level and can be committed at the start of a navigation mission when the mission is called for at an unknown and random point in time.

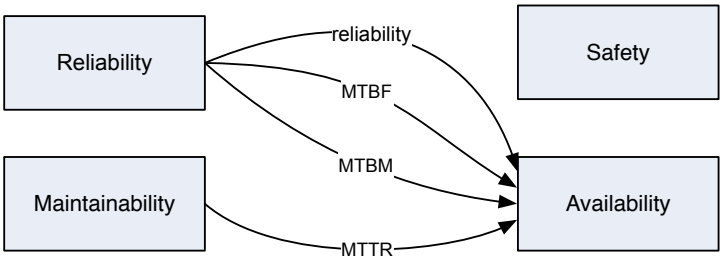


Figure 2.6: Overview of RAMS analysis.

The relationship between availability, reliability, maintainability, and safety is depicted in Figure 2.6. In general, availability heavily depends on reliability and maintainability. For repairable satellites, the term Mean Time between Failure (MTBF) has been commonly used. MTBF denotes the average length of time from one failure to the next, and also includes the repair time. The Mean Time To Repair (MTTR), is the average length of time taken to repair a failed satellite. System designers should aim to allow for a high MTTR value and still achieve the reliability requirements.

Availability is a mathematical function of MTBF and MTTR. We assume that there is negligible delay before the repair commences on a failed satellite begins. The availability factor can be computed using the following formula, and clearly a satellite system that can offer high availability is more desirable than one that offers lower availability.

$$Availability = \frac{MTBF}{MTBF + MTTR} \tag{2.1}$$

Availability can range from 0% (never available) to 100% (always available). Satellite systems that can offer high availability are more desirable than ones that offer lower availability. As a result, the availability requirement for a system is that it should provide a sufficient guarantee that the system is in an operable state at any time. Informally, availability properties can be classified as the following five types:

1. How often do failures occur that require corrective maintenance?

2. How often is preventative maintenance performed?

3. How quickly can indicated failures be isolated and repaired?

4. How quickly can preventive maintenance tasks be performed?

5. How much do logistics support delays contribute to down time?

In this thesis we propose a modified and unambiguous concept for satellite system availability properties associated with the underlying specification. The current approach involves prediction of the "mean" availability over the system lifetime, assuming that the system is in a steady state. This approach is not suited to the specification of GNSS based positioning systems, where the objective is to guarantee what can be obtained from the system during short periods of time that are meaningful to users, and that this short term availability will be maintained during the lifetime of the system. This requires a modification of the availability concept, as it is currently understood.

2.4 Towards Verification of Satellite and Space Systems

In this section, we give some background on related work on verification of satellite and space systems.

2.4.1 Traditional Verification Techniques

Predictive verification of satellite availability is useful for numerous applications such as airplane navigation missions and in-car navigation systems. Simulation is nowadays widely used, and there have been a number of notable attempts to use this technique to address the problem of design exploration for satellite systems [31, 139]. In [139], software simulation based on a Markov model of a GPS constellation of 24 satellites is used to obtain availability estimates of GNSS in Taiwan. The primary input data for the availability model is the MTBF and failure rate of the GPS satellites.

In [132], an automated method for predicting the number of satellites available to a GPS receiver, at any point on the Earth's surface at any time, is described. Availability analysis between a GPS receiver and each potentially visible GPS satellite is performed using a number of different surface models and satellite orbit calculations. In [72], the availability of an Navigation-Communication Satellite System (NCSS) is studied to examine the feasibility of using an NCSS constellation in Australia. A performance model was proposed in [70] to evaluate the availability of satellite systems over geographic grid averaging areas over a given period of time. The corresponding

cost model and performance model are designed in such a way as to minimise cost and maximise performance of the systems.

In [41], a method for determining the availability of three different GPS services (positioning, supplemental navigation, and sole means navigation) is described for both two-dimensional and three-dimensional applications. A 21-satellite and a 24-satellite constellation are considered. In the companion paper [40], state probability analyses of 21- and 24-satellite constellations based on a Markov chain model are discussed. Availability characteristics for GPS and GPS augmented by geostationary satellites (GSs) are compared in [119]. Availability is determined for users in the contiguous zone[3] in the United States, based on the planned operational GPS constellation and various GS deployments.

2.4.2 Formal Verification Techniques

Formal methods have significantly impacted aerospace systems engineering, and have successfully been applied to the verification and validation of many aspects of satellite and space systems.

A small aircraft transportation system consisting of a number of approaching aircraft has been formally modelled and its safety properties analysed in [112] using the interactive theorem prover PVS. PVS [113] was also used to verify desired properties in system models of Ariane 5 rocket where the cost of failure is high. In the PICGAL project [35], ground-based software for launch vehicles similar to Ariane 5 were analysed. In a NASA report [127], formal methods and their application to critical systems are explained to stakeholders from the aerospace domain. In [71] Markov models are used to evaluate the cost of availability of coverage of satellite constellation. The potential role of formal methods in the analysis of software failures in space missions is discussed in [66], .

Similarly, in [20], the use of or the potential to use verification techniques, such as static analysis, model checking, and compositional verification, can be used to gain trust in space-based systems is discussed. Model checking has proved to be a suitable formal technique for exposing errors in satellites, mainly due to classical concurrency errors. Unforeseen interleaving between processes may cause undesired events to occur. In [54], the SPIN model checker [63] was used to formally analyse a multi-threaded plan execution module, which is a component of NASA's artificial

[3]Contiguous zone: the zone of the ocean extending 3-12 nautical miles (nms) from the US coastline.

intelligence-based spacecraft control system as a part of the Deep Space 1 mission. Five previously undiscovered errors were identified in the spacecraft controller, in one case representing a major design flaw.

The model checker Murψ [36] has been used in [130] to model the Entry, Descent and Landing phase of the Mars Polar Lander. It was then used to search for sequences of states that led to the violation of a Murψ invariant. This stated that the thrust of the pulse-width modulation, which controls the thrust of the descent engines, should always be above a certain altitude. In [48] the model checker NuSMV [25] is used to model and verify the implementation of a mission and safety critical embedded satellite software control system. The control system is responsible for maintaining the altitude of the satellite and for performing fault detection, isolation, and recovery decisions, at a detailed level.

Furthermore, model checking is used in [24] to simulate satellite operational procedures, and it exploits a simulator of a satellite as a black box in order to formally verify operational procedures. In [24, 49], exhaustive search of all possible simulation scenarios is performed, using the simulator as a model. Thus the verification is automated and complete. Moreover, the approach of system level formal verification to exploit a simulator in order to carry out formal verification has been further developed in [95, 96] and applied to biological contexts. Finally, all these approaches use the explicit model checker CMurphi [118].

Chapter 3

Probabilistic Model Checking

Some materials in this chapter are included in the papers [93, 117, 91, 114]. We provide a background on basic concepts of probabilistic model checking, which is the main approach used for the verification of satellite systems for critical applications throughout this thesis.

3.1 Formal Methods

Formal methods can be regarded as different concepts to different people. The term "formal methods" originates in formal logic, but now is mainly used in the area of Computer Science to refer to a wide range of mathematically based approaches for the analysis of computerised systems. In this section, we investigate the role of formal methods both in the system development life cycle and in the critical aerospace engineering context.

3.1.1 Formal Methods in System Life Cycle

From the perspective of a system life cycle, one relevant and widely recognised definition to this thesis of formal methods is that a formal method is a collection of syntax and formal semantics associated with automatic verification tools. They can be used to precisely model the requirements of a system design, and to prove properties of the underlying model, and to prove correctness of an eventual implementation with respect to that model [60]. Thus, formal methods can loosely be defined as either specification, theorem proving, or model checking. In this thesis, we concentrate on the latter.

As shown in Figure 3.1, in order to apply model checking to a level of system

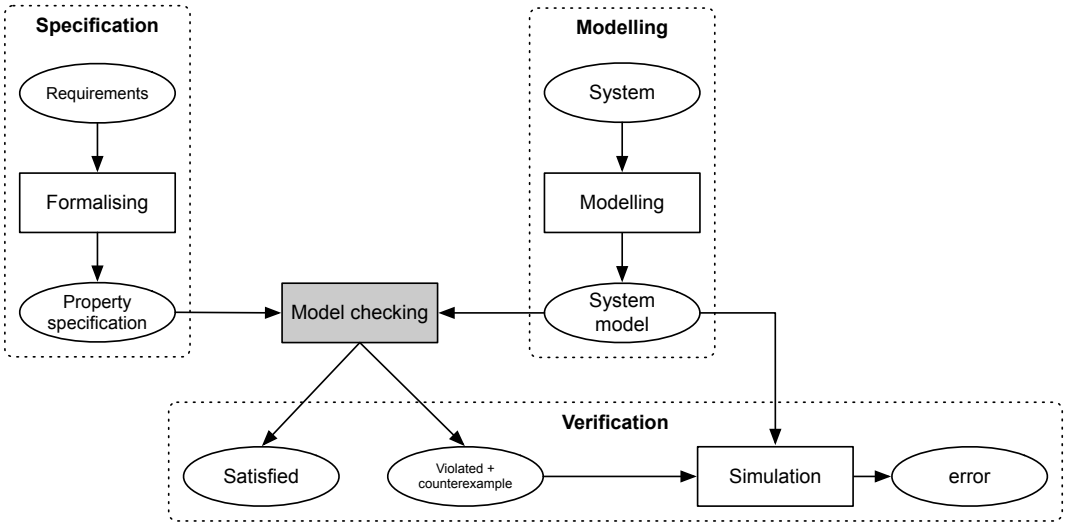


Figure 3.1: Three parts of model checking (adapted from [12]).

design and development, the tasks to be performed are logically divided into three phases:

- Modelling: in the first phase, the system design or mission scenario is formally represented in a formalism that is acceptable to automatic verification tools, known as model checkers. For some cases, abstraction may be used to remove or hide less important or unnecessary details of the system design.

- Specification: the next phase is to state and express the important and necessary properties that the system must satisfy. Again, it is essential to use a formalism that is acceptable to the model checker. In this phase, a formal temporal logic is usually used for hardware and software systems.

- Verification: model checkers then are used to verify the validity of the specification against the proposed model exhaustively. The most common mode of operation is for these tools to verify the state space of a system for satisfaction of the specifications and to generate verification results. For negative results produced by the model checker, the counter example is available to be analysed, and then the problem can be traced back either to the model or specification due to such incorrectness. After that, suitable modification actions can be subsequently taken.

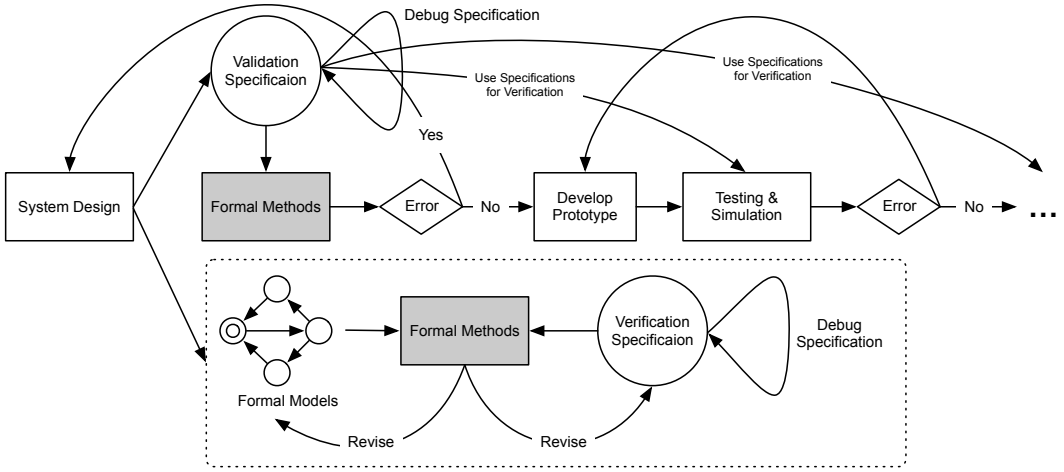


Figure 3.2: The role of formal methods in system development lifecycle in aerospace engineering.

3.1.2 Formal Methods for Aerospace Engineering

There are three key application domains to which formal methods have been applied extensively and successfully. The first domain is safety critical systems where failure may endanger human life, such as fly-by-wire control systems and railway signalling systems. The second domain is security critical systems, involving security protocols and applications where failure means unauthorised access to sensitive information, such as medical records. The third is standardisation and certification: where systems are designed to meet specific, internationally recognised, standards or regulations. In this case, it is important that the standards can be interpreted uniformly.

In addition, as underlying formal verification techniques have matured, and a number of automatic tools have become available as well, formal methods have had an increasingly significant impact on aerospace engineering. Formal methods are mainly used in the design time of system development for various aerospace and space missions. They also have been recommended in the DO-178B[1] standard for certification, and successfully applied in many aerospace contexts. Figure 3.2 illustrates a typical system development process in aerospace engineering that integrates formal methods as a fundamental approach to the early design of satellite and space systems.

Model checking is a formal method for verifying finite state systems, such as analysing sequential circuit designs and communication protocols. During the de-

[1]DO-178B: a guideline that deals with the safety of safety-critical applications used in certain airborne systems.

velopment of traditional satellite and space systems, model checking methods have been used effectively to validate onboard software [19, 48]. Model checking based design tools enable the early validation of software requirements. Examples of model checkers for this purpose includes NASA's JavaPathfinder [55], Cadence SMV [62], Carnegie Mellon's NuSMV [26], and Bell Lab's SPIN [63], etc. Model checkers can be used to detect dead code, integer overflow, division by zero, and violations of system design properties. Modelling approaches vary to some extent in the way that systems are represented and the associated model checking can be categorised accordingly as: referred to as: explicit state, bounded, symbolic, approximate, or probabilistic. In general, most model checkers analyse and operate on discretised models of systems.

Model checking and probabilistic model checking have been extensively applied to aerospace systems in the area of safety-critical applications. In [50], authors propose an approach to quantitatively assessing safety properties using the PRISM probabilisitic model checker, and illustrate the approach with a representative system design from the airborne industry. A verification technique is developed in [33] for analysing the decision-making component in agent-based hybrid systems. In [59], authors developed a model checker called ETMCC, which has been used in several non-trivial case studies to support the automated verification of performability properties. The paper [3] provides a review of the usage of formal methods and model checking for reliability, availability, maintainability, safety analysis for safety-critical applications in the area of aerospace and transportation. The paper [127] describes the advantages, disadvantages, and challenges in applying formal methods and model checking to safety-critical applications. In [144], the authors performed formal specification, verification, and model validation of a coordination protocol for an automated air traffic control system for safety-critical applications using NuSMV and Cadence SMV.

Explicit state model checking refers to the way that the state space is represented when checking properties. Here, states are represented explicitly and not abstracted or merged. Conversely, in symbolic state model checking the state space is stored in a reduced form, which is represented symbolically as a binary decision diagram (BDD). Probabilistic model checking is an effective approach to analysing systems exhibiting stochastic behaviours, and is a highly suitable approach for use in the design, analysis, and implementation of satellite systems and their missions. Within the theory of computation, model checking is essentially rooted in automata and temporal logic. In the next section we introduce important preliminaries of model checking, with a focus on labelled transition systems and temporal logic.

3.2 Model Checking

Modelling and verification of computerised systems has been one of the major research topics of the last 25 years of computer science. Nowadays, the challenges are becoming increasingly harder, mainly because the scenarios we are dealing with often exhibit a complexity that is intractable. Computerised systems are now less structured due to their underlying mobility and concurrency. Another crucial element to consider is that classical computerised systems are closely integrated within satellite and space systems. This means that applications dependant on satellites are everywhere and the results of their computations affect lives and missions. Therefore, the guarantee that systems are working correctly, reliably, and dependably is becoming vital.

Accordingly, three dimensions must to be tackled. First, model checking methods: various methods and tools have been applied to verify reliability, availability, maintainability and safety by employing tools based on calculi for mobile processes [58, 9, 2]. However, these methods lack generality and are not applicable to complex and uncertain environments. Thus, it is necessary to revisit current model checking methods and to use appropriate model checking techniques which can be applied to a wide range of uncertain characteristics. Second, dynamic analysis methods: some static methods and tools have already been successful for protocol analysis in the static case. But for real-time and time-sensitive applications of satellites, dynamic analysis methods are also crucial, because they are closer to the real world and user behaviours. Therefore, using a representation language for handling mobility is also important. Finally, the validation of methods presented in this thesis has to be evaluated within a realistic application domain of satellite systems.

3.2.1 Labelled Transition Systems

Transition systems are often used in Computer Science as models to describe the behaviour of systems. They are directed graphs where nodes represent states, and edges model transitions. A labelled transition system (LTS) comprises some number of states, with arcs between them labelled by actions of the system. We consider labelled transition systems with action names for state changes and atomic propositions for the states. Action names are used for describing communication mechanisms between processes. Labels consist of atomic propositions (representing properties of interest) that are true at each state. A formal definition of LTSs is as follows:

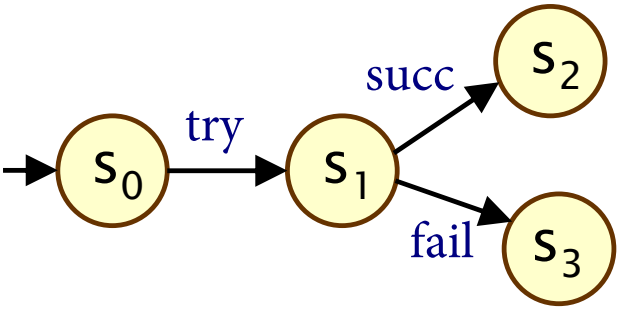


Figure 3.3: An example of a labelled transition system.

Definition 1. *Formally, a labelled transition system (LTS) C is a tuple (S,I,Act,$\rightarrow$,L) where:*

- $S = \{s_0, s_1, ..., s_n\}$ *is a finite set of states.*
- $I \subseteq S$ *is a set of initial states.*
- *Act is a set of actions.*
- $\rightarrow \subseteq S \times Act \times S$ *is the transition relation.*

Traditionally, $s_i \xrightarrow{\alpha} s_{i+1}$ is instead denoted as $(s, \alpha, s') \in \rightarrow$. The fundamental behaviour of an LTS can be described as follows. The LTS starts in some initial state $s_0 \in I$ and evolves based on the transition relation $\rightarrow$. If $s_i (0 \leq i \leq n)$ is the current state, a transition $s_i \xrightarrow{\alpha} s_{i+1}$ that originates from s_i is then enabled in a nondeterministic manner. Thus, the action α is performed, the LTS evolves from state s_i to the state s_{i+1}. This process is repeated in state s_{i+1} until a state is reached at which that there is no outgoing transitions. Thus, an LTS has an important property that if a state has more than one outgoing transition, the next transition is selected according to nondeterminism. This causes the successor of the chosen state to be unknown until the choice is made. As a result, no statement can be made about the likelihood with which a certain transition is chosen.

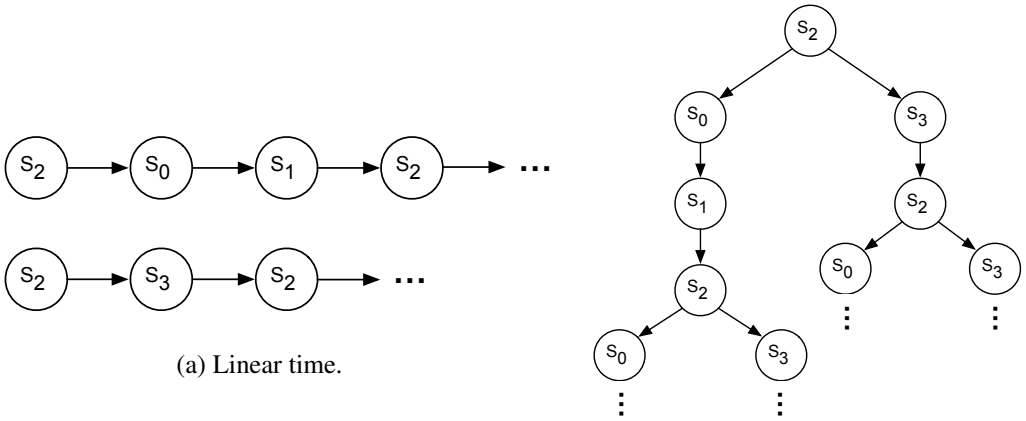
LTSs are suitable for modelling discrete state systems that evolve through actions. Generally, we distinguish certain states: a start state and perhaps one or more final states. Figure 3.3 gives a labelled transition system that models a simple communication protocol from the perspective of a sender. The state space is $S = \{s_0, s_1, s_2, s_3\}$.

The set of initial states is $I = \{s_0\}$. The sender's action "try" denotes the transmission of a message along the communication channel, while the actions "succ" and "fail" denote the successful delivery of the message and failure of transmission, respectively. Then we have: $Act = \{try, succ, fail\}$.

3.2.2 Temporal Logic

Temporal logic can be used to describe the behaviours of systems over time. Pnueli [120] suggested a unified approach to program verification of sequential and parallel systems. The main proof method suggested involved temporal reasoning about systems. Further, the use of temporal logic for reasoning about properties of reactive systems was proposed [55]. This logic uses a set of atomic properties, Boolean connectives, and four temporal operators. The temporal operators normally mean future operators: invariant, eventually, next, and until.

There are two views of time: linear time and branching time, in which time is perceived as an ordered set and a tree respectively (see Figure 3.4). Logics which allow us to reason upon these notions of time are Linear-time Temporal Logic (LTL) [64] and Computational Tree Logic (CTL) [43].



(a) Linear time.

(b) Branching time.

Figure 3.4: Comparison between two views in temporal logics.

3.2.3 Linear-Time Temporal Logic (LTL)

Linear-time temporal logic (LTL) is a rich specification language that can express many desired properties, such as reachability, safety, invariance. Linear temporal logic for-

mulas consist of temporal operators, Boolean operators, and atomic propositions connected in any sensible way. In this subsection, we formally describe linear temporal logic by giving its syntax and semantics. Specifications based on LTL formulas are built from a finite set of atomic propositions ($\mathcal{AP}$). The LTL formulas are recursively defined according to the following grammer:

- **true**, **false**, and a are LTL formulas for all $a \in \mathcal{AP}$;
- if ϕ is a LTL formula, then $\neg\phi$ is a LTL formula;
- if ϕ_1 and ϕ_2 are LTL formulas, then $\varphi_1 \wedge \phi_2$ and $\phi_1 \vee \varphi_2$ are LTL formulas;
- if φ_1 and φ_2 are LTL formulas, then $\bigcirc\phi_1$ and $\phi_1 \mathbf{U} \phi_2$ are LTL formulas.

The operator $\bigcirc$ is called "next", and the formula $\bigcirc\phi$ is true if ϕ will be true in the next time step. The other operator $\mathbf{U}$ is called "until", and the formula $\phi_1 \mathbf{U} \phi_2$ is true if ϕ_1 is true until ϕ_2 is true (ϕ_1 must hold until ϕ_2 holds). The additional classical temporal operators "exists" ($\Diamond$) and "always" ($\Box$) can be derived as follows:

- $\Diamond\phi = \mathbf{true}\mathbf{U}\phi$;
- $\Box\phi = \neg\Diamond\neg\phi$.

LTL formulas are interpreted over the observed sequences of the transition system from the initial state. The set of observations O is defined by $\mathcal{AP} \cup \{\tau\}$ for some element $\tau \notin \mathcal{AP}$. The special symbol τ is used to represent observations not corresponding to any atomic proposition. This allows us to use LTL formulas to specify sequences of observations. LTL formulas are interpreted over sequences of observations $\gamma : \mathbb{N} \to O$ as follows:

- $\gamma(i) \models p$ iff $p = \gamma(i)$;
- $\gamma(i) \models \phi_1 \wedge \phi_2$ iff $\gamma(i) \models \varphi_1$ and $\gamma(i) \models \varphi_2$;
- $\gamma(i) \models \phi_1 \vee \phi_2$ iff $\gamma(i) \models \varphi_1$ or $\gamma(i) \models \phi_2$;
- $\gamma(i) \models \bigcirc\phi_1$ iff $\gamma(i+1) \models \phi_1$;
- $\gamma(i) \models \phi_1 \mathbf{U} \phi_2$ iff $\exists j \geq i$ such that for all k, $0 \leq k < j$, $\gamma(k) \models \phi_1$ and $\gamma(j) \models \phi_2$.

where $p \in \mathcal{AP}$, ϕ_1 and ϕ_2 are LTL formulas, and $i \in \mathbb{N}$. We say that a sequence γ satisfies formula ϕ iff $\gamma(0) \models \phi$.

3.3 Probabilistic Models

Table 3.1: Classification of probabilistic models.

	Discrete Time	**Continuous Time**
Fully Probabilistic	DTMCs	CTMCs
Nondeterministic	Probabilistic Automata (PAs) MDPs	PTAs

Probabilistic model checking has been applied to numerous systems based on different types of probabilistic models. Some of these models incorporate a discrete notion of time, while others incorporate a continuous notion of time. Further classification can be added based on how to specify uncertainty of systems that involve aspects of control or concurrency. We distinguish probabilistic models that exhibit nondeterministic behaviours with those that exhibit completely stochastic behaviours. Popular and useful models with mature verification algorithms and tools based on these classification can be shown in Table 3.1. In this section, we define 4 types of probabilistic models relevant to our work, namely Discrete-Time Markov Chains (DTMCs), Continuous-Time Markov Chains (CTMCs), Markov Decision Processes (MDPs), and Probabilistic Timed Automata (PTAs).

3.3.1 (Discrete-Time and Continuous-Time) Markov Chains

3.3.1.1 Discrete-Time Markov Chains

The basics of discrete-time Markov chains (DTMCs) can be shown considering the example in Figure 3.5, showing a DTMC that models a very simple probabilistic communication protocol. After one step, the sender starts trying to transmit a message through the communication channel. Then, a random choice is made among 3 probabilities: (1) with probability 0.01, the sender waits a time step due to that the channel is not ready; (2) with probability 0.01, the process of sending message fails due to the unreliable channel or transmission collision, and the process restarts; (3) with probability 0.98, the sender successfully transmits the message and finally stops.

Definition 2. *Let $\mathcal{AP}$ be a fixed, finite set of atomic propositions. Formally, a discrete-time Markov chain (DTMC) $\mathcal{D}$ is a tuple (S,s_{init},P,L) where:*

- $S = \{s_1, s_2, ..., s_n\}$ *is a finite set of states.*

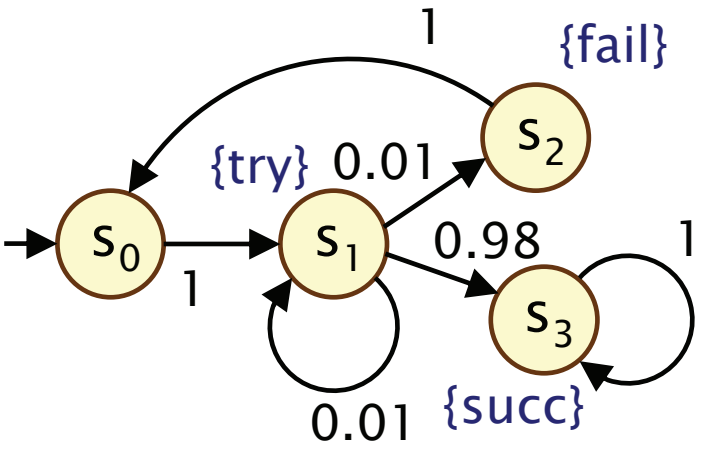


Figure 3.5: An example of an DTMC.

- $s_{init} \in S$ *is the initial state.*
- $P: S \times S \to [0,1]$ *is the transition probability matrix where* $\Sigma_{s' \in S} P(s,s') = 1$ *for all* $s \in S$
- $L: S \to 2^{AP}$ *is a labelling function which assigns to each state* $s_i \in S$ *the set* $L(s_i)$ *of atomic propositions* $a \in \mathcal{AP}$ *that are valid in* s_i.

In a DTMC, the underlying labelled transition system is augmented with probabilities, and the discrete set of states of a DTMC representing possible configuration of the system being modelled. Transitions between these states occur in discrete steps due to the discrete nature of time in this paragigm.

3.3.1.2 Continuous-Time Markov Chains

For continuous-time Markov chains (CTMCs), the time variable associated with the system evolution is continuous, and state changes can occur at any arbitrary time. Figure 3.7 illustrates a simple CTMC for a machine availability which includes only two states: working and failed. The state of the machine is checked continuously. It can be captured that the average time to failure of a working machine is 50 days, and the average time for repair of a failed machine is 2 days.

When the machine is in state working, it is vulnerable to failures, which transition the machine to failed state. This transition rate is modelled as R(working,failed)= $1/50 = 0.02$. When the machine is in state failed, it can be repaired, and the machine transitions back to the working state at the rate R(failed,working)= $1/2 = 0.5$. In

$D = (S, s_{init}, \mathbf{P}, L)$

$S = \{s_0, s_1, s_2, s_3\}$

$s_{init} = s_0$

$AP = \{try, fail, succ\}$

$L(s_0) = \emptyset$,
$L(s_1) = \{try\}$,
$L(s_2) = \{fail\}$,
$L(s_3) = \{succ\}$

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3.6: Formal representation of the DTMC in Figure 3.5.

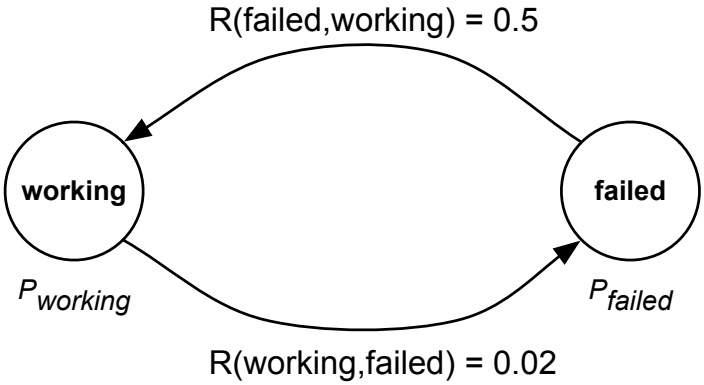


Figure 3.7: An example of an CTMC.

addition, if we assume $P_{working}$ is the probability of the machine being in the working state, while P_{failed} denotes the probability of the machine being in the failed state. Thus, the sum of both probabilities is obviously 1: $P_{working} + P_{failed} = 1$.

In Chapters 4 and 5, the approach adopted is event based because of the fault and failure events that can be sensed and monitored in the satellite systems. Rates are assigned to events and our underlying semantics is continuous-time Markov chains (CTMCs): the state space is discrete but time is continuous. In this section, we briefly review the basic concept of CTMCs.

Definition 3. *Let $\mathcal{AP}$ be a fixed, finite set of atomic propositions. Formally, a continuous-*

time Markov chain (CTMC) $\mathcal{C}$ is a tuple (S,s_{init},R,L) where:

- $S = \{s_1, s_2, ..., s_n\}$ *is a finite set of states.*
- $s_{init} \in S$ *is the initial state.*
- $R: S \times S \to \mathbb{R}_{\geq 0}$ *is the transition rate matrix.*
- $L: S \to 2^{\mathcal{AP}}$ *is a labelling function which assigns to each state $s_i \in S$ the set $L(s_i)$ of atomic propositions $a \in \mathcal{AP}$ that are valid in s_i.*

Intuitively, $R(s_i, s_j) > 0$ if and only if there is a transition from state s_i to state s_j. Furthermore, $1 - e^{-R(s_i,s_j)\cdot t}$ specifies the probability of moving from s_i to s_j within t time units, which is an exponential distribution with rate $R(s_i, s_j)$. If $R(s_i, s_j) > 0$ for more than one state s_j, a competition between the transitions originating in s_i exists, known as the race condition.

The probability to move from a non-absorbing state s_i to a particular state s_j within t time units, i.e., the transition $s_i \to s_j$ wins the race, is given by:

$$P(s_i, s_j, t) = \frac{R(s_i, s_j)}{E(s_i)} \cdot (1 - e^{-E(S_i)\cdot t}), \tag{3.1}$$

where $E(s_i) = \sum_{s_j \in S} R(s_i, s_j)$ denotes the total rate at which any transition outgoing from state s_i is taken. More precisely, $E(s_i)$ specifies that the probability of taking a transition outgoing from the state s_i within t time units is $1 - e^{-E(S_i)\cdot t}$, since the minimum of two exponentially distributed random variables is an exponentially distributed random variable with rate the sum of their rates. Consequently, the probability of moving from a non-absorbing state s_i to s_j by a single transition, denoted $P(s_i, s_j)$, is determined by the probability that the delay of going from s_i to s_j finishes before the delays of other outgoing edges from s_i; formally, $P(s_i, s_j) = R(s_i, s_j)/E(s)$. For an absorbing state s_i, the total rate is $E(s_i)$. In that case, we have $P(s_i, s_j) = 0$ for any state s_j.

3.3.2 Markov Decision Processes

In this subsection, we briefly review the basic concepts of Markov Decision Processes (MDPs). For the purpose of this thesis, we consider the number of states are finite, together with a description of the possible transitions among the states. In MDPs, the choice as to which transition to take from a particular state s is made according to two stages: the first stage comprises a nondeterministic choice among a number of

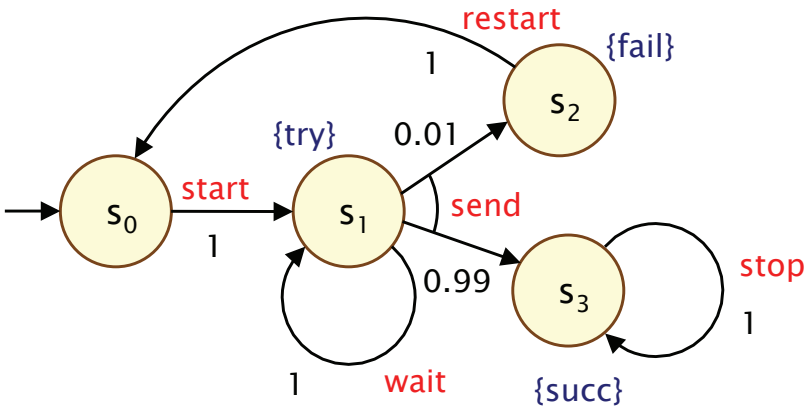


Figure 3.8: An example of an MDP.

actions available in the state s; whereas the second stage involves a probabilistic choice between the possible target states of the transition. During the the second stage, the probability distribution used to choose the next state of the MDP is determined by the choice of action made in the first stage.

The integration of nondeterministic and probabilistic choice in MDPs can be useful in numerous application scenarios. In the context of the formal modelling of systems, "nondeterministic choice can be used to represent such factors as interleaving between concurrent processes, unknown implementation details, and (automatic or manual) abstraction" [14]. In Chapter 6, we consider a more high-level formalism for the description of satellite based aviation systems which are based on MDPs (more precisely, MDPs provide the underlying semantics of the high-level formalism).

The concepts are illustrated based on the example MDP in Figures 3.8 and 3.9. We model a very simple probabilistic communication protocol using an MDP in Figure 3.8. Our MDP comprises a set of states: $\{s_0, s_1, s_2, s_3\}$. There is a single process. After one step, the process starts trying to send a message. Then, a nondeterministic choice is made between: (a) waiting because the channel is not ready; (b) sending the message. If the latter choice is made, with probability 0.99 the message is sent successfully and stops, and with probability 0.01, message sending fails, and the process restarts.

Definition 4. *Let Act be a set of actions, and $\mathcal{AP}$ a fixed, finite set of atomic propositions. Formally, a Markov decision process (MDP) $\mathcal{M}$ is a tuple $(S, s_{init}, Steps, \mathcal{L})$ where:*

- $S = \{s_1, s_2, ..., s_n\}$ *is a finite set of states;*

- $s_{init} \in S$ *is the initial state;*
- $Steps: S \to 2^{Act \times Dist(S)}$ *is the transition probability function where Act is a set of actions and* $Dist(s)$ *is the set of discrete probability distributions over the set S;*
- $\mathcal{L}: S \to 2^{\mathcal{AP}}$ *is a labelling function with atomic propositions.*

$M = (S, s_{init}, \mathbf{Steps}, L)$

$S = \{s_0, s_1, s_2, s_3\}$
$s_{init} = s_0$

$AP = \{init, try, fail, succ\}$
$L(s_0) = \{init\}$,
$L(s_1) = \{try\}$,
$L(s_2) = \{heads\}$,
$L(s_3) = \{tails\}$

$\mathbf{Steps}(s_0) = \{ (start, s_1 \mapsto 1) \}$
$\mathbf{Steps}(s_1) = \{ (wait, s_1 \mapsto 1), (send, [s_2 \mapsto 0.01, s_3 \mapsto 0.99]) \}$

$\mathbf{Steps}(s_3) = \{ (stop, s_3 \mapsto 1) \}$

Figure 3.9: Formal representation of the MDP in Figure 3.8.

Unlike DTMCs, deterministic probability cannot be measured in an MDP, but a countably infinite number of probabilities is measured in the MDP. In general, each probability corresponds to a different way of resolving the nondeterministic choice during the execution of the system [14].

An execution path of an MDP requires both non-deterministic and probabilistic transitions to be resolved. Non-deterministic choices are made by an adversary where the decision is determined by the choices made in all previous runs. An adversary $\mathcal{A}$ can be defined formally as a function that maps every finite path π_{fin} in an MDP onto a distribution $\mathcal{A}(\pi_{fin}) \in Steps(last(\pi_{fin}))$ where $last(\pi_{fin})$ is the final state of the finite path. For convenience the subset of Path(s) which corresponds to adversary $\mathcal{A}$ is denoted $Path^A(s)$.

3.3.3 Probabilistic Timed Automata

Full details about probabilistic timed automata (PTAs) can be found in [79, 111]. We outline the important aspects in this section. PTAs allow us to use the real-valued

clocks of timed automata [5], which is one of the most popular and powerful formalisms for the formal verification of real-time systems, together with the discrete probabilistic choice of MDPs. PTAs have real-valued clocks and, like MDPs [39] and therefore allow us to model systems with a range of different characteristics (nondeterminism, probability, and real time).

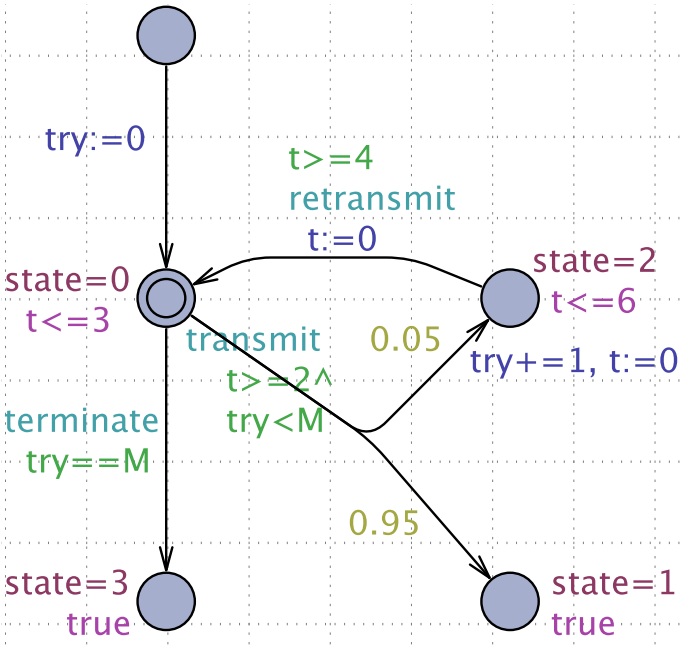


Figure 3.10: An example of a PTA.

In this subsection, we illustrate a number of basic PTA concepts using the example in Fig. 3.10. The Figure illustrates a probabilistic timed automaton, with clock t and integer variable try, modelling a simple probabilistic communication protocol. In the protocol, a sender repeatedly attempts to transmit a message over an unreliable channel. The probability that the sender's transmission fails due to that the channel is unreliable is 0.05, and the sender successfully transmit the message to the receiver with probability 0.95. If message data from the sender is lost, the sender suspends its activity, and there is a delay (of between 4 and 6 time units) before the sender tries to resend its message (up to $M-1$ times).

The control states of the automaton model, "$state = 0$", "$state = 1$", "$state = 2$", "$state = 3$", have the meaning of "transmit", "wait", "quit", and "finish" respectively. They are depicted as the nodes (circles) of the underlying graph, and the available transmissions between these control states are indicated as the edges (with arrow) of the graph. In the initial state, "$state = 0$", shown as the extra border, a communication

is being initialised by the sender along the transmission channel. After between 2 and 3 time units, the sender attempts to send the message, and with probability 0.95 message is sent correctly, meantime with probability 0.05 message is lost. In "$state = 2$", when 4 to 6 time units have elapsed from whenever the message is lost, the sender tries to re-transmit the message.

Definition 5. *A probabilistic timed automaton PTA $\mathcal{P}$ is a tuple of the form $(L, l_0, \Sigma, inv, prob)$ where:*

- $L = \{l_0, l_1, l_2, ..., l_n\}$ *is a finite set of locations;*
- $l_0 \in L$ *is the initial location;*
- $\chi = \{x, y, z, ...\}$ *is a finite set of clocks;*
- $\Sigma = \{a, b, c, ...\}$ *is a finite set of events, of which $\Sigma_u \subseteq \Sigma$ are declared as being urgent;*
- *the function $inv : L \to CC(\chi)$ is the invariant condition;*
- *the finite set $prob \subseteq L \times CC(\chi) \times \Sigma \times Dist(2^{\chi} \times L)$ is the probabilistic edge relation.*

Note that clocks are real-valued. The values of the clocks synchronise and increase together over time. Transitions and states may have guards and invariants over clock variables and other variables which indicate when transitions can occur and how long can be spent in a state. In our example, the transition between states $state = 0$ and $state = 1$ (or $state = 2$) has the clock guard $t \geq 2$. The state $state = 0$ and $state = 3$ have the invariant t $<$ 3 and t $<$ 6 respectively.

The semantics of PTAs are formally defined as an infinite state MDP. As clocks are real-valued the MDP will have an infinite state-space (both in terms of set of states, and the set of transitions). Since model-checking algorithms are designed to work on finite state spaces, the analysis of PTAs requires some form of abstraction, to a finite state representation. PTAs have been used to verify a variety of protocols, e.g. the CSMA/CD back-off protocol [38], the FireWire root contention protocol [81], and the IPv4 Zeroconf protocol [79].

3.4 Probabilistic Model Checking

All probabilistic models in previous sections include a notation of probability by labelling transition with likelihood that it will occur. Thus, it allows us to make quantitative description of the system, in addition to the qualitative description made by conventional model checking approaches. In recent years, many researchers are working on applying probabilistic model checking for design and implementation of satellite and space systems [42, 44, 145, 137]. The focus of such work is on both theoretical foundations and industrial applications of probabilistic model checking.

Probabilistic model checking is a variant of model checking. Model checking not only requires formal model of a system, but also requires a formal specification of the system requirements that the system model should guarantee. These system requirements are typically represented in temporal logic, and refer to sequences of system events, such as "a target state is reached within 50 execution steps" or "a response always follows a request".

A model checking algorithm is executed to establish automatically whether the system model satisfies the property. Model checking has been extended to the case of probabilistic systems. In this case, properties "are specified in a probabilistic extensions of classical temporal logic, and refer to the maximum or minimum probability of temporal logic properties over execution sequences " [14]. For instance, a system may be regarded as correct if the maximum probability of reaching a target state within 50 steps is greater than 0.8. Model checking for probabilistic models depends on the combination of different techniques for computing optimal rewards and costs together with theory from the field of qualitative model checking.

Unlike the traditional approaches described above, the probabilistic analysis performed in our thesis is applicable to address desired properties based on temporal logic, returning quantities computed by model checking rather than a true or false answer. Moreover, probabilistic model checking is based on Markov models and timed automata, and considers continuos time in a natural way, this has overcome the problem such as the lack of time factors in the traditional system model, or discrete time when time is included in the system model. Moreover, since probabilistic model checking involves the exhaustive exploration of all possible paths of the model, it can also supply significant information about the system. For example, the optimal path and cost, average rewards, the maximal or minimal probability, time bound, and so on.

3.4.1 Selection of Model Checking Tool: PRISM

Numerous scientists and engineers have used model checking to help design and analyse finite state concurrent systems. In particular, model checking is able to verify hardware and software systems such as complex sequential circuit designs and communication protocols [30]. Further, a number of automated verification tools based on model checking, which are known as model checkers, have been developed for many years. There are many examples of earlier model checkers, such as SPIN, SMV/NuSMV/NuSMV 2, KRONOS.

SPIN [62] is a qualitative model checker for linear temporal logic (LTL), and was developed for the verification of communication protocols and software systems in the 1980s. Users need to translate the system of interest into the PROcess Meta LAnguage (PROMELA), which is the underlying modelling language of SPIN. The SPIN is popular and successful, but it is difficult to model unbounded data variables.

SMV [100, 29] is another popular qualitative model checker for computation tree logic (CTL) and uses binary decision diagrams (BDD), which is a mainstream verification technique. NuSMV [26] and NuSMV 2 [25] are symbolic model checkers originated from the re-engineering, re-implementation and extension of SMV. Both model checkers apply the bounded model checking methods for LTL in addition to the BDD for the CTL. Similar to SPIN, SMV and NuSMV/NuSMV 2 need to translate the system into their own input modelling languages. In general, such translations require abstraction and refinement of the underlying systems, due to which false counterexamples may exist.

KRONOS [143, 18] is a real-time model checker based on timed automata. Properties of interest in KRONOS are specified using timed computation tree logic (TCTL). The timed automata can be given in a textual form which is constructed based on a system consisting of a number of components. It then generates the product automaton based on the synchronisation of automata .

There are also several popular quantitative model checking tools for probabilistic verification of systems, each of which offers and supports one or a number of models. Support and availability are generally the key factors for us to consider selecting the most appropriate verification tool. Thus, we choose the state of the art probabilistic symbolic model checker PRISM since not only it supports various different probabilistic models, but also it is currently available for free and still actively developed. We then summarise some of most well-known and successful probabilisitic model check-

ers.

UPPAAL [15, 83] is a real-time model checker for for automatic verification of safety and bounded liveness properties of real-time systems modelled as networks of timed automata. UPPAAL was developed in collaboration between Uppsala University in Sweden and Aalborg University in Denmark. It is free for noncommercial applications in academia only, and for commercial applications a commercial license is required. An extension of UPPAAL has been made for considering cost optimal reachability, which is known as UPPAAL CORA. Recently, another extension of UPPAAL (known as UPPAAL SMC) is adding probability to timed automata using statistical model checking techniques, which is a computationally efficient verification approach based on selective system sampling [57]. UPPAAL SMC has been used extensively in various research areas for industrial applications, such as systems biology and software engineering.

The Markov Reward Model Checker (MRMC) [68, 69] is a probabilistic model checker with a particular focus on systems modelled as CTMCs and DTMCs. It supports model checking of PCTL and CSL, and their reward extensions. MRMC also supports to calculate both time bounded and reward bounded reachability probabilities. MRMC is also freely available.

PRISM is a probabilistic model checker, which can be used to model, analyse and verify systems that exhibit random or probabilistic behaviour based on different types of probabilistic models, such as DTMCs, CTMCs, MDPs, and PAs. PRISM recently starts to support for analysing probabilistic real-time systems, using PTAs. PRISM was developed and is maintained by researchers from University of Oxford, University of Birmingham, and University of Glasgow in UK.

In Fig. 3.11, we show the graphical user interface for PRISM in which the results of a model checking experiment are displayed. In this example a reachability property is being verified and the graph shows how the maximum expected number of steps varies with a defined property defined as *Reliability*. It is free and open source, and it runs on several operating systems such as Mac OS X, Windows, and Linux. It has also been used to analyse systems from many different industrial applications, such as communication and multimedia protocols, security protocols, dynamic power management, biological systems, and autonomous systems.

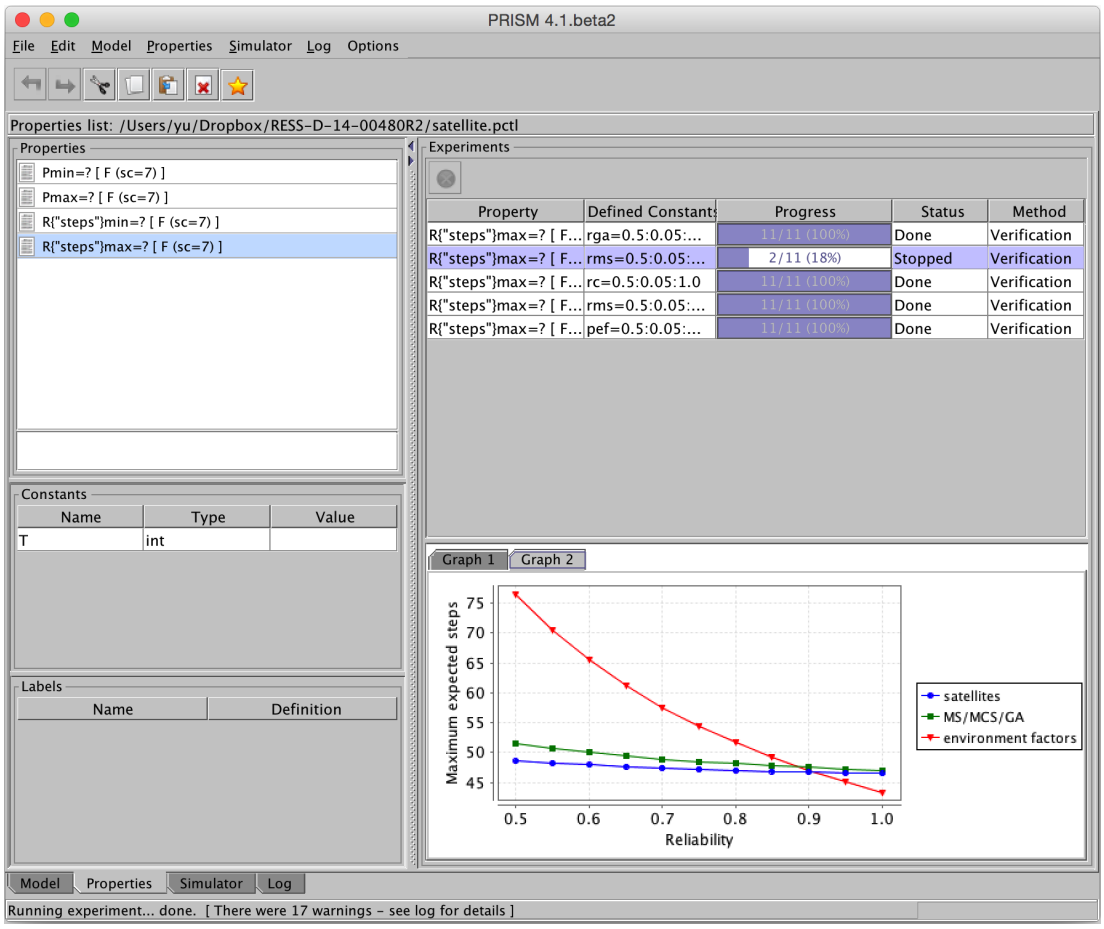


Figure 3.11: A screenshot of the PRISM probabilisitic model checker.

3.4.2 Reactive Modules

PRISM supports the analysis of several types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), Markov decision processes (MDPs), probabilistic automata (PAs), and also probabilistic timed automata (PTAs), with optional extensions of costs and rewards [77]. Moreover, PRISM allows us to verify properties specified in the temporal logics PCTL for DTMCs and MDPs and CSL for CTMCs and PTAs. Models are described using the PRISM language, a simple, state based language.

Markov models to be verified in PRISM are specified using the PRISM modelling language which is based on the Reactive Modules formalism [6]. A fundamental component of this language is a *module*. A system is constructed as the parallel composition of a number of modules. A module is specified as:

module *name* ... **endmodule**

A module definition consists of two parts: one containing variable declarations, and the other *commands*. At any time, the *state* of a model is determined by the current value of all of the variables of all of the components (modules). A variable declaration has the form:

$$x \ : \ [0..2] \ \textbf{init} \ 0;$$

In this example, variable x is declared, with range $[0..2]$ and initial value 0. The behaviour of each module is specified using commands, comprising a guard and one or more updates of the form:

$$[action] \ guard \ \rightarrow \ rate \ : \ update$$

or,

$$[action] \ guard \ \rightarrow \ rate_1 : update_1 + rate_2 : update_2 + ...$$

The (action) label is optional, and is used to force two or more modules to synchronise. Updates in commands are labelled with positive-valued rates [77] for CTMCs. The + indicates the usual non-deterministic choice. Within a module, multiple transitions can be specified either as several different updates in a command, or as multiple commands with overlapping guards. The following examples:

$$[\,] \ x = 0 \ \rightarrow \ 0.05 : (x' = 0);$$
$$[\,] \ x = 0 \ \rightarrow \ 0.2 : (x' = 1);$$

and

$$[\,] \ x = 0 \rightarrow \ 0.05 : (x' = 0) \ + \ 0.2 : (x' = 1);$$

are equivalent. The guard $x = 0$ indicates that command is only executed when variable x has value 0. The updates $(x' = 0)$ and $(x' = 1)$ and their associated rates indicate that the value of x will remain at 0 with rate 0.05 and change to 1 with rate 0.2. In a CTMC, when multiple possible transitions are available in a state, a race condition occurs [75]. The rate of the synchronised transition is the product of all the individual rates.

3.5 Property Specification

3.5.1 Probabilistic Computation Tree Logic (PCTL)

3.5.1.1 Syntax and Semantics of PCTL

We use Probabilistic Computation Tree Logic (PCTL) to specify various availability properties. PCTL allows us to express properties to do with probabilistic models; i.e.,

models with probabilities associated with transitions. The following definitions are taken from [104] (for a more explicit definition of PCTL see [76]).

Definition 6. *Let $a \in \mathcal{AP}$ be an atomic proposition, $p \in [0,1]$ be a real number, $\bowtie \in \{\leq, <, >, \geq\}$ be a comparison operator, and $I \subseteq \mathbb{R}_{\geq 0}$ be a non-empty interval. The syntax of PCTL formulas over the set of atomic propositions $\mathcal{AP}$ is defined inductively as follows:*

- $\phi := \top \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{P}_{\bowtie p}[\psi]$
- $\psi := \mathbf{X}\phi \mid \phi \mathbf{U}^{\leq k} \phi \mid \phi \mathbf{U} \phi$

In the syntax of PCTL a state s satisfies the probabilistic path formula $\mathbf{P}_{\bowtie p}[\psi]$ if the probability of leaving s via a path satisfying ψ is in the interval I. The path formulae $\mathbf{X}\phi$ is true if ϕ is satisfied in the next state; $\phi_1 \mathbf{U} \phi_2$ is true if ϕ_2 is satisfied at some point in the future and ϕ_1 is true until that point. Finally, $\phi_1 \mathbf{U}^{\leq k} \phi_2$ is true if ϕ_2 is satisfied within k time steps and ϕ_1 is true until that point.

Given an MDP $\mathcal{M}$, the probability of a path from s satisfying path formula ψ under the adversary $\mathcal{A}$ is denoted $\mathbf{P}_s^{\mathcal{A}}(\psi) = Prob_s^{\mathcal{A}}(\{\pi \in Path^{\mathcal{A}}(s) | \mathcal{M}, \pi \vDash \psi\})$.To formally define the semantics of the PCTL formula $\mathbf{P}_{\bowtie p}[\psi]$ a set of adversaries Adv is selected and quantified over. It follows that the satisfaction relation is parameterised by Adv. Therefore, a state s satisfies the formula $\mathbf{P}_{\bowtie p}[\psi]$ if $\mathbf{P}_s^{\mathcal{A}}(\psi) \bowtie p$ for all adversaries $\mathcal{A} \in Adv$. The semantics of PCTL are defined as follows.

Definition 7. *The semantics of PCTL over MDPs is defined as follows:*

- $\mathcal{M}, s \vDash_{Adv} \top$
- $\mathcal{M}, s \vDash_{Adv} a$ *if* a *is true in* $\mathcal{M}, s$
- $\mathcal{M}, s \vDash_{Adv} \neg\phi$ *iff* $\mathcal{M}, s \nvDash_{Adv} \phi$
- $\mathcal{M}, s \vDash_{Adv} \phi_1 \wedge \phi_2$ *iff* $\mathcal{M}, s \vDash_{Adv} \phi_1$ *and* $\mathcal{M}, s \vDash_{Adv} \phi_2$
- $\mathcal{M}, s \vDash \mathbf{P}_{\bowtie p}[\psi]$ *iff* $\mathbf{P}_s^{\mathcal{A}}(\psi) \bowtie p$ *for all* $\mathcal{A} \in Adv$
- $\mathcal{M}, \pi \vDash_{Adv} \mathbf{X}\phi$ *iff* $\mathcal{M}, \pi_1 \vDash_{Adv} \phi$
- $\mathcal{M}, \pi \vDash_{Adv} \phi_1 \mathbf{U}^{\leq k} \phi_2$ *iff for some* $i \leq k$, $\mathcal{M}, \pi_i \vDash_{Adv} \phi_2$ *and* $\mathcal{M}, \pi_j \vDash_{Adv} \phi_1$ *for all* $0 \leq j < i$
- $\mathcal{M}, \pi \vDash_{Adv} \phi_1 \mathbf{U} \phi_2$ *iff for some* $k \geq 0$, $\mathcal{M}, \pi \vDash_{Adv} \phi_1 \mathbf{U}^{\leq k} \phi_2$

3.5.1.2 PRISM Usage of PCTL

We use PCTL to specify various availability properties in our PRISM models. The probabilistic model checker PRISM provides support for automated analysis of a wide range of quantitative properties, such as "what is the probability of a failure causing the satellite to stop working within 12 hours?", "what is the worst-case probability of the satellite on-board system terminating due to an error, over all possible initial configurations?", or "what is the worst-case expected time taken for the satellite signal to be received?".

One of the most important operators in PCTL is the **P** operator, which is used to reason about the probability of an event's occurrence. It is often useful to compute the actual probability that some behaviour of a model is observed. Therefore, PRISM allows a variation of the **P** operator to be used in a query, i.e., $\mathbf{P}_{=?}[pathprop]$, which returns a numerical rather than a Boolean value.

In MDP models, there are two types of branching, nondeterministic, determined by a scheduler, and probabilistic, governed by the probability distribution. In order to interpret this, the properties in PCTL consider under any scheduling of processes, yielding the minimum or maximum over all the possible ways of resolving nondeterminism instead of the exact probability. Simple examples of such properties are "the maximum probability of an error occurring within T time steps": $\mathbf{P}_{max=?}[F \leq T\ ''error'']$; and "what is the worst-case expected time taken for a backup satellite to be launched?": $\mathbf{R}^{time}_{max=?}[\mathbf{F}\ ''launch'']$, where both $''error''$ and $''launch''$ are labels on system states specified in PRISM.

PRISM includes support for the specification and verification of properties based on costs and rewards. This means that PRISM can be used to reason, for example, about properties such as "expected time", "expected number of lost messages" or "expected energy consumption". The basic idea is that probabilistic models developed in PRISM can be augmented with costs (something bad) or rewards (something good): real values associated with certain states or transitions of the model (the costs and rewards are numerically identical). For MDPs, where time proceeds in discrete steps, the time interval is simply an integer upper bound. In our study, we use rewards "steps" to calculate expected time "T". Rewards are associated with models using the rewards...endrewards construct.

```
rewards "time"
true : 1;
```

```
endrewards
```

3.5.2 Continuous Stochastic Logic (CSL)

3.5.2.1 Syntax and Semantics of CSL

We also use Continuous Stochastic Logic (CSL) [11, 13] to specify reliability, availability, and maintainability properties. CSL is inspired by the logic Computation Tree Logic (CTL) [43], and its extensions to discrete-time stochastic systems (PCTL) [53], and continuous-time non-stochastic systems (TCTL) [4]. There are two types of formulae in CSL: state formulae, which are true or false in a specific state, and path formulae, which are true or false along a specific path.

Definition 8. *Let $a \in \mathcal{AP}$ be an atomic proposition, $p \in [0,1]$ be a real number, $\bowtie \in \{\leq, <, >, \geq\}$ be a comparison operator, and $I \subseteq \mathbb{R}_{\geq 0}$ be a non-empty interval. The syntax of CSL formulas over the set of atomic propositions $\mathcal{AP}$ is defined inductively as follows:*

- *true is a state-formula.*
- *Each $a \in \mathcal{AP}$ is a state formula.*
- *If ϕ and ψ are state formulas, then so are $\neg\phi$ and $\phi \wedge \psi$.*
- *If ϕ is state formula, then so is $\mathbf{S}_{\bowtie p}(\phi)$.*
- *If φ is a path formula, then $\mathbf{P}_{\bowtie p}(\varphi)$.*
- *If ϕ and ψ are state formulas, then $\mathbf{X}\phi$, $\phi\mathbf{U}\psi$, and $\phi\mathbf{U}^{\leq k}\psi$ are path formulas.*

Formula $\mathcal{S}_{\bowtie p}(\phi)$ asserts that the steady-state probability for a state satisfying ϕ meets the bound $\bowtie p$. Similarly, formula $\mathbf{P}_{\bowtie p}(\varphi)$ asserts that the probability measure of the paths satisfying φ meets the bound given by $\bowtie p$. The operator $\mathbf{P}_{\bowtie p}(.)$ replaces the usual CTL path quantifiers $\exists$ and $\forall$. Intuitively, $\exists\varphi$ represents that there exists a path for which φ holds and corresponds to $\mathcal{AP}_{>0}(\varphi)$, and $\forall\varphi$ represents that for all paths φ holds and corresponds to $\mathbf{P}_{>=1}(\varphi)$. The temporal operator $\mathbf{X}$ is the timed variant of the standard next operator in CTL; the path formula $\mathbf{X}\phi$ asserts that a transition is made to a ϕ state at some time point $t \in I$. Operator $\mathbf{U}$ is the timed variant of the until operator of CTL; the path formula $\phi\mathbf{U}_I\psi$ asserts that ψ is satisfied at some time instant in the interval I and that at all preceding time instants ϕ holds.

Definition 9. *The semantics of CSL over CTMCs is defined here. The satisfaction relation for the state formulas is defined as follows:*

- $\mathcal{C}, s \vDash true$ *for all* $s \in S$
- $\mathcal{C}, s \vDash a$ *iff* $a \in L(s)$
- $\mathcal{C}, s \vDash \neg\phi$ *iff* $\mathcal{C}, s \nvDash \phi$
- $\mathcal{C}, s \vDash \phi \wedge \psi$ *iff* $\mathcal{C}, s \vDash \phi$ *and* $\mathcal{C}, s \vDash \psi$
- $\mathcal{C}, s \vDash \mathcal{S}_{\bowtie p}(\phi)$ *iff* $\pi_{Sat(\phi)}(s) \in I_{\bowtie p}$, *where* $Sat(\phi) = \{s \in S \mid s \vDash \phi\}$
- $\mathcal{C}, s \vDash \mathcal{P}_{\bowtie p}(\varphi)$ *iff* $Prob(s, \varphi) \in I_{\bowtie p}$

For each state s, the set $\{\sigma \in Path(s) | \sigma \vDash \phi\}$ *is measurable. The satisfaction relation for the path formulas is defined as follows:*

- $\mathcal{C}, \sigma \vDash \mathbf{X}\phi$ *iff* $\mathcal{M}, \sigma_1 \vDash \phi$
- $\mathcal{M}, \sigma \vDash \phi \mathbf{U}^{\leq k} \psi$ *iff for some* $i \leq k$, $\mathcal{M}, \sigma_i \vDash \psi$ *and* $\mathcal{M}, \sigma_j \vDash \phi$ *for all* $0 \leq j < i$
- $\mathcal{M}, \sigma \vDash \phi \mathbf{U} \psi$ *iff for some* $k \geq 0$, $\mathcal{M}, \sigma \vDash \phi \mathbf{U}^{\leq k} \psi$

3.5.2.2 PRISM Usage of CSL

One of the most important operators is the **P** operator, which is used to reason about the probability of an event. This operator was originally proposed for use in the logic PCTL but also features in the other logics supported by PRISM, such as CSL. The **P** operator is applicable to all types of models supported by PRISM.

It is often useful to compute the actual probability that some behaviour of a model is observed. Therefore, PRISM allows a variation of the **P** operator to be used in a query, i.e., $\mathbf{P}_{=?}[pathprop]$, which returns a numerical rather than a Boolean value (i.e., the probability that *pathprop* is true). In our thesis, we are interested in directly specifying reliability, availability, maintainability, and safety properties which evaluate to a numerical value. For example, we might wish to calculate the probability that x is eventually equal to 5, and that x remains less than 5 up until that point. This can be specified as $\mathbf{P}_{=?}[z < 5 \ \mathbf{U} \ z = 5]$, where **U** is the "until" temporal operator.

Another important operator we use is the **R** operator, which specifies a cumulative reward property that associate a reward with each path of a model, but only up to a given time bound. The property $\mathbf{R}_{=?}[C <= t]$ corresponds to the reward cumulated

along a path until t time units have elapsed. For CTMCs, the bound t can evaluate to a real value. Some typical examples of properties using **P** and **R** operators can be found on the Property Specification section of the PRISM website.

3.5.3 Modelling PTAs in PRISM

```
pta
const int M;
module sender
        state : [0..3] init 0;
        try : [0..M] init 0;
        t : clock;
invariant
        (state = 0 =>  t <= 3) & (state = 3 => t <= 6)
endinvariant
        [transmit] state = 0 & t >= 1 & try < M -> 0.95 : (state' = 1) +
                           0.05 : (state' = 2) & (try' = try+1) & (t' = 0);
        [retransmit] state = 3 & t >= 4 -> (state' = 0) & (t' = 0);
        [terminate] state = 0 & try == M -> (state' = 3);
endmodule
rewards ''energy''
        (state=0) : 3.5;
endrewards
```

Figure 3.12: The PRISM code of the PTA in Figure 3.10.

PTAs can be enhanced with rewards and costs, which allow us to track the number of occurrences of a transition. This might correspond to energy used or goals achieved. The corresponding model is known as a priced PTA and it allows us to measure accumulated rewards (reflected in properties expressing the expected value of the rewards/costs). A further extension is linearly priced PTAs, where costs or rewards are accumulated linearly (with respect to the elapsed time) [78]. Finally, parallel composition can be modelled by PTAs as well, which is the same feature of other probabilistic models supported by PRISM. Thus, multiple probabilistic timed automaton are able to work in parallel, and synchronise by taking transitions with the help of labels that can be matched [79].

The PRISM model checker employs a popular modelling language to specify all the probabilistic models, which include DTMCs, CTMCs, MDPs, and PTAs. In particular,

PTAs is a textual language, based on guarded command notation. A clock variable has been added in PTAs that PRISM can then recognise. Clock variables can be placed in guards, which is located on the left-hand side of a command. Like other regular variables in the model, clock variables can also be reset, which results in an update on the right-hand side of the command. There is an additional keyword "invariant" for the purpose of specifying invariants [78]. Figure 3.12 shows a PRISM modelling language description for the example PTA described in Figure 3.10. It further illustrates a case of including a PRISM reward structure, labelled "energy", to generate a priced probabilistic timed automata, which assigns a reward rate of 3.5 when $s = 0$ (during message transmission).

Part II

Modelling, Specification and Verification

Chapter 4

Reliability, Availability, Maintainability Analysis of The Space Segment

This chapter is based on the papers [117, 115]. We present formal models of both a single satellite and a navigation satellite constellation and logical specification of their reliability, availability and maintainability properties respectively. The model checker PRISM has been used to perform automated analysis of these quantitative properties.

4.1 Introduction

The space segment is a core component of satellite systems. Satellites in the space segment are designed for operating on orbit to perform tasks and have lifetimes of 10 years or more. Before formally introducing this technique and discussing the role of formal verification, we briefly review some traditional verification techniques that can be applied to analysing satellite systems, which are prototype testing and model simulation.

Prototype testing is a dynamic verification technique that involves actually running prototype of a system. Correctness is thus verified by running the prototype to traverse a set of execution paths. Based on the results during execution, the actual output of the prototype is compared to the system specification which is usually in the form of documents. Model simulation is similar to prototype testing, but is applied to system models. Models are usually described using hardware description languages. A simulator is used to examine execution paths of the model based on configuration inputs. These inputs can be provided by a user, or by automated approaches such as using a random generator. A mismatch between the simulator's result and the specification of

the system exhibits the incorrect behaviours.

Both verification techniques are limited in that they only allow exploration of a small subset of many possible scenarios. Formal methods is the application of mathematical modelling and reasoning to prove that an implementation coincides with precisely expressed notion of formal specification. In this context, the purpose of formal analysis and verification is to analyse the performance and reliability properties and to verify the correctness of satellite systems in such a way that faults and failures can be identified. Model checking and theorem proving are formal techniques that can be used to detect faults and failures in a formal specification.

Although historically these forms of verification were used to prove correctness of explicit software and hardware designs, these days they are also used for failure analysis. They are generally applied during the design phase, where they are arguably most effective, for verifying correctness and other essential properties such as safety, liveness and fairness. Model checking is an automated analysis technique that requires expert knowledge to use. The user must provide an initial specification of the system itself, as well as logical properties describing its desired behaviour.

One strength of model checking to traditional analysis techniques is that it is not sensitive to the probability that a fault or failure is exposed; this contrasts with prototype testing and model simulation that are aimed at tracing the most probable faults or failures. Moreover, it allows one to precisely analyse results of checking desired properties. Model checking is a general analysis technique that is applicable to a wide range of applications such as embedded systems, software engineering, and hardware design. It also supports analysing properties individually, thus allowing one to focus essential properties first. This enables incomplete formal models to be specified and verified.

The formal model of systems can be defined using a high-level formalism or extracted directly form software using methods such as abstract interpretation. Verification involves checking paths of the state transition graph (or state-space) of the model. Traditionally this involves either exhaustive or on-the-fly search of the state-space in which states are stored explicitly. Another method, which is symbolic model checking [30], involves search of a symbolic representation of the state space, in which groups of states and transitions are explored in a single step.

Quantitative verification is a analysis technique for establishing quantitative properties of a system model. Models analysed through this method are typically variants of Markov chains, annotated with costs and rewards that describe resources and

their usage during execution. Properties are expressed in temporal logic extended with probabilistic and reward operators. Quantitative verification involves a combination of a traversal of the state transition system of the model and numerical computation. In this chapter, we employ the power of probabilistic model checking, which is a leading quantitative verification and analysis technique for a wide variety of systems.

In this chapter, our models are Continuous-Time Markov Chains (CTMCs), and we verify reliability, availability and maintainability properties using probabilistic model checking. RAM analysis of systems has been indispensable in the design phase of space segment in order to achieve minimum failures or to increase mean time between failures ($MTBF$) and thus to plan maintenance strategies, optimise reliability and maximise availability. For example, a typical reliability property that can be checked is what is the probability that a satellite will need to be replaced by a new one in a certain years, while a typical maintainability property could be what is the number (reward) of times that satellites need to be repaired on orbit in 15 years. Finally, a classical availability property that can be checked is what is the availability (reward) of the satellite in a certain of years.

Our chapter is organised as follows. In Section 4.2 we describe the underlying space segment. In Section 4.3 we present our formal specifications of a single satellite and constellation systems and their associated continuous-time Markov chain models respectively. Then, we analyse reliability, availability, and maintainability using the probabilistic model checker PRISM for a single satellite and a satellite constellation in Section 4.4. Finally, in Section 4.5 we conclude and outline directions for future research.

4.2 The Space Segment

As an important application of satellite constellation, navigation satellite systems consist of three major segments: a space segment, a control segment, and a user segment (see Chapter 2). The space segment is made up of a number of satellites, and is responsible for sending the navigation signal on the specific frequency. It is constantly orbiting the surface at an altitude of approximate three earth radii, and emitting signals that travel at approximately the speed of light. The control segment monitors the health and status of the space segment and controls the state of satellites, and updates the data of those satellites. The user segment consists antennas and receiver processors, which receive the signals broadcasted by the satellites and decode them to provide precise

information about the receiver's position and velocity.

In a satellite constellation, fault or failure of more than one satellite will have a direct impact on the stable state of the space geometry and temporal relationship, and the performance of the constellation. So the performance of the constellation is a direct consequence of the state of the constellation. Therefore, the state of the constellation has a close relationship with the state of every satellite in the constellation. So each satellite is critical to the constellation.

In this chapter, our task is to help the end users of satellite systems to evaluate the probability and consequences of faults or failures. The terms of fault and failure in our context can be defined according to [32] as follows:

- Fault: the condition of a satellite that occurs when one of its components or assemblies degrades or exhibits abnormal behaviour;
- Failure: the termination of the ability of a satellite to perform a required function.

Failure is an event as distinguished from fault, which is a state. According to [32], the failure mode is the result by which a failure is observed. After a failure, a satellite in the constellation will be systematically examined in order to identify the failure mode, and to determine the nature of the failure and its basic cause. There are three kinds of failure mode of the satellite: long-term failure (unrecoverable failure), short-term failure, and Operations and Maintenance (O&M) failure. These failure modes are described as follows:

- Long-term failure: this failure is vital to the satellite. If a long-time failure has happened, it usually needs to launch another satellite to replace the failed one. Practically, it indicates that the failed satellite is at the end of its life. It has also been called wear out failure;
- Short-term failure: this refers to a failure that can be repaired in several hours or days. This kind of failure mode means that there is usually no need to launch a new satellite to replace the failed satellite;
- O&M failure: is due to planned maintenance operations, such as navigation satellite orbit manoeuvre and atomic clock switching. We usually do not consider the outage time that is induced by these operations as a failure. It is not expected to impact the continuity of the constellation, but the performance of the constellation.

Whenever a satellite has a fault or fails, there is a chance to repair the satellite on orbit by, for example, rebooting the satellite system, updating the satellite software, or switching the orbit of the satellite. There are three satellite backup modes available for maintenance strategies: on orbit backup, parking orbit backup, and Launch on Need (LON). The on orbit backup mode and parking orbit backup mode are further referred to as space backup. In this chapter, we consider both space backup and LON backup. The main navigation satellite system to be modelled and analysed is depicted in Figure 4.1.

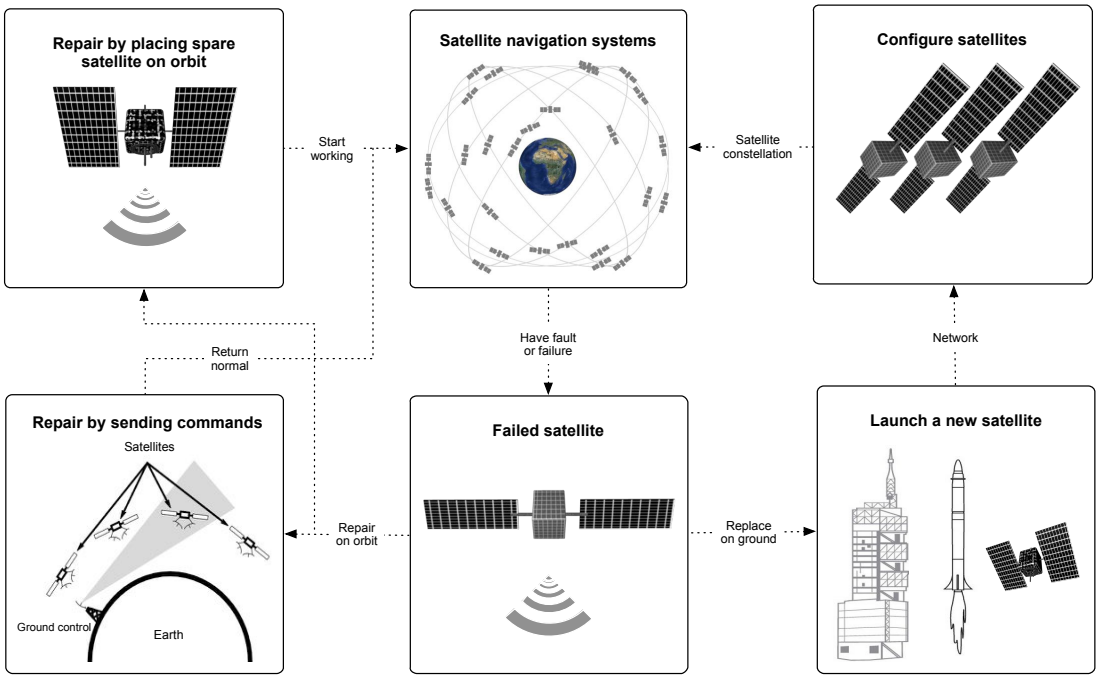


Figure 4.1: An overview of navigation satellite systems.

Satellites deployed at the parking orbit backup mode can also be used to work with on orbit satellites. For LON backup mode, it usually takes several months to replace failed satellite, while for space backup mode it only takes one or two days. Because of the lower mean time to repair ($MTTR$) for the space backup mode, it has been widely applied in most constellation projects. In the GPS project, the redundant satellites are working with on orbit satellites, so failed satellites can be replaced in a short time.

4.3 Formal Specification of Space Segment

In this section, we give an description of the basic formal models of both a single satellite and a constellation of navigation satellites.

4.3.1 A Formal Model of a Single Satellite

The abstract model of a single satellite is illustrated in Figure 4.2, parameters are omitted. We take a CTMC as our underlying PRISM model for our abstract model.

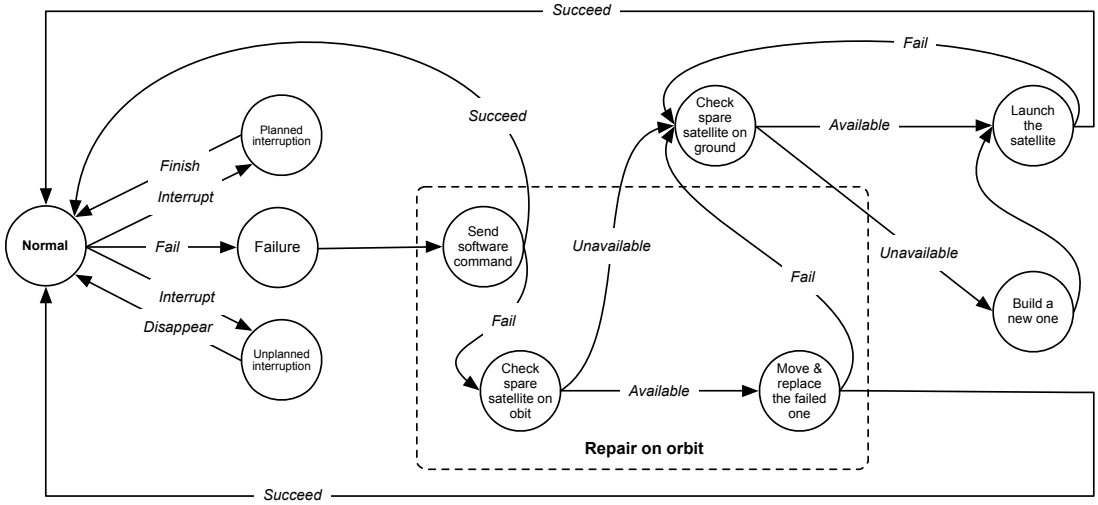


Figure 4.2: A reference model of a single satellite.

We specify our CTMC model with states, a transition rate matrix, and a labelling function. Initially, the satellite runs in the normal state. After a period of execution it could be interrupted by an planned or unplanned interruption. Planned interruptions are normally caused by certain types of Operations and Maintenance (O&M), which could include manoeuvring the station, atomic clock maintenance, software updates, and hardware maintenance. Unplanned interruptions can be caused by solar radiation, the earth's magnetic field cosmic rays, which result in a satellite Single Event Upset (SEU). However, both planned and unplanned interruptions are usually temporary, lasting just several hours. An unplanned interruption usually disappears automatically. The satellite can fail at any time during its lifetime due to End-of-Life (EOL) outage or other vital failures.

When the satellite fails, staff on the ground must decide upon the best approach to repair it. It may be possible that failures can be resolved on orbit by giving spe-

cific software commands to the satellite. Otherwise it might be necessary to move a redundant satellite into position to replace the failed satellite. If no redundant satellite is available then a new satellite must be manufactured and launched. In the worst case the new satellite does not launch successfully due to a known probability of satellite launch failure.

Most of our parameter values correspond to those of the latest United States' GPS system, GPS Block III satellites. The GPS III series is the newest block of GPS satellites. GPS III provides more powerful signals than previous versions in addition to enhanced signal reliability, accuracy, and integrity. The key improvement is the 15 years' design lifespan [108]. Due to privacy and secrecy reasons, NASA does not release all actual data of GPS III that we need in our analysis. Thus, in order to perform the analysis convincingly, we use some generic data of some very similar satellites instead. We believe this this will not result in a loss of generality since all data come from real satellites.

Table 4.1: Parameters used in the model for the single satellite system.

r	$MTBF$ years	$MTTR$ hours	t_u hours	t_p hours	p_b	t_r hours	t_d hours	t_e hours	p_y	t_k hours
0.80	15	24	4320	4320	0.80	24	1440	4320	0.90	24

All parameters used in our CTMC model and properties are specified in Table 4.1, and are described as follows. We use p to express probability and t for time, and the reliability of the satellite is r. If the satellite fails, we say that it moves from a "normal" state to a "failure" state. The mean time to unplanned interruption is t_u, while the mean time to planned interruption is t_p. When the satellite fails, the probability of the failure being resolved on orbit by moving a redundant satellite to replace the failed one is p_b. If on orbit repair is not possible, a new satellite is needed. The time taken to decide to build a new satellite and for one to be manufactured are t_r and t_d respectively. If a new satellite is to be manufactured, the probability of successful launch is p_y. After successful launch, the time taken for the satellite to move to the right position and a normal signal sent from it to be received on the ground is t_k. Our PRISM specification is given as the following:

```
ctmc
const double r; //reliability of satellite
```

```
const double MTBF; //life time of the satellite
const double life = MTBF*12*30*24; //mean time between
      the unplanned interruption
const double tu = 180*24; //mean time between the
      unplaned interruption
const double tp = 180*24; //failure rate of the satellite
const double lan = -life/log(r,2.71828183);
const double d = 1;
const double e = 1; //meant time to repair
const double MTTR = 24; //time to decide to build a new satellite
const double tr = 24; //time to launch if it is available
const double td = 2*30*24; //time to launch if it is unavailable
const double te = 6*30*24;
const double tj = 24;
const double ts = 24;
const double tk = 24;
const double tm = 24;
const double tn = 2;  //time of the unplanned interruption
const double to = 2; //time of the planned interruption
const double a2 = 1; //probability of the failure can be
      eliminated on orbit
const double pb = a2*4;
const double a3 = 1; //probability of the satellite can be
      successfully carried to the orbit
const double py = a3*9;

module satellite
        s : [0..15];
        [a] s = 0 -> 1/tu : (s' = 1); //Normal -> Unplaned
        [b] s = 0 -> 1/tp : (s' = 2); //Normal -> Planned
        [c] s = 0 -> 1/lan : (s' = 3); //Normal -> Failure
        [d1] s = 3 -> 1/a2 : (s' = 8); //Failure -> Software
        [d2] s = 3 -> 1/pb : (s' = 9);
        [d] s = 8 -> 1/d : (s' = 4); //Software -> On orbit
        [e] s = 9 -> 1/e : (s' = 5);
```

```
        [f1] s = 4 -> 1/a2 : (s' = 10); //On orbit -> Move
        [g1] s = 4 -> 1/pb : (s' = 11); //On orbit -> On ground
        [f] s = 10 -> 1/MTTR : (s' = 0); //Move-> Normal
        [g] s = 11 -> 1/tr : (s' = 5); //On orbit -> On ground
        [h] s = 5 -> 1/td : (s' = 6); //On ground -> Launch
        [i] s = 5 -> 1/te : (s' = 7); //On ground -> Build
        [j1] s = 6 -> 1/a3 : (s' = 12); //Launch -> successful
        [k1] s = 6 -> 1/py : (s' =13); //Launch -> unsuccessful
        [j] s = 12 -> 1/tj : (s' = 0); //Launch -> Normal
        [k] s = 13 -> 1/ts : (s' = 5); //Launch-> On ground
        [l1] s = 7 -> 1/a3 : (s' = 14); //Build -> successful
        [m1] s = 7 -> 1/py : (s' = 15); //Build -> unsuccessful
        [l] s = 14 -> 1/tk : (s' = 0); //Build -> Normal
        [m] s = 15 -> 1/tm : (s' = 5); // -> On ground
        [n] s = 1 -> 1/tn : (s' = 0); //Unplaned->Normal
        [o] s = 2 -> 1/to : (s' = 0); //Planned->Normal
endmodule
```

Note that there are 15 states in our PRISM code, while only 10 states are specified in Figure 4.2. This is due to that for the simplicity of the diagram we omit some activities that can contribute to one state. For instance, the state of "Build a new one" actually includes activities such as: making decision to build a new satellite, actually building a new one.

Specifically, tj is the time from launching the satellite to moving it to the right orbit when the satellite has been successfully carried to the orbit if there are no spare satellites on the ground, and ts is the time from launching the satellite to moving it to the right orbit when the satellite has not been carried to the right orbit if there are no spare satellites on the ground, and tk the time from launching the satellite to moving it to the right orbit when the satellite has been successfully carried to the orbit if any spare satellite is available on the ground, and tm is the time from launching the satellite to moving it to the right orbit when the satellite has not been carried to the right orbit if any spare satellite is available on the ground.

4.3.2 A Formal Model of Satellite Constellations

We have modelled a single satellite as a CTMC, by specifying it in PRISM. However, the RAM analysis of a single satellite appears insufficient for larger navigation satellite systems. For a large global navigation system, at least 24 satellites are required. Even for a regional navigation system, at least 4 satellites are required. Our PRISM model for a satellite constellation is thus constructed using our specification for a single satellite, with a number of modifications as follows:

- the number of satellites is declared as a global variable, and multiple satellite modules are instantiated;
- the configuration of the satellite constellation is defined;
- redundant satellites that are usually called spare satellites are included.

Note that the last modification above is due to the fact that, in a real system, if an on orbit satellite fails, redundant on orbit satellites are used to move and replace them, to ensure the availability of the constellation.

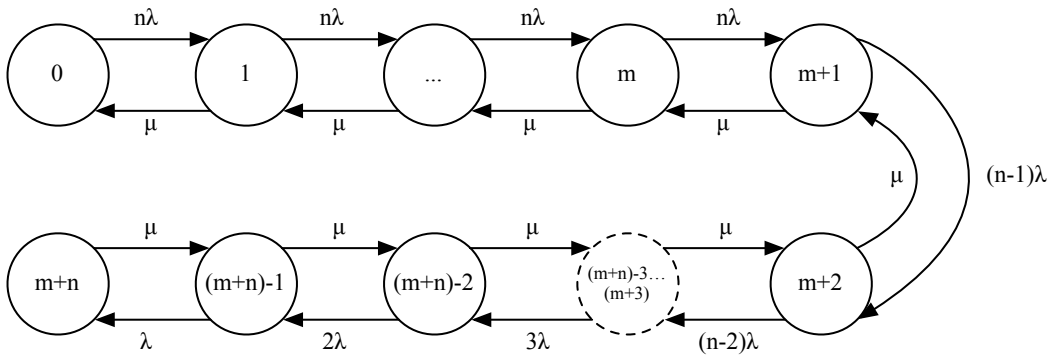


Figure 4.3: A reference model of a constellation of navigation satellites.

The reference model of the satellite constellation is depicted in Figure 4.3. The constellation has n satellites on orbit, and m spare satellites. If the on orbit satellites do not fail, the state of the constellation keeps n satellites available. Once an on orbit satellite fails, one of the spare satellites will replace it immediately to keep n in working condition. If any on orbit satellite fails and there is no spare satellite available to replace it, the number of satellites in the constellation will be reduced to a number smaller than n. Thus, spare satellites play a crucial effect on the availability of the satellite constellation.

In the reference model, if the number of satellites in the constellation is n and the number of spare satellites is m, where $m \geq 0$ and $n \geq 1$, the launch on schedule (LOS) strategy is to not launch a new satellite. At any time at most one satellite can be repaired. If any on orbit satellite fails, it is immediately replaced by a spare satellite, and repair of the failed satellite commences. If there are no spare satellites, the constellation must operate with fewer than n satellites.

Since the focus of our research is to apply the probabilistic model checking approach and to study its applicability to a satellite constellation, the object of our chapter is not limited to any specific navigation satellite system. The system we study here follows a standard configuration for global navigation system. Due to the fact that the current United States' GPS is the most widely used navigation system, parameter values of the constellation also refer to the latest basic parameter settings of such constellation. The parameter values are shown in Table 4.2.

Table 4.2: Parameters for the navigation satellite systems.

r	$MTBF(T)$	$MTTR$	n	m
0.80	15 years	5 months	24	3

Our PRISM specification is given as follows. Assume that the failure and repair rates of a satellite are λ and μ respectively. When the constellation is operating with n usable satellites, the state transfer rate of the constellation is $n\lambda$. When there are no spare satellites and satellites begin to fail, the transfer rate reduces accordingly to $n\lambda$, where n is the number of functioning satellites.

```
ctmc
const double r; //reliability of satellite system
const double MTBF;
const double life = MTBF*12*30*24; //life time of the satellite
const double lan = -life/log(r,2.71828183); //failure rate of
      the satellite
const double m = 3; //the number of the spare satellites
const double n = 24; //the number of the on orbit satellites
const double a = lan/n;
const double ai = lan/(n-(i-3)); //where 4<=i<=27
const double MTTR; //the mean time to repair
```

```
module constellation
        s:[0..27];
        [ai1] s = i -> 1/a : (s' = i+1); //where 0<=i<=3
        [ai2] s = i -> 1/ai : (s' = i+1); //where 4<=i<=26
        [bi] s = i -> 1/MTTR : (s' = i-1); //where 1<=i<=27
endmodule
```

4.4 Quantitative Properties and Automated Analysis

4.4.1 Desired Properties

We have identified the need to analyse reliability, availability, and maintainability properties of navigation satellite systems. In the GPS standard proposed in [34], there are two definitions of availability. The first one is the probability that the slots in the constellation will be occupied by a satellite transmitting a trackable and healthy Standard Positioning Service (SPS) Signal in Space (SIS). The second definition is the percentage of time that the SPS SIS is available to a SPS receiver. According to the same standard, there are two kinds of availability of satellites. The first is the per-slot availability, and the second is the constellation availability, which can be described as follows,

- Per-slot availability: The time that a slot in the constellation will be occupied by a satellite that is transmitting a trackable and healthy SPS SIS;
- Constellation availability: the time that a specified number of slots in the constellation are occupied by satellites that are transmitting a trackable and healthy SPS SIS.

In our research, we do not consider the environmental effect of the signal for the availability analysis. We only consider fault or failure of satellites. In our context, availability means the ratio of running time for normal satellites to total running time for both normal and failed satellites. The availabilities that we have analysed are: single satellite availability and satellite constellation availability.

The reliability of a satellite depends on planned interruptions, unplanned interruptions, and failure states in the system. The probability of successful launch is the reliability of the satellite, and the maintainability of the satellite is the probability that a satellite can be repaired on orbit. Generally, both reliability and maintainability can

be considered as availability properties of the satellite. Reliability must be sufficient to support the mission capability needed in its expected operating environment.

If reliability and maintainability are not adequately designed into satellite systems, there is risk that the design will breach desired availability requirements. Therefore, such system availability baseline is determined by the threshold of design or development costs, which is significantly higher due to resulting corrective action costs. This will cost more than anticipated to use and operate, or will fail to provide the expected availability.

Satellites will deteriorate with time due to failure mechanisms. We assume that time delay is a random variable selected from an exponential distribution, which is an assumption used in PRISM. According to system reliability theory [124], the reliability of a satellite $R(t)$ can be defined as:

$$R(t) = Pr\{T > t\} = e^{-\lambda t}, \tag{4.1}$$

from which we obtain:

$$\lambda = \frac{-lnR(t)}{E(s_i)}. \tag{4.2}$$

As defined in Section 3.3.1.2, $E(s_i) = \sum_{s_j \in S} R(s_i, s_j)$ denotes the total rate at which any transition outgoing from state s_i is taken. More precisely, $E(s_i)$ specifies that the probability of taking a transition outgoing from the state s_i within t time units is $1 - e^{-E(S_i)\cdot t}$

Satellite failures typically occur at some constant failure rate λ, and failure probability depends on the rate λ and the exposure time t. According to [32], typically failure rates are carefully derived from substantiated historical data such as mean time between failure ($MTBF$). We have:

$$\lambda = \frac{-lnR}{T} \implies \lambda = \frac{-lnR}{MTBF}, \tag{4.3}$$

where $t = T = MTBF$, and $MTBF$ is the design parameter or the statistics parameter. Referring to the latest characteristics of satellites used for Global Positioning Systems (GPSs), we assume the $MTBF$ of the satellite to be 15 years. As a result, $R = 0.80$ and $MTBF$ is 15 years. Further, the mean time to repair ($MTTR$) is 24 hours.

$$\mu = \frac{1}{MTTR}. \tag{4.4}$$

For the evaluation of the availability of the constellation, we focus on long-term failure effects to the constellation. The long term reflect the lifetime of the satellite,

and can be described by the $MTBF$ and $MTTR$. The $MTBF$ is used to get the parameter failure rate λ according to the Equation 4.3. The $MTTR$ is used to calculate the parameter repair rate μ according to the Equation 4.4.

4.4.2 Formal Analysis of a Single Satellite

In this section we describe the parameters used in our model and their values. We then use the PRISM probabilistic model checker to analyse some important properties of the single satellite system. The properties include reliability, maintainability, and availability. The temporal logic CSL is used to analyse the navigation systems because PRISM supports the use of CSL to verify properties of a CTMC. We then present and analyse our model checking results.

4.4.2.1 Reliability Properties

Reliability properties of a single satellite that we can analyse using PRISM include:

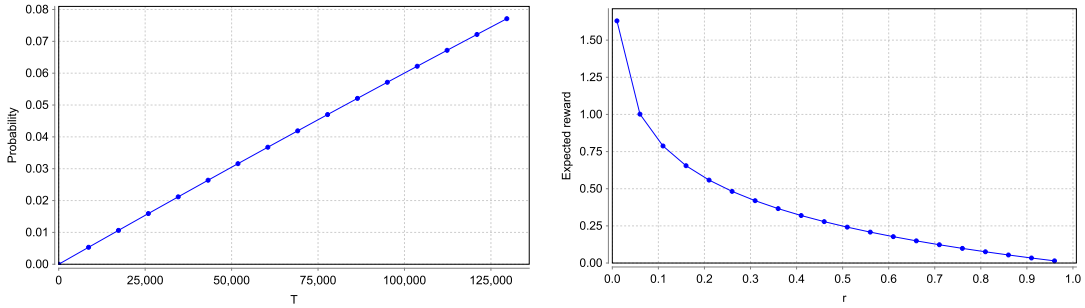
1. when $r = 0.80$, the probability that a satellite will need to be replaced by a new one in 15 years:
   $P_{=?}[F <= T\ s = 5];\ T = 129600$

2. when $r = 0.80$, the probability that a satellite will need to be replaced by a new one due to complete failure in 15 years over time T:
   $P_{=?}[F <= T\ s = 5];\ r = 0.80;\ T = 0 : 129600 : 8640$

3. when $r = 0.80$, how many times a satellite will need to be replaced by a new one in 15 years:
   $R_{=?}[C <= T];\ T = 129600;\ r = 0.80$
   The reward expression in the PRISM model is the following:
   rewards $''num\_replace''$
   $[g]\ true : 1;$
   $[e]\ true : 1;$
   endrewards

4. how many times a satellite will need to be replaced by a new one over different reliabilities, in 15 years:
   $R_{=?}[C <= T];\ T = 129600;\ r = 0.01 : 0.99 : 0.05$
   The reward expression is the same as that for reliability property 3.

In the properties above (and in all other contexts henceforth), 129600 is the lifetime of a satellite in hours (evaluating to approximately 15 years). Parameter r denotes reliability and proposition $s = 5$ asserts that there is a spare satellite on the ground. The expression $r = 0.01 : 0.99 : 0.05$ indicates that the reliability ranges from 0.01 to 0.99 with interval size 0.05.



(a) Reliability property 2

(b) Reliability property 4

Figure 4.4: Results of reliability properties of a single satellite.

The analysis results of reliability properties which we obtain from PRISM are as follows. The result of the property 1 is 0.0771; the result of property 2 is shown in Figure 4.4(a); the result of property 3 is 0.08; the result of property 4 is shown in Figure 4.4(b). From Figure 4.4(b), we can see that the number of times the satellite will have a failure and be unable to be repaired in 15 years is 0.08, under the precondition that the reliability is 0.80. If the reliability is set to 0.5, the number of vital failures will be smaller than 0.25 during 15 years. The number of times of unplanned interruptions can be also obtained from the PRISM by checking the rewards of the unplanned interruption, which is 29.95 times unplanned interruption for the satellite in 15 years.

4.4.2.2 Maintainability Properties

Maintainability properties of a single satellite that we can analyse using PRISM include:

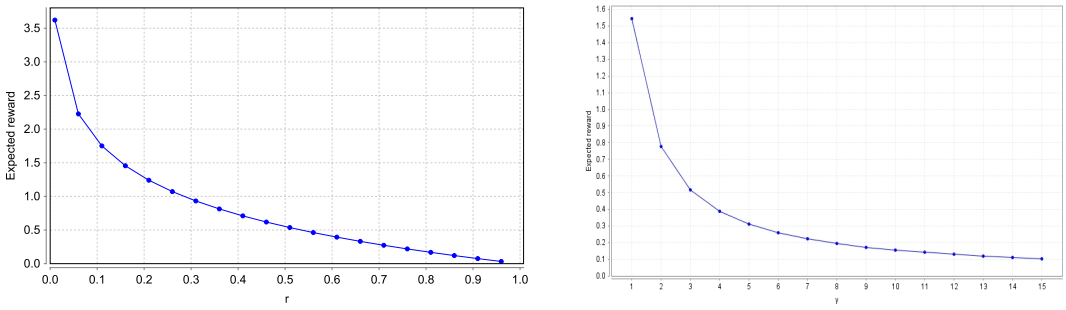
1. when $r = 0.80$, the number of times that satellites need to be repaired on orbit in 15 years:
   $R_{=?}[C <= T]$; $T = 129600$; $r = 0.80$
   The reward expression in PRISM model is the following:
   rewards $''num\_repair''$
   $[d]\ true : 1;$
   endrewards

2. the number of times that the satellite needs maintenance when the reliability is from 0.01 to 0.99 in 15 years:
   $R_{=?}[C <= T];\ T = 129600;\ r = 0.01 : 0.99 : 0.01$

3. the number of cases that a satellite needs to be repaired when the $MTBF$ is from 1st year to 15th years:
   $R_{=?}[C <= T];\ T = 129600;\ r = 0.01 : 0.99 : 0.01; MTBF = 1 : 129600 : 8640$
   The reward expression is the same as that for maintainability property 1.

4. when $r = 0.80$, the number of cases that a satellite needs to be repaired on orbit, but not eventually succeed in 15 years:
   $R_{=?}[C <= T];\ T = 129600;\ r = 0.80$
   The reward expression is the same as that for maintainability property 1.

The analysis results of maintainability properties which we obtain from PRISM are as follows. The result of the property 1 is 0.18; the result of property 2 is shown in Figure 4.5(a); the result of the property 3 is shown in Figure 4.5(b); the result of property 4 is 0.036. The number of times the satellite needs to be repaired on orbit over time is shown in Figure 4.5(a). When the reliability of the satellite is increased to 0.5, the number of times the satellite needs to be repaired will decrease to 0.5. Figure 4.5(b) illustrates that the number of times that the satellite needs to be repaired is below 1 when the $MTBF$ is 2 years.



(a) Maintainability property 2

(b) Maintainability property 3

Figure 4.5: Results of maintainability properties of a single satellite.

4.4.2.3 Availability Properties

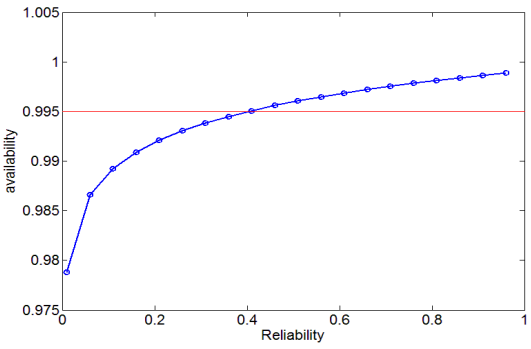
Availability properties of a single satellite that we can analyse using PRISM includes:

1. when $r = 0.80$, the availability of the satellite in 15 years:
   $(R_{=?}[C <= T])/T;\ T = 129600;\ r = 0.80$
   The reward expression in PRISM model is as the following:
   rewards $"availability"$
   $s = 0 : 1;$
   endrewards

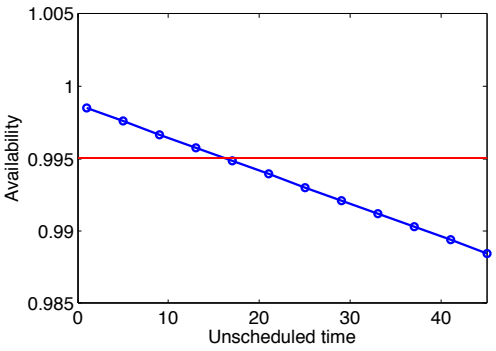
2. the availability of a satellite over the satellite reliability in 15 years:
   $R_{=?}[C <= T];\ T = 129600;\ r = 0.01 : 0.99 : 0.01$
   The reward expression is the same as that for availability property 1.

3. the relationship between satellite availability and its maintenance time taken for planned interruption:
   $(R_{=?}[C <= T])/T;\ T = 129600;\ r = 0.80,\ o = 1 : 48 : 3$
   The reward expression is the same as that for availability property 1.

The analysis results of availability properties which we obtain from PRISM are as follows. The result of property 1 is 129378 hours; the result of property 2 is shown in Figure 4.6(a); the result of property 3 is shown in Figure 4.6(b). From Figure 4.6(a) we see that if the reliability increases to 0.4, the availability of the satellite reaches 0.995. So if the required probability of the available satellite is 0.995, the reliability must have minimum value 0.4. Figure 4.6(b) indicates that if the required availability is 0.995, the time taken for planned interruption for the satellite will be smaller than 16 hours.



(a) Availability property 2



(b) Availability property 3

Figure 4.6: Results of availability properties of a single satellite.

4.4.3 Formal Analysis of a Constellation of Satellites

In this section, we analyse the properties of the satellite system that is made up of a constellation of navigation satellites. Similar to the case of the single satellite, we use PRISM to check the reliability, maintainability, and availability of the navigation system. We first present properties and their corresponding CSL, and then present and analyse the results of verifying these properties.

4.4.3.1 Reliability Properties

Reliability properties of the navigation satellite system that we can analyse using PRISM include:

1. when the reliability is 0.80, the probability that the number of the useable satellites in the constellation is smaller than 24 in 15 years:
   $P_{=?}[F <= T\ (s = 4)];\ T = 129600$

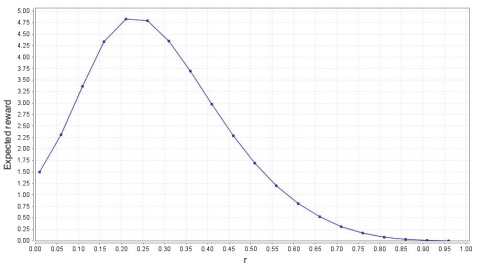
2. when the reliability is 0.80, the probability that the number of the useable satellites in the constellation is smaller than 22 in 15 years:
   $P_{=?}[F <= T\ (s = 6)];\ T = 129600$

3. the number of times that all redundant satellites fail in 15 years over the reliability and time:
   $R_{=?}[C <= T]$
   The reward expression in PRISM model is the following:
   rewards $''num\_fail''$
   $[a2]\ true : 1;$
   endrewards

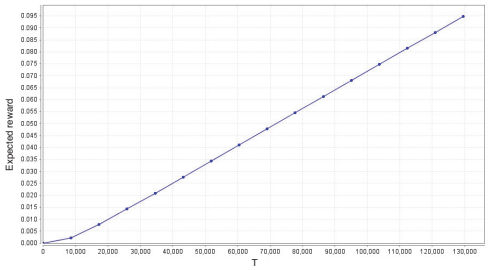
The proposition $s = n$ states that n satellites in the constellation fail. The analysis results of reliability properties which we obtain from PRISM are as follows. The result of property 1 is 0.01171; the result of property 2 is 0.0796; the result of property 3 is shown in Figure 4.7.

From Figure 4.7(a), when the reliability is between 0 and 0.25, the number of times that all redundant satellites need to be repaired is proportional to the reliability. As the reliability increases so does the number of required repairs, until the number of repairs reaches 4.76. However when the reliability is between 0.25 and 1, the number of times that all redundant satellite need to be repaired is inversely proportional to reliability.

This is due to the fact that when the reliability decreases to below a specific value, redundant satellites can no longer be repaired. According to Figure 4.7(b), the number of times that all redundant satellites need to be repaired is between 0 and 0.095 in 15 years.



(a) Reliability property 3: part 1



(b) Reliability property 3: part 2

Figure 4.7: Results of reliability properties of satellite constellations.

4.4.3.2 Maintainability Properties

Maintainability properties of the navigation satellite system that we can analyse using PRISM include:

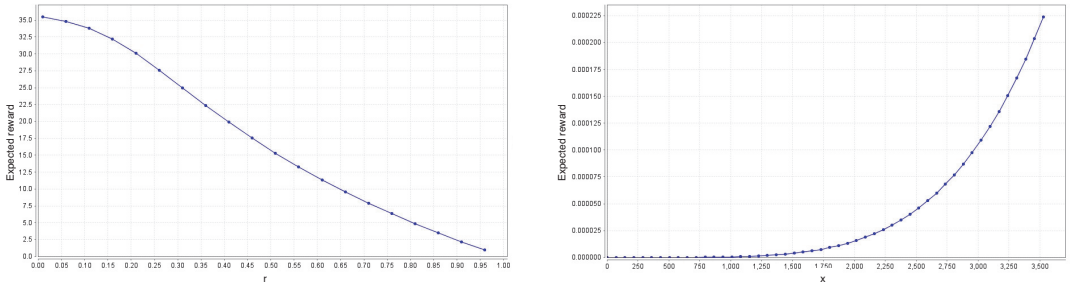
1. the average number of times to repair all satellites in the constellation in 15 years:
   $R_{=?}[C <= T]$
   The reward expression in PRISM model is shown as the following:
   rewards $"num\_repair"$
   $[bi]\ true : 1;$
   $for\ all\ i, 1 <= i <= 27$
   endrewards

2. The number of times to repair all satellites in the constellation over the reliability in 15 years:
   $R_{=?}[C <= T];\ r = 0.01 : 0.99 : 0.05$
   The reward expression in PRISM model is the same as that for maintainability property 1.

3. The probability of the case that the number of useable satellites in the constellation is smaller than 22 in 15 years over the number of times for repairing satellites:
   $P_{=?}[F <= T\ (s = 6)];\ T = 129600;\ MTTR = 0.1 : 3600 : 72$

The analysis results of maintainability properties which we obtain from PRISM are as follows. The result of property 1 is 5.18; the result of property 2 is shown in Figure 4.8(a) and the result of the property 3 is shown in Figure 4.8(b).

From Figure 4.8(a) we see that as reliability increases, the number of times that all satellites in the constellation need to be repaired over 15 years decreases from 35 to 2.5 when the reliability reaches 0.90. As depicted in Figure 4.8(b), the probability that the constellation consisting of n satellites with n is smaller than 22 in 15 years is 0.0225.



(a) Maintainability property 2 (b) Maintainability property 3

Figure 4.8: Results of maintainability properties of satellite constellation.

4.4.3.3 Availability Properties

Availability properties of the navigation satellite system that we can analyse using PRISM include:

1. the period of time that the constellation consisting of 24 satellites in 15 years:
   $R_{=?}[C <= T]$;,
   and reward expression in the PRISM model is shown as below:
   rewards $"reward"$
   $s = i : 1$; where $0 <= i <= 3$
   endrewards

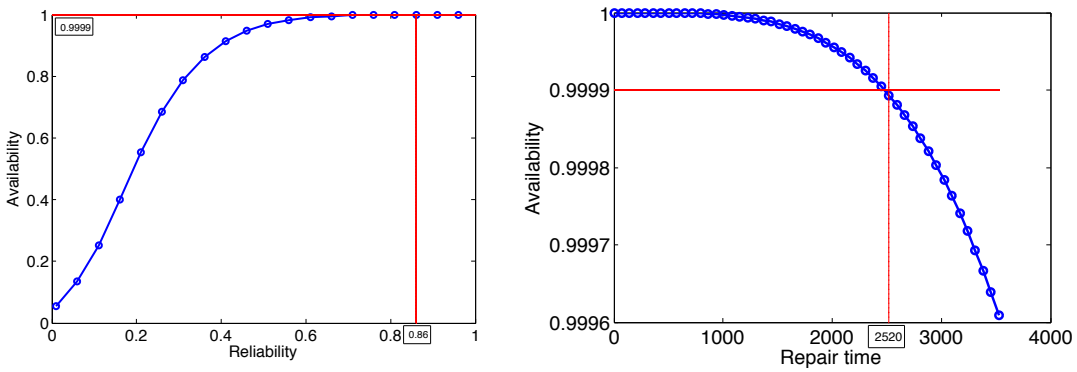
2. the availability of the constellation consisting of 24 satellites in 15 years:
   $(R_{=?}[C <= T])/T$;,
   and reward expression is the same as the availability property 1;

3. the availability of the constellation consisting of 24 satellites in 15 years over the reliability:
   $(R_{=?}[C <= T])/T;\ r = 0.01 : 0.99 : 0.05$,
   and reward expression is the same as the availability property 1;

4. the availability of the constellation consisting of 24 satellites in 15 years over the repair time:
   $(R_{=?}[C <= T])/T$; $MTTR = 0.1 : 3600 : 72$
   and reward expression is the same as the availability property 1.

The analysis results of availability properties which we obtain from PRISM are as follows. The result of property 1 is 129545 hours; the result of property 2 is 0.99958; the results of properties 3 and 4 are shown in Figures 4.9(a) and 4.9(b) respectively.

The availability of the satellite constellation as the reliability and time taken to repair satellites increases is shown in Figures 4.9(a) and 4.9(b) respectively. According to Figure 4.9(a), if the availability of the constellation is 0.9999 and the time taken to repair a satellite is 5 months, the reliability is at least 0.86. When the reliability is 0.80 , for the same availability requirement of the constellation, when the satellite has a fault or fails, the time taken to repair a satellite is at most 2520 hours (3.5 months).



(a) Availability property 3

(b) Availability property 4

Figure 4.9: Results of availability properties of satellite constellations.

4.4.4 Discussion of results

Since parameter settings of our formal models are based on GPS Block III which is the newest generation of GPS systems, our analysis results can be compared to existing GPS statistical analysis. According to a report of Lockheed Martin [65], a leading global security and aerospace company, the availability of the GPS Block III is given as 99.9%. The availability we evaluate in this chapter is close to the actual data. According to a further Lockheed Martin report [129], the constellation availability of the GPS Block III is given as 99.88%.

In this chapter, the availability we evaluate for two scenarios is in each case close to the actual data. This has proved to be both useful and efficient to use probabilistic model checking approach for the modelling and analysis of a singe satellite and a constellation of navigation satellites. To the best of our knowledge, we are the first to use the formal technique of probabilistic model checking to perform RAM analysis of satellite systems. These results indicates that our approach can also be applied to a wider range of quantitative properties of formal models taken from many application domains for satellite systems.

4.4.5 Benefits of the approach

To address the performance of satellite systems, it is essential to accurately quantify aspects such as reliability, availability and maintainability. There are two common techniques can be used for evaluating these features. One is the reliability block diagrams (RBDs) [37], and the other is the fault trees (FTs) [92, 116]. RBDs and FTs are both symbolic analytical logical techniques that can be applied to analyse complex system reliability and availability, and related properties. However, neither technique is suitable to evaluate probabilistic properties, due to the fact that they are static techniques. In a fault tree or reliability block formalism, it is necessary to assume the probabilities of each fault or failure are independent, while this is not the case in reality.

Other benefits of applying probabilistic model checking with PRISM for the specification and analysis of satellite systems is that the results can be plotted as graphs that can be inspected for trends and anomalies. Furthermore, we are able to compute exact quantities, rather than approximations based on a large number of simulations, thus enabling us to obtain complete and exhaustive conclusions for all possible parameter values. In addition, PRISM enables automated analysis. This helps manual analysis with automatic analysis support, thus making development more efficient and minimising human errors during the design phase.

There are also some disadvantages to using Markov models, not least that their specification, and the specification of useful properties, requires a high degree of mathematical skill. Markov models may be large and cumbersome in some cases, and the specification can be error-prone. In addition, as a system increases in complexity, so does the size of the state-space associated with a corresponding model. This results in a longer (possibly intractable) search.

4.5 Conclusion and Future Work

Reliability, availability and maintainability (RAM) analysis of systems has been indispensable in the design phase of satellites in order to achieve minimum failures or to increase mean time between failures ($MTBF$) and thus to plan maintainability strategies, optimise reliability and maximise availability. Traditional approaches are not suitable for performing RAM analysis of navigation satellite systems. We present formal models of both a single satellite system and a constellation of navigation satellites and logical specification of reliability, availability and maintainability properties. We have analysed a set of properties using PRISM.

There are many technical and theoretical challenges that remain to be addressed. In particular, satellite failure often forms part of more complex problems that show through different aspects of the engineering of space based systems. The technical challenges also include basic issues with the representation of safety and space mission critical characteristics of satellite telecommunications due to a group of satellites working together given the limitations of classical modelling approach.

In order to fully explore satellite behaviour, it will be necessary to exploit further formal techniques. For instance, if we want to model the mobility of connection between satellites it may be necessary to express behaviour via an extension to the π-calculus, and model check using PRISM (a technique identified in [110]). This kind of issue must be addressed in order to identify the causes of satellite system failure and to support the development of satellite systems.

As PRISM assumes events to occur according to an exponential distribution, we are limited to making the same assumption about the events in our systems. In fact, many types of satellite failure follow a different distribution. In particular, a number of failures of satellites have a Weibull distribution [17], which follows the conventional three-component bathtub curve which models a burn-in and wear-out phase for failure prediction. For future work, we will look at how to represent arbitrary distributions in probabilistic models, and to what extent such kind of distributions are able to be supported by the probabilistic model checking approach.

Chapter 5

Reliability Approximations of Satellite Subsystems with Weibull Failures

This chapter is based on previously published work [89, 90]. Satellite systems are complex due to the fact that they consist of a large number of subsystems. Each subsystem may itself have complex and different failure modes. For example, A satellite subsystem can suffer whole or partial failures, which may belong to a variety of failure classes. It has been shown that Weibull distributions can be used to properly model on-orbit failure behaviours of satellite subsystems. Markov chains have been used extensively to model reliability and performance of engineering systems or applications. However, the exponentially distributed sojourn time of CTMCs can sometimes be unrealistic for satellite systems that exhibit Weibull failures. In this chapter, we develop novel semi-Markov models that characterise failure behaviours, based on Weibull failure modes inferred from realistic data sources. A semi-Markov chain is a model in which state holding times are governed by general distributions, which is a natural extension of CTMCs. We approximate and encode these new models with Continuous-Time Markov Chains (CTMCs) and use the PRISM probabilistic model checker to answer meaningful questions concerning the reliability and performance of satellite subsystems. The key benefit of this integration is that CTMC-based model checking tools allow us to automatically and efficiently verify reliability properties relevant to industrial critical systems.

5.1 Introduction

Satellite systems are complex due to the fact that they consist of a large number of interacting subsystems (e.g., gyro / sensor / reaction wheels; control processors (CPs); and telemetry, tracking, and command (TTC)), which ensure redundancy without an unnecessary increase in power or mass requirements. Each subsystem may itself have complex and different failure modes. The failure modes are more complex than for conventional systems because of the limited opportunities for repair except through reconfiguration. A satellite subsystem can suffer whole or partial failures, which may belong to a variety of failure classes.

Simulation is a widely used and powerful technique for analysing satellite systems. Simulation is flexible since it allows us to use arbitrary normal distributions (such as Pareto, Weibull, or Lognormal distributions) in reliability studies. However, simulations may take a long time to run as the events (e.g., failure) that we are trying to model may be very rare. In addition, it involves complex design of valid simulation models and interpretations of simulation results.

Probabilistic model checking is a formal verification technique for the modelling and analysis of complex systems that exhibit stochastic behaviours. The automation in the probabilistic model checker PRISM is essential for analysing reasonably large and non-trivial Markov models with exponential distributions. Continuous-time Markov chain (CTMC) models have been used extensively to model reliability and performance of engineering systems or applications. However, the exponentially distributed sojourn time of CTMCs sometimes can be unrealistic to model satellite systems that exhibit Weibull failures. PRISM is useful for analysing realistic satellite subsystems, and we can obtain results with high accuracy if good approximations of Weibull distributions are possible without resulting in a state space that is too large to yield to model checking.

Model checking of semi-Markov chains is more complicated than that of Markov chains. Techniques for model checking semi-Markov chains have been developed [86, 80], whereas the methods are practically negative or infeasible. In recent years, applying practical probabilistic model checking tools to analyse non-Markov models has attracted a lot of attention. In [51], the authors analyse disk reliability of reasonable sized systems (such as RAID4/5/6) based on non-exponential distributions in PRISM [73]. Approximations of Weibull models are considered in [94], using an M-stage Erlang model, and in [141] where 3-state Hidden Markov Models (HMMs) are

used. In both cases, results are validated via simulation. In [125], a stochastic performance model is constructed and the hyper Erlang distribution of real-world data is used in PRISM to analyse a public bus transportation network in Edinburgh. In [27], phase-type distributions are used to analyse a collaborative editing system in PRISM.

It has been shown that Weibull distributions are able to properly model on-orbit failure behaviours of satellite subsystems [22, 21, 23]. It has also been shown that it is possible to approximate many common distributions using a sum of many exponential distributions, although this has proved computationally difficult. Given the maturity of a CTMC solver such as PRISM, and its focus on minimising state spaces, this difficulty is less of an issue as we show below. To show the effectiveness of this approach, we first show how the reliability of satellite systems can be computed more efficiently than by using simulation where satellite subsystems failure is modelled by Weibull distributions.

This chapter describes novel phase-type approximations to Weibull distributions, specifically approximating them by either Erlang or hyper-exponential distributions. This technique allows models containing Weibull distributions, which commonly occur in reliability modelling, to be approximately analysed using traditional CTMC model checkers such as PRISM. We also demonstrate our approximations on several Weibull-distributed failure times of satellite components inferred from realistic data sources, and investigate the goodness of such approximation and implementation.

5.2 Satellite Subsystems and Multi-state Failure

We propose an approach to building semi-Markov models for reliability analysis of satellite subsystems using a real-world database. The main data source consists of 1584 Earth-orbiting satellites launched between January 1990 and October 2008, which are provided by the SpaceTrak database[1]. The SpaceTrak launch and satellite analytical system and its database are used by most global key launch providers, satellite manufacturers, insurance companies, and satellite operators. It provides a variety of data and important information about satellite on-orbit failures and anomalies, as well as launch attempts since 1957. This has enabled us to predict and analyse failure rates.

One of the problems with stochastic approaches on-orbit is the prior validation given the specialised nature of many designs. Common core components e.g. NOAH and the DoD have a core platform that is then configured but many components and

[1] http://www.seradata.com/

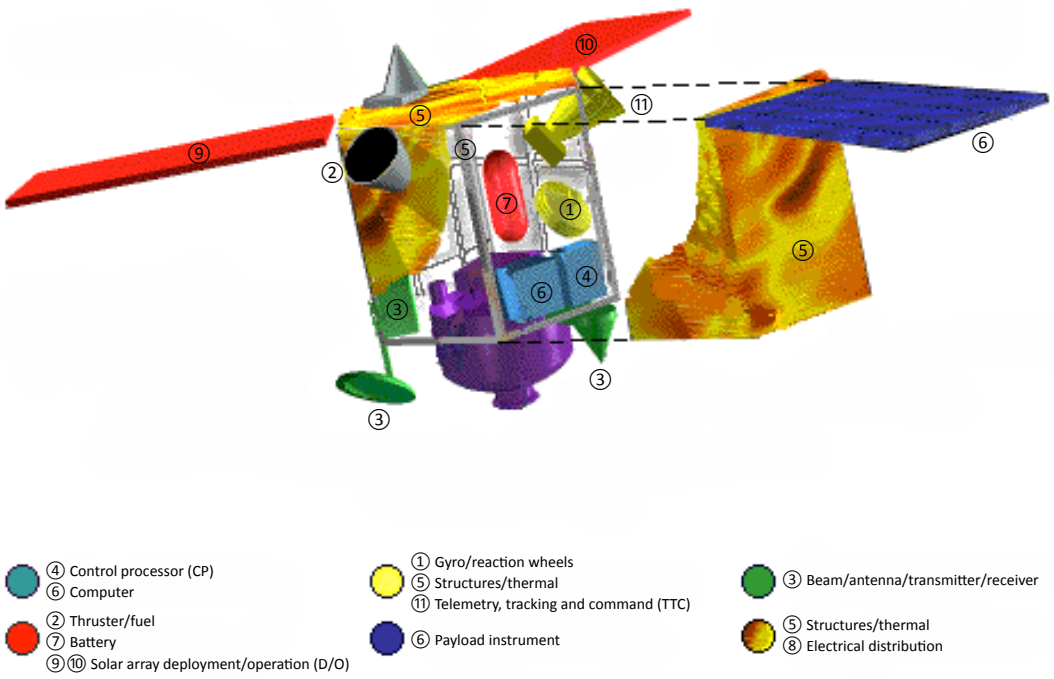


Figure 5.1: Satellite subsystems.

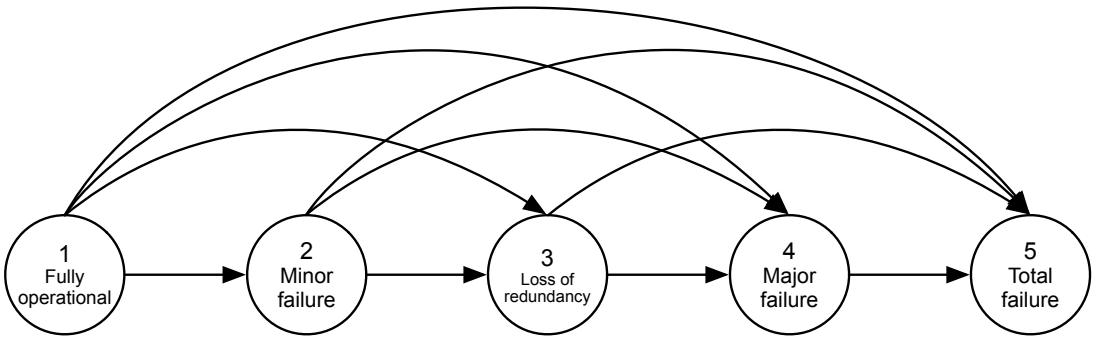


Figure 5.2: Multi-state and transition for satellite subsystems failure behaviour.

architectures are unique. The database used here is likely to provide a conservative base case but is not tailored to specific missions.

Several satellite subsystems are contained in the database. In this chapter, we use the following 11 subsystems (as shown in Figure 5.1): (1) Gyro/sensor / reaction wheel; (2) thruster / fuel; (3) beam / antenna operation / deployment; (4) control processor (CP); (5) mechanisms / structures / thermal; (6) payload instrument / amplifier / on-board data / computer / transponder; (7) battery / cell; (8) electrical distribution; (9) solar array deployment (SAD); (10) solar array operating (SAO); and (11) telemetry, tracking and command (TTC). In addition, we use the category of "unknown"to classify a satellite failure due to an unidentifiable accountable subsystem.

Instead of traditional binary models of reliability analysis for which satellite subsystems are considered to be either fully operational or suffering a complete failure, additional intermediate states which characterise partial failures are introduced (as shown in Figure 5.2). This multi-state modelling approach provides more insights into the failure behaviours of a satellite system and their relationship to the total failure through a finer level abstraction. These states are also defined in the SpaceTrak database, and are shown as follows:

- State 1: a satellite subsystem is fully operational;
- State 2: minor, temporary, or repairable failure that does not have a significant permanent impact on the operation of the satellite subsystem;
- State 3: major non-repairable failure that causes loss of redundancy to the operation of the satellite subsystem on a permanent basis;
- State 4: major non-repairable failure that affects operation of the satellite subsystems on a permanent basis;
- State 5: subsystem failure causing satellite retirement, which implies total failure of the satellite.

We approximate the semi-Markov chains in Figure 5.3 using the underlying semantics of continuous-time Markov chains (CTMCs). In CTMCs, the state space is discrete but time is continuous.

5.3 Approximation of Weibull Models

5.3.1 Weibull Distributions

In systems engineering, the Weibull distribution [140] is one of the most extensively used lifetime distributions for reliability analysis. It includes two parameters: (1) the shape parameter γ and (2) the scale parameter α, together with key formulas such as cumulative density function (CDF) and probability density function (PDF). A Weibull PDF is expressed as:

$$f(\gamma,\alpha,t) = \frac{\gamma}{\alpha}(\frac{t}{\alpha})^{\gamma-1}e^{-(\frac{t}{\alpha})^{\gamma}}, t \geq 0, \gamma, \alpha > 0 \tag{5.1}$$

and a Weibull CDF as:

$$F(\gamma,\alpha,t) = 1 - e^{-(\frac{t}{\alpha})^{\gamma}} \tag{5.2}$$

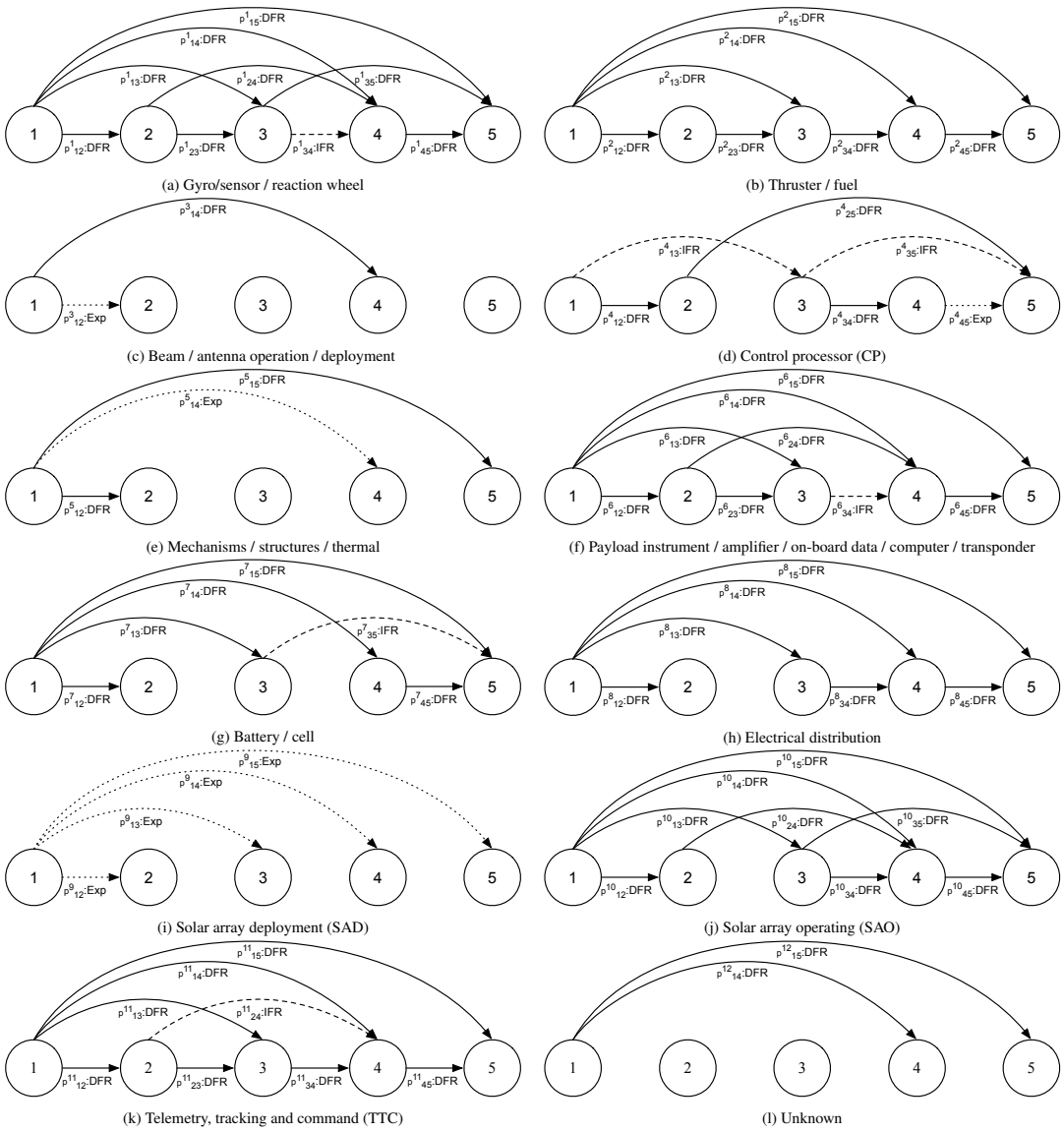


Figure 5.3: Multi-state and transitions for satellite subsystems failure behaviour.

We abbreviate f(t) and F(t) as the PDF and CDF of the Weibull distribution respectively, then the instantaneous failure rate is $\frac{f(t)}{1-F(t)}$. The failure rate is proportional to a power of time t. The shape parameter, γ, is equal to this power plus one.

The semantics of the Weibull distributions (also known as the bathtub curve) with different γ can be shown in Fig. 5.4 and explained as follows: (1) $\gamma < 1$ means that the failure rate decreases over time (decreasing failure rates). This occurs whenever a clear infant mortality [2] exists, and the failure rate decreases over time as the failure is discovered and the subsystem removed; (2) $\gamma = 1$ means that the failure rate is constant at any time. This is the useful life of the satellite ; (3) $\gamma > 1$ means that the failure rate

[2] infant mortality: a subsystem fails early due to defects designed into or built into it.

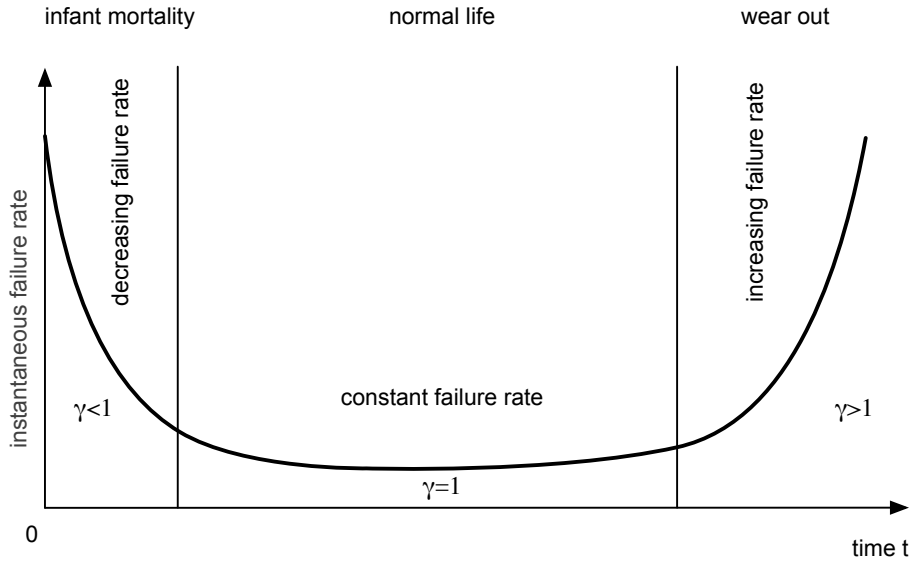


Figure 5.4: Semantics of the Weibull distribution (the bathtub curve).

increases with time (increasing failure rates). It occurs whenever a wear out exists, or a subsystem failure becomes more likely over time.

Generally, the ways to approximate the Weibull distributions is non-trivial. The simple technique of phase-type distributions is useful in some cases. Thus, we follow this line of work that Weibull IFR approximated by a M-stage Erlang distribution and Weibull DFR by a hyper-exponential distribution since there are intuitive and strong justifications for the model [94, 47]. Further, these general distributions provide simple mathematical structures such that the their underlying semi-Markov chains can be included in the Markov model framework.

5.3.2 Increasing Failure Rates (IFR)

A simple technique for the realisation of approximations to the Weibull distribution models is *matching moments*, where the mean is the first moment and the variance the second moment. We first consider the approximation of a Weibull distribution modelling increasing failure rates (IFR) using an M-stage Erlang distribution. The M-stage Erlang probability density function (PDF) can be expressed as:

$$f(M,\lambda,t) = \frac{\lambda^M}{\Gamma(M)} x^{M-1} e^{-\lambda x}, t \geq 0, \lambda > 0 \tag{5.3}$$

Table 5.1: Approximations of Weibull Distributions (IFR) using Erlang Distributions.

$P^i_{x,y}$	Weibull Distributions IFR		Erlang Distributions	
	γ	α	k	λ
P^1_{34}	1.1593	17	2	0.1239
P^4_{13}	1.1229	664	2	0.0031
P^4_{35}	1.0366	15	2	0.1353
P^6_{34}	1.2452	16	5	0.3352
$P^7_{3,5}$	28.6487	9	20	2.2652
$P^{11}_{2,4}$	2.8232	23	3	0.1464

The Erlang cumulative density function (CDF) can be expressed as:

$$F(M,\lambda,t) = 1 - e^{-\lambda x} \sum_{n=0}^{M-1} \frac{(\lambda t)^n}{n!} \tag{5.4}$$

According to [94], we have the first two moments of the M-Erlang:

$$m_1 = \frac{M}{\lambda},\ m_2 = \frac{M(M+1)}{\lambda^2} \tag{5.5}$$

As a result, we have:

$$M = \frac{m_1^2}{m_2 - m_1^2},\ \lambda = \frac{m_1}{m_2 - m_1^2} \tag{5.6}$$

where m_1 and m_2 are equated with the first two moments of the Weibull distribution with IFR, which are given as follows:

$$m_1 = \alpha\Gamma(\frac{\gamma+1}{\gamma}),\ m_2 = \alpha^2\Gamma(\frac{\gamma+2}{\gamma}) \tag{5.7}$$

The value of M is rounded to the nearest integer and the value of λ is recalculated depending on this rounded value, so that the mean is matched.

For example, we consider Weibull parameters for one of the satellite subsystems, which is the control processor. The Weibull parameters for the reliability of the control processor are given by: $\gamma = 1.4560$, $\alpha = 408$ (years). Then, according to equations (6)-(8), we obtain that $M = 2$ and $\lambda = 0.0054$ for the M-Erlang distribution. Using the Erlang distribution, the approximation result of the Weibull distribution with increasing failure rate for the relevant satellite subsystems is given in Table 5.1.

5.3.3 Decreasing Failure Rates (DFR)

The procedure for approximating Weibull distributions with decreasing failure rates (DFR) by hyper-exponential distributions can be summarised as follows, for details see [47].

First, we choose the number k of exponential components and k arguments: $m_1 > ... > m_i > m_{i+1} > ... > m_k$, for which the ratios $\frac{m_i}{m_{i+1}}$ have to be sufficiently small (e.g., $\frac{m_i}{m_{i+1}} \geq 10$).

Second, we choose the number n such that for all i, $1 < n < \frac{m_i}{m_{i+1}}$.

Then, regarding the cumulative density function (CDF) of the Weibull distribution (see equation (3)), we have a complementary CDF (CCDF) given by:

$$F^c(\gamma,\alpha,t) = 1 - F(t;\gamma,\alpha) = e^{-(\frac{t}{\alpha})^\gamma} \tag{5.8}$$

and we choose λ and p_1 to match the CCDF $F^c(t;\gamma,\alpha)$ (we abbreviate $F^c(t;\gamma,\alpha)$ as $F^c(t)$) at the arguments m_1 and nm_1, so we solve the two equations:

$$p_1 e^{-\lambda_1 m_1} = F^c(m_1) \tag{5.9}$$

and:

$$p_1 e^{-\lambda_1 n m_1} = F^c(nm_1) \tag{5.10}$$

for p_1 and λ_1. As a result, we obtain:

$$\lambda_1 = \frac{1}{(n-1)m_1} ln\left(\frac{F^c(m_1)}{F^c(nm_1)}\right) \tag{5.11}$$

and

$$p_1 = F^c(m_1) e^{\lambda_1 m_1} \tag{5.12}$$

Then, for $2 \leq i \leq k$, we have:

$$F_i^c(m_i) = F^c(m_i) - \sum_{j=1}^{i-1} p_j e^{-\lambda_j m_i} \tag{5.13}$$

and:

$$F_i^c(nm_i;) = F^c(nm_i) - \sum_{j=1}^{i-1} p_j e^{-\lambda_j n m_i} \tag{5.14}$$

and similarly, we solve further two equations:

$$p_i e^{-\lambda_i m_i} = F_i^c(m_i) \tag{5.15}$$

and:

$$p_i e^{-\lambda_i n m_i} = F_i^c(nm_i) \tag{5.16}$$

for p_i and λ_i when $2 \leq i \leq k-1$. As a result, we obtain:

$$\lambda_i = \frac{1}{(n-1)m_i} ln\left(\frac{F_i^c(m_i)}{F_i^c(nm_i)}\right) \tag{5.17}$$

and:

$$p_i = F_i^c(m_i)e^{\lambda_i m_i} \tag{5.18}$$

Finally, for $i = k$, we can have:

$$p_k = 1 - \sum_{j=1}^{k-1} p_j \tag{5.19}$$

and

$$p_k e^{-\lambda_k m_k} = F_k^c(m_k) \tag{5.20}$$

We have:

$$\lambda_k = \frac{1}{m_k} ln\left(\frac{p_k}{F_k^c(m_k)}\right) \tag{5.21}$$

Using the hyper-exponential distribution, the approximation result of the Weibull distribution with decreasing failure rate for the relevant satellite subsystems is given in Table 5.2.

5.4 Encoding Weibull Models with CTMCs in PRISM

5.4.1 Encoding the Weibull distribution with IFR

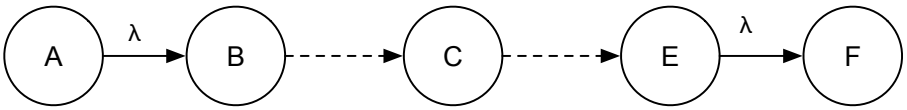


Figure 5.5: Modelling the Weibull distribution (IFR) with a CTMC.

Non-exponential holding time distributions can be approximated by inserting multiple intermediate states between any two conventional degradation states. We approximate a Weibull IFR with an Erlang distribution, which is a special case of a phase-type distribution. In Figure 5.5, $\frac{k}{\lambda}$ is the total transition interval between A and F, and the total number of intermediate stages used to approximate it, is $k-1$. The transition rate is proportional to k which ensures a constant total transition time.

In the PRISM model in Figure 5.6, the occurrence of the labelled action *sync* occurs with an Erlang distribution with scale μ and shape k. The special case of $k = 1$ is

Table 5.2: Approximations of Weibull distributions (DFR) using hyper-exponential distributions

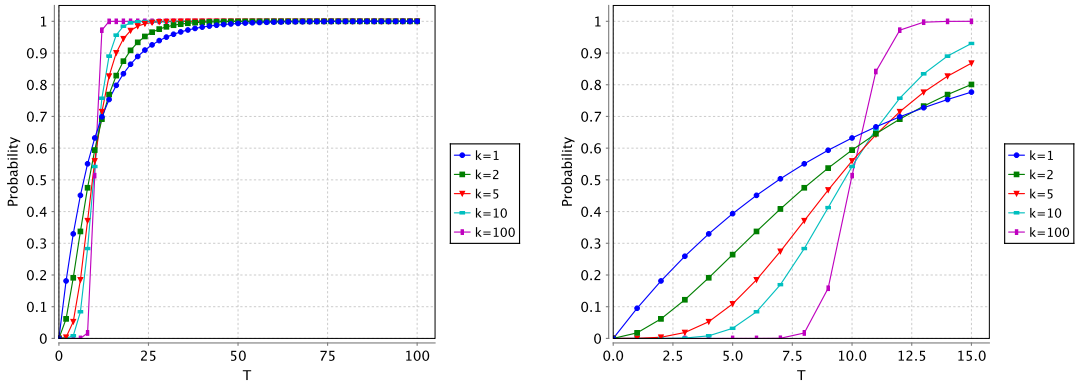
$P^i_{x,y}$	Weibull Distributions DFR		Hyper-exponential Distributions							
	γ	α	p_1	λ_1	p_2	λ_2	p_3	λ_3	p_4	λ_4
P^1_{12}	0.4482	12,526	0.8149	0.000117	0.1258	0.0038	0.0384	0.0433	0.0210	0.8802
P^1_{13}	0.4334	80,050	0.9074	0.000052	0.0630	0.0037	0.0189	0.0434	0.0108	0.9015
P^1_{14}	0.3815	210,126	0.9133	0.000039	0.0548	0.0038	0.0188	0.0444	0.0131	0.9903
P^1_{15}	0.5635	65,647	0.9518	0.000045	0.0377	0.0034	0.0077	0.0408	0.0028	0.7348
P^1_{23}	0.8229	59	0.0933	0.007895	0.6383	0.0132	0.2326	0.0458	0.0359	0.5320
P^1_{24}	0.5600	4,003	0.7852	0.000218	0.1631	0.0037	0.0378	0.0411	0.0139	0.7382
P^1_{35}	0.7115	221	0.3461	0.001866	0.5000	0.0058	0.1258	0.0404	0.0281	0.6022
P^1_{45}	0.4703	135	0.2068	0.000988	0.4133	0.0058	0.2396	0.0466	0.1404	0.8653
P^2_{12}	0.2724	531,935,049	0.9784	0.000006	0.0112	0.0039	0.0049	0.0467	0.0055	1.2407
P^2_{13}	0.4449	138,315	0.9312	0.000040	0.0477	0.0036	0.0136	0.0431	0.0074	0.8836
P^2_{14}	0.4763	8,591	0.8037	0.000140	0.1376	0.0038	0.0393	0.0427	0.0194	0.8398
P^2_{15}	0.3114	29,975,117	0.9698	0.000010	0.0170	0.0038	0.0068	0.0459	0.0063	1.1378
P^2_{23}	0.8867	46	0.0986	0.013020	0.6729	0.0181	0.2285	0.0608	n/a	n/a
P^2_{34}	0.6810	18	0.0022	0.009304	0.2842	0.0198	0.5152	0.0622	0.1984	0.6389
P^2_{45}	0.2632	589,301	0.8614	0.000037	0.0676	0.0041	0.0322	0.0473	0.0389	1.2775
P^3_{14}	0.2468	436,409,190	0.9675	0.000008	0.0156	0.0040	0.0074	0.0474	0.0095	1.3202
P^4_{12}	0.4785	127,433	0.9421	0.000039	0.0418	0.0035	0.0108	0.0424	0.0052	0.8357
P^4_{25}	0.3822	1,832	0.5754	0.000241	0.2423	0.0043	0.1046	0.0456	0.0778	1.0019
P^4_{34}	0.5808	218	0.2948	0.001201	0.4651	0.0055	0.1765	0.0430	0.0636	0.7168
P^5_{12}	0.3840	4,952,368	0.9739	0.000012	0.0168	0.0037	0.0055	0.0442	0.0038	0.9840
P^5_{15}	0.3572	19,794,952	0.9792	0.000008	0.0128	0.0037	0.0045	0.0448	0.0034	1.0356
P^6_{12}	0.4135	24,908	0.8380	0.000088	0.1057	0.0038	0.0348	0.0439	0.0216	0.9353
P^6_{13}	0.4278	86,653	0.9075	0.000051	0.0625	0.0037	0.0190	0.0435	0.0111	0.9104
P^6_{14}	0.4691	3,170	0.6988	0.000224	0.2040	0.0040	0.0643	0.0432	0.0329	0.8512
P^6_{15}	0.6701	119,172	0.9835	0.000024	0.0139	0.0031	0.0026	0.0623	n/a	n/a
P^6_{23}	0.2483	534,535	0.8431	0.000039	0.0729	0.0041	0.0363	0.0477	0.0478	1.3272
P^6_{24}	0.4607	131	0.2037	0.000960	0.4062	0.0058	0.2419	0.0469	0.1482	0.8814
P^6_{45}	0.2513	169,438,854	0.9615	0.000009	0.0187	0.0040	0.0088	0.0473	0.0110	1.3062
P^7_{12}	0.6109	332,145	0.9865	0.000015	0.0110	0.0032	0.0025	0.0672	n/a	n/a
P^7_{13}	0.3233	536,957,969	0.9895	0.000004	0.0061	0.0038	0.0023	0.0455	0.0021	1.1086
P^7_{14}	0.4134	357,357	0.9429	0.000029	0.0381	0.0037	0.0118	0.0437	0.0072	0.9332
P^7_{15}	0.9239	4,431	0.9744	0.000227	0.0256	0.0036	n/a	n/a	n/a	n/a
P^7_{45}	0.2355	1,915	0.4936	0.000152	0.1903	0.0045	0.1206	0.0496	0.1955	1.4244
P^8_{12}	0.4277	6,358,947	0.9846	0.000008	0.0105	0.0036	0.0031	0.0433	0.0018	0.9093
P^8_{13}	0.3064	3,760,000,000	0.9926	0.000002	0.0041	0.0038	0.0017	0.0459	0.0016	1.1489
P^8_{14}	0.3526	11,894,073	0.9739	0.000010	0.0160	0.0037	0.0057	0.0449	0.0044	1.0452
P^8_{15}	0.5215	144,569	0.9587	0.000033	0.0312	0.0034	0.0071	0.0415	0.0030	0.7814
P^8_{34}	0.9574	25	0.1368	0.032194	0.6978	0.0368	0.1654	0.0755	n/a	n/a
P^8_{45}	0.4618	376	0.3759	0.000593	0.3686	0.0048	0.1635	0.0449	0.0920	0.8698
P^{10}_{12}	0.3397	27,344,009	0.9774	0.000008	0.0135	0.0038	0.0050	0.0452	0.0041	1.0724
P^{10}_{13}	0.3117	29,510,906	0.9698	0.000010	0.0171	0.0038	0.0068	0.0459	0.0063	1.1371
P^{10}_{14}	0.4734	4,393	0.7382	0.000193	0.1801	0.0039	0.0544	0.0430	0.0273	0.8446
P^{10}_{15}	0.2527	34,600,000,000	0.9900	0.000002	0.0049	0.0040	0.0023	0.0472	0.0028	1.2993
P^{10}_{24}	0.9160	63	0.2409	0.011160	0.6328	0.0152	0.1263	0.0563	n/a	n/a
P^{10}_{34}	0.7384	9	0.0000	0.021656	0.1410	0.0396	0.5920	0.0885	0.2670	0.6213
P^{10}_{35}	0.5980	206	0.2862	0.001321	0.4775	0.0056	0.1765	0.0427	0.0597	0.6988
P^{10}_{45}	0.4301	3,802	0.6925	0.000196	0.1975	0.0040	0.0690	0.0440	0.0411	0.9111
P^{11}_{12}	0.4765	12,078	0.8305	0.000119	0.1195	0.0037	0.0335	0.0427	0.0165	0.8393
P^{11}_{13}	0.3816	3,795,383	0.9704	0.000013	0.0190	0.0037	0.0063	0.0443	0.0043	0.9885
P^{11}_{14}	0.9676	180	0.7918	0.005022	0.2082	0.0086	n/a	n/a	n/a	n/a
P^{11}_{15}	0.3111	30,621,064	0.9699	0.000010	0.0170	0.0038	0.0068	0.0459	0.0063	1.1385
P^{11}_{23}	0.6046	1,541	0.6913	0.000401	0.2388	0.0039	0.0530	0.0405	0.0169	0.6930
P^{11}_{34}	0.1417	28,879,780	0.8112	0.000024	0.0553	0.0043	0.0359	0.0505	0.0976	1.8264
P^{11}_{45}	0.3662	1,310	0.5251	0.000262	0.2571	0.0044	0.1203	0.0462	0.0975	1.0380
P^{12}_{14}	0.3766	1,471,383	0.9560	0.000019	0.0279	0.0037	0.0094	0.0444	0.0067	0.9983
P^{12}_{15}	0.4020	5,578,316	0.9791	0.000010	0.0139	0.0036	0.0043	0.0439	0.0028	0.9518

an exponential distribution. In Figure 5.7, we show the probability distribution of the delay, i.e. of $\mathbf{P}_{=?}[F \leq T\ x = 1]$ for different values of k, where $k = 1, 2, 5, 10, 100$.

Figure 5.7 illustrates the results of implementing the CTMC of Figure 5.5 in PRISM

```
ctmc
const int k; const double mu = 10/k;
module weibull_ifr1
        i : [1..k+1];
        [] i < k -> 1/mu : (i' = i + 1);
        [sync] i = k -> 1/mu : (i' = i + 1);
endmodule
module weibull_ifr2
        x : [0..1];
        [sync] x = 0 -> (x' = 1);
endmodule
```

Figure 5.6: Encoding the Weibull distribution (IFR) in PRISM.

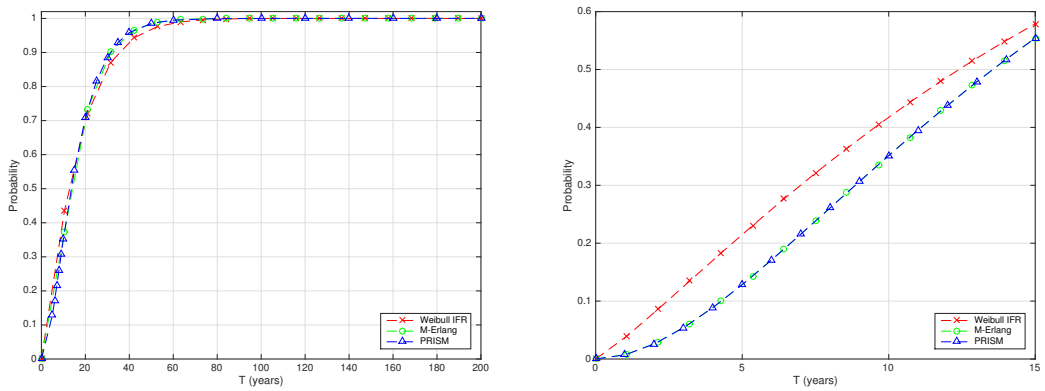


(a) T=100 (b) T=15

Figure 5.7: Results of encoding the Weibull distribution (IFR) in PRISM.

to show the approximation of a probability distribution with a constant delay (i.e. of $\mathbf{P}_{=?}[F \leq T \; x = 1]$ for different values of k, where $k = 1,2,5,10,100$) of both 100 years and 15 years. This is useful for modelling failure rates with multi-state, while preserving the Markov property. There is a clear and obvious trade-off here between the accuracy and the resulting expansion in the size of the model. For example, when $k = 100$, we can see from Figure 5.7(a), that the approximation is very close to the actual distribution. However, increasing k from 1 to 100 increases the size of the underlying model by 100.

To understand the differences better, we compare the CDF of the original Weibull IFR distribution with its approximation as an Erlang distribution and its implementa-



(a) T=200 (b) T=15

Figure 5.8: Comparative analysis of Weibull IFR, its approximation, and its implementation.

tion as a CTMC model in PRISM. As shown in Figure 5.8(a), the difference between Weibull and the other two curves apparently tends to zero, indicating the approximation and implementation both to be accurate for right long tail probabilities. In Figure 5.8(b), we see that the difference is at most 0.05, this is due to the fact that we lose a little accuracy in order to reduce the size of the state space associated with our PRISM model.

5.4.2 Encoding the Weibull distribution with DFR

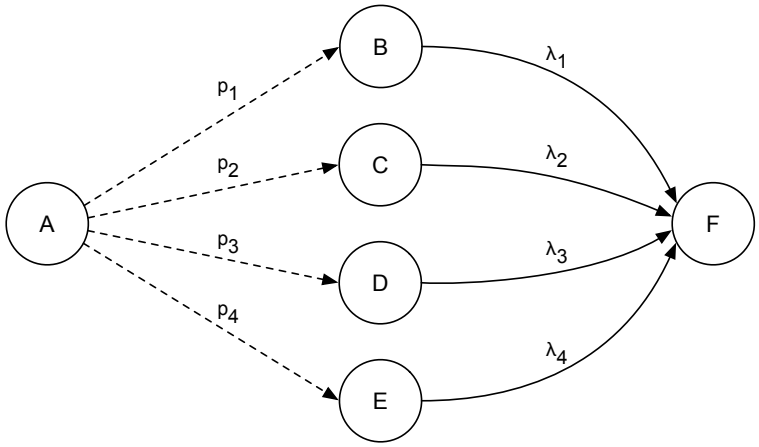


Figure 5.9: Modelling the Weibull distribution (DFR) with a CTMC.

We approximate a Weibull DFR with an hyper-exponential distribution, which is a mixture of exponential distributions. The hyper-Erlang distribution is also a gener-

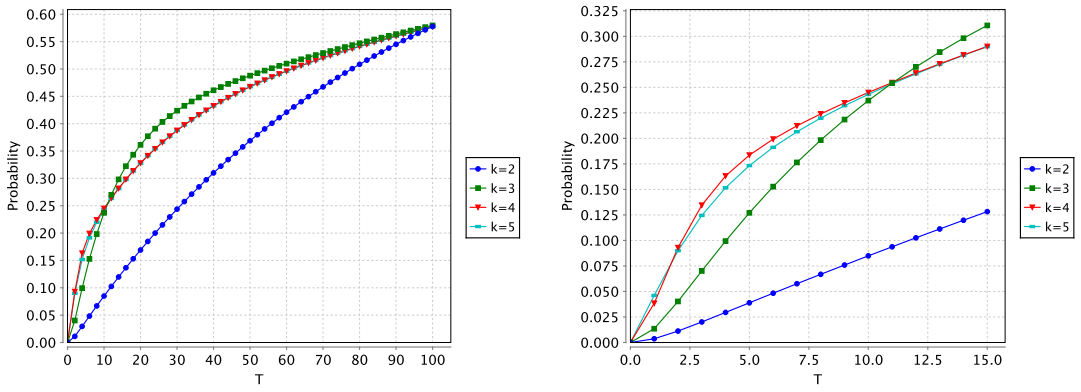
alisation of the hyper-exponential distribution, and the hyper-exponential is a special case of the hyper-Erlang. So, the hyper-exponential distribution is a phase-type distribution, and it can be described in terms of the time until absorption in a CTMC. An example of an hyper-exponential distribution having four branches ((p_1,λ_1), (p_2,λ_2), (p_3,λ_3),(p_4,λ_4)) represented by a CTMC model is illustrated in Figure 6. Dotted arrows indicate instantaneous probabilistic transitions, and solid arrows transitions with exponentially distributed durations.

```
ctmc
const double p1, p2, p3, p4, lambda1, lambda2, lambda3, lambda4;
module weibull_dfr
        s : [0..5] init 0;
        [] s = 0 -> p1 : (s' = 1) + p2 : (s' = 2) +
                    p3 : (s' = 3) + p4 : (s' = 4);
        [] s = 1 -> lambda1 : (s' = 5);
        [] s = 2 -> lambda2 : (s' = 5);
        [] s = 3 -> lambda3 : (s' = 5);
        [] s = 4 -> lambda4 : (s' = 5);
endmodule
```

Figure 5.10: Encoding the Weibull distribution (DFR) in PRISM.

Figure 5.11 illustrates the results of implementing the associated CTMC as shown in Figure 5.9 to give an approximation of the probability distribution of a constant delay (i.e. of $\mathbf{P}_{=?}[F \leq T\ x = 1]$ for $k = 2,3,4,5$ of both 100 years and 15 years. Although there is trade-off between the accuracy and the size of the resulting state space between $k = 2$ and $k = 4$, the difference is not so obvious between $k = 4$ and $k = 5$. Therefore, we consider $k = 4$ to be a good approximation parameter for the implementation of Weibull DFR in PRISM.

For the same purpose, we compare the CDF of the original Weibull DFR distribution with its approximation in a hyper-exponential distribution and its implementation with a CTMC in PRISM. As shown in Figures 5.12(a) and 5.12(b), for a time scale ($\alpha = 5000$ years), the difference between the Weibull DFR and the other two curves in the left short head is at most 0.01, and in the right long tails apparently becomes zero, indicating the approximation and implementation both to be accurate for a short scale for both left short head and right long tail probabilities. Though for a large scale



(a) T=100 (b) T=15

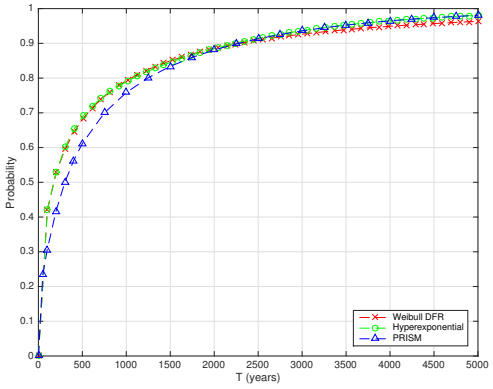
Figure 5.11: Results of encoding the Weibull distribution (DFR) in PRISM.

($\alpha = 50000$ years) in Figure 5.12(c), we can see that the difference can be very large in the right long tails. However, in Figure 5.12(d), for $T \leq 15$ years, the approximation and implementation both to be accurate for large scale and for left short head probabilities.

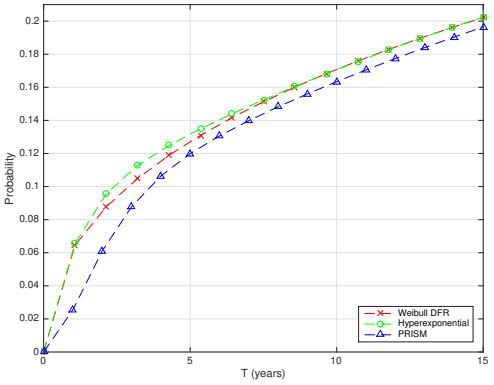
5.5 Conclusions and Future Work

In this chapter, we have shown that difficulties in modelling Weibull distributions for satellite failures can be handled if appropriate approximations and modelling methods are considered. We have also proposed novel non-exponential models that characterise failure behaviours, based on Weibull failure modes (both increasing failure rates and decreasing failure rates) inferred from real-world datasets. We have approximated and encoded these new models with CTMCs in the probabilistic model checker PRISM, and proved the goodness of such approximation and implementation.

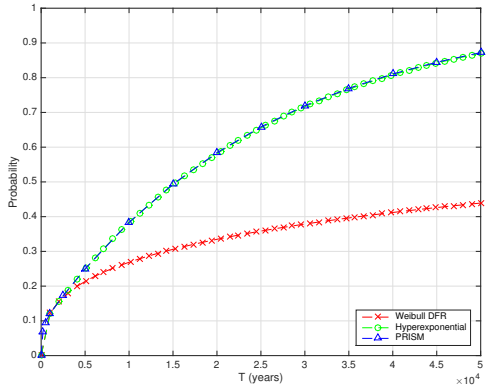
The key contribution of this chapter is that the CTMCs-based formalisms come equipped with mature model checking tools, such as PRISM, allow a wide range of analyses relevant to industrial critical systems to be performed automatically and efficiently. In future work, we will specify essential properties in the continuous stochastic logic (CSL) for CTMCs, and analyse desired satellite subsystems reliability and performance using PRISM. Another research direction is to use various techniques such as symmetry reduction [103, 74] for reducing the state space of the satellite system.



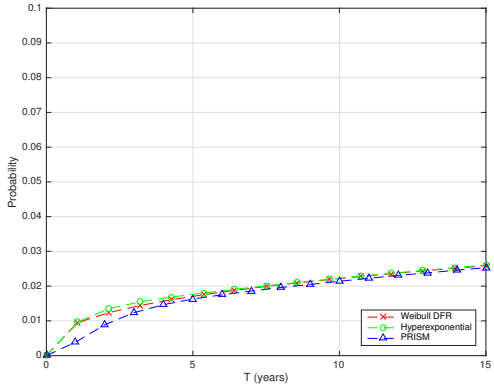
(a) Weibull DFR with small scale and T has value of 5000



(b) Weibull DFR with small scale and T has value of 15



(c) Weibull DFR with large scale and T has value of 50000



(d) Weibull DFR with large scale and T has value of 15

Figure 5.12: Comparison of the Weibull distribution (DFR) and its approximation and implementation.

Chapter 6

Availability Analysis of Satellite Navigation for Aircraft Guidance

This chapter is based on the papers [93, 91]. We highlight the promising application of probabilistic model checking for civil aviation missions. Whereas in Chapter 4 we used CTMCs model to analyse the reliability, availability, and maintainability of the standalone space segment, in this chapter we use MDPs to model the space, control, user segments.

We prove that it is able and suitable to analyse GNSS based positioning in aviation sectors for aircraft guidance. In particular, the focus is a widely used formalism called Markov Decision Processes (MDPs), and its generalisation to the analysis of quantitative aspects of a specific civil flight. We construct a formal model of the GNSS based positioning system for this application in the probabilistic π-calculus, a process algebra which supports modelling of concurrency, uncertainty, and mobility. After that, we encode our model in language of the PRISM. We then formalise and analyse the logical properties that relate to the dependability of the underlying system to check the system reliability and availability. We demonstrate how model specification and verification techniques can be successfully applied to the reliability and availability analysis of our case study.

6.1 Introduction

Satellite positioning systems are used within the transport industries such as marine, rail, and aviation sectors extensively. For example, in aviation, a three-dimensional global navigation satellite system (GNSS) enables an aircraft to determine its position

(latitude, longitude, and altitude) anywhere on or above the earth. Data transmitted from a navigation and communication satellite provides the user with the time, the precise orbital position of the satellite and the position of other satellites in the system. In the past, satellites were only deployed for military purposes. However nowadays they are used for a wide range of civil aviation applications, including navigation, communication, tracking, and flight management.

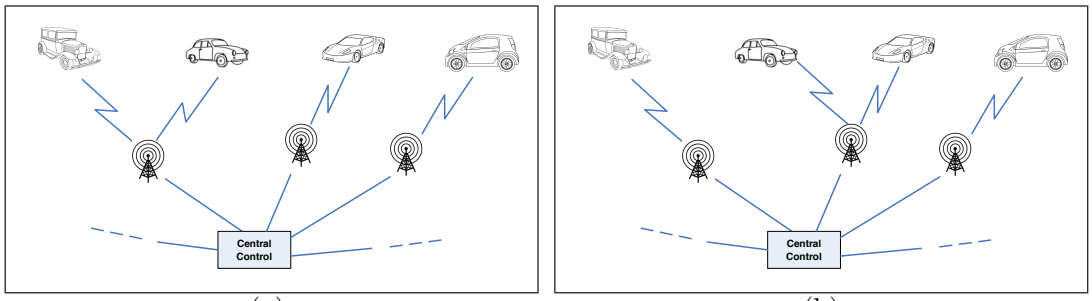
Our work has been inspired by a number of previous European Commission (EC) projects such as GADEROS, GRAIL, LOCASYS, and SATLOC. These projects have proved the feasibility of introducing GNSS in non-critical systems by means of theoretical studies and demonstrations. The current EC project EATS [10] proposes a novel positioning system based on different techniques that have proved useful from other industry viewpoints such as using information sources from GNSS, UMTS, and GSM. Furthermore, reliability, availability, maintainability, and safety (RAMS) analysis [67] is used to study the dependability properties of the technical solution in the critical applications, which aims to verify the proposed solution.

Availability requirements are identified as the most challenging obstacles towards GNSS aided positioning systems in [67]. Many approaches [87, 16, 134, 88] can be used to analyse availability properties. Among them, simulation, analytical analysis, and quantitative analysis are popular and practical. Each approach has its advantages and disadvantages that we have already discussed in Chapter 3. Since probabilistic model checking is a formal verification technique for analysing and verifying quantitative properties of a system's design, such as time, stochastic behaviour or resources. It is therefore highly suitable for modelling characteristics of our underlying system.

The mobility of an aircraft and satellites is universally recognised as an essential parameter for analysing the availability of satellite navigation systems. Our first task is to specify the communication between the airplane and satellites and their combined mobility. The second task is rendering these two models independently, in order to study the availability of the system in terms of different mobility models without changing the communication models.

In an example illustrated in Figure 6.1 (a), some cars are on the road, and each is connected by a unique wavelength to a single transmitter. The transmitters have fixed connections to a central control. On some events such as signal fading, a car may be switched to another transmitter. We distinguish two types of movement: the physical movement of vehicles and virtual movement of communication links between vehicles and transmitters. The two types of movements are independent, but the physical

movement of a vehicle may give rise to the virtual movement of its link to a transmitter (Figure 6.1 (b)).



(a) (b)

Figure 6.1: Component mobility

In our study, we mainly deal with this kind of relationship between movement of satellites and aircraft, and we study how the physical movement of both satellites and aircraft give rise to the mobility of links between them. As well as mobility, the π-calculus can be used to model parallel composition, alternative composition and sequential composition. Properties of the modelled system can be verified by studying the underlying labelled transition system. For this purpose, we specify the underlying models using the probabilistic π-calculus, an extension of the π-calculus [105, 106] for modelling mobile systems.

Therefore, we first specify the communication between an aircraft and the associated satellites, taking into account their combined mobility. We then analyse the models of the aircraft and satellite set independently before the combined system. Note that behaviour of the system contains a high level of uncertainty (e.g., in signal transmission unreliability due to solar radiation, etc.). Since PRISM only model checks expressions in the reactive modules language, and this does not allow for component mobility, it is not currently possible to model check the underlying process algebraic models directly. In order to allow for automatic verification using PRISM, the underlying Markov Decision Processes (MDPs) semantic models of our specification are first constructed using rules presented in [109].

The basic idea is to first build a Markov models that captures the behaviour of the system, and then to use the model to analyse precisely specified properties using temporal logics. This analysis is automatically performed using the model checker PRISM [77], using a combination of a traversal of the state transition system of the model and numerical computation. A PRISM specification can be generated directly

via a Markov chain variant described using the PRISM reactive modules language [6]. Alternatively, a high level model (using timed automata, or a process algebra, say) can be translated into the PRISM language. According to PRISM's manual, the latter approach can be more efficient than the former. This is due to the fact that PRISM is a symbolic model checker and the underlying data structures used to represent the system specification may function better when there is a high-level structure and regularity to exploit.

This chapter is organised as follows. In Section 6.2 we describe the underlying satellite navigation systems. In Section 6.3 the application of probabilistic verification is introduced. In Section 6.4 we present our formal specifications of a satellite navigation system for a specific navigation mission and their associated Markov decision processes respectively. Then, we verify availability properties using PRISM in Section 6.5. In Section 6.6 we discuss related work on analysing availability of satellite systems. Finally, in Section 6.7 we conclude the chapter.

6.2 Reference Models

6.2.1 GNSS-based Navigation Systems for Aviation

A GNSS-based navigation system consists of three major parts: the space segment, control segment and user segment. Recent theoretical research and standards have added a fourth environment segment to the satellite navigation system. The Galileo navigation system includes the environment segment in the composition of its navigation system. Although not explicitly mentioned, the environment segment is implied in the GPS system. To be conservative, the three traditional segments were used in this chapter as the components of the study, and the environmental segment was treated as an influencing factor on the system. Failure of any subsystem will lead to errors in the final positioning. Figure 6.2 is a schematic diagram of GNSS segments.

First, the monitor stations measure the pseudo-range of visible satellites every 6 seconds, correct them with ionospheric and meteorological data, smooth the measurement to generate data with a time interval of 15 seconds, perform smoothing again to generate data with a 15 minutes' time interval, and finally send the data to the master control station. The master control station is responsible for collecting and tracking data from each monitor station and calculating the satellite orbit and clock parameters using a Kalman estimator [52]. The results are transmitted to ground antennas and then

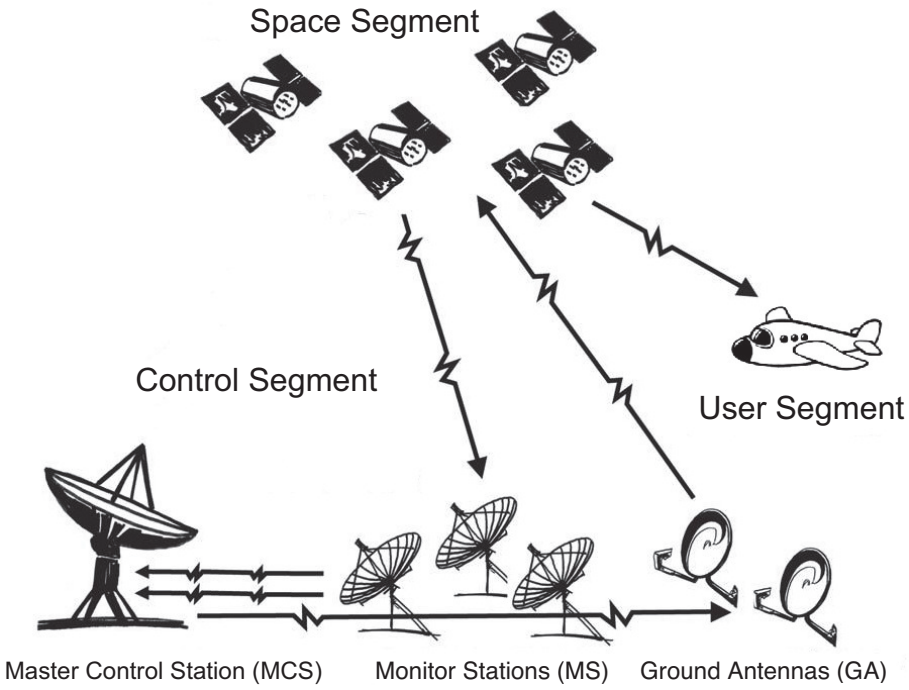


Figure 6.2: Three segments of a GNSS system.

to the satellite. Under the control of the master control station, the clock error, satellite ephemeris, navigation data, etc., are calculated and then transmitted to the corresponding satellite, and at the same time, the information is verified. The satellites transmit data associated with their current states to the users. The users need to use the position information provided by the satellites for positioning during navigation. According to [84], in general, at least four satellites are required to determine the user's position.

The accuracy of the information that each subsystem provides is critical and depends directly on the navigation accuracy. From the monitor station to the master control station, from the master control station to the ground antenna, from the ground antenna to the satellite, and from the satellite to the user, the entire process is implemented by information transmission. Errors may exist in the process of information transmission, and if these errors are passed on all the way to the user, the position provided by the navigation system is unusable.

6.2.2 Reference Models

In our study, commercial aircraft is considered to be the user, and the analysis considers the impact of navigation satellites' availability on aircraft navigation throughout the

mission of the specific flight from Beijing to Guangzhou. Figure 6.3 shows a schema of satellite navigation. At least four satellites are required for satellite navigation. In the schematic diagram, the user receives navigation signals from satellites with the serial numbers *A*, *B*, *C*, and *D*. However, both users and navigation satellites are constantly moving as the users are processing the information. Thus, the information that the users receive from the navigation satellites is also constantly changing.

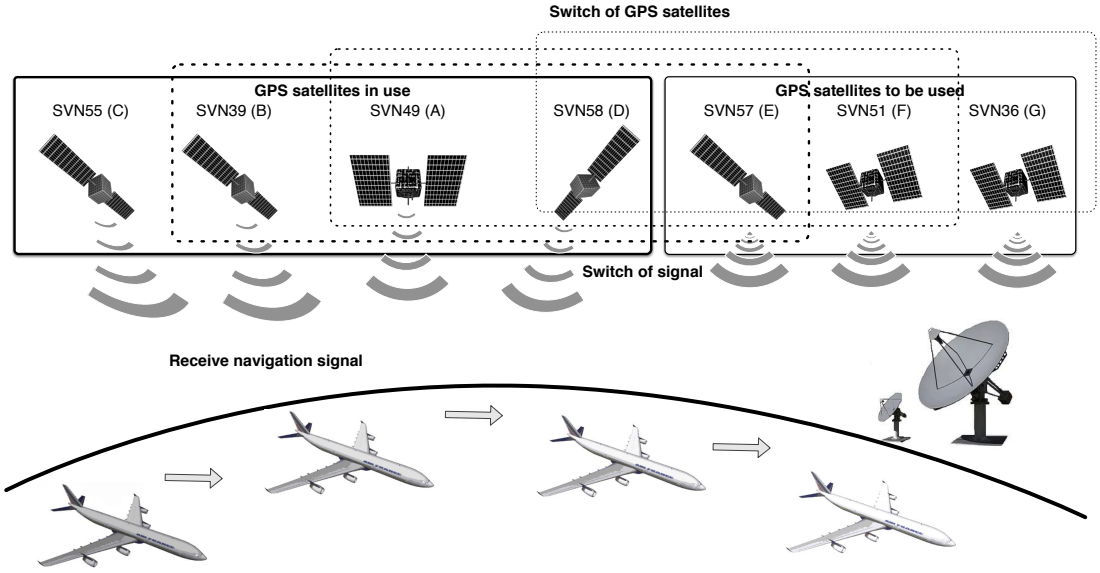


Figure 6.3: GNSS (GPS) based navigation for an air line.

A particular flight was studied in this chapter. The flight was from Beijing to Guangzhou, and the entire flight time was 2 hours 35 minutes. The specific time was January 2, 2012 (Beijing time); the flight departed at 12:00 and arrived in Guangzhou at 14:39. The entire flight was guided sequentially by 17 navigation satellites.

Although the airplane could generally receive satellite signals from more than 4 satellites at a time, usually only the signals from the four satellites with the best signals were used by the receiver for calculating the position. Therefore, 7 out of 17 satellites were determined to be the navigation satellites to be analysed in this study based on their navigation times and the task of the flight.

The space segment of the satellite system is a mix of old and new satellites. The 7 satellites of the space segment used are based on 3 models as shown in Figure 6.4, including Block IIA (2nd generation, “advanced”), Block IIR (“replenishment”), and Block IIR-M (“modernised”). The model GPS III in Figure 6.4 is still in production, and will be available for launch to replace the old satellites in 2016. The SVNs (Space

Figure 6.4: Satellite models used (reproduced from [85]).

Vehicle Numbers) of these satellites were $SVN49$, $SVN39$, $SVN55$, $SVN58$, $SVN57$, $SVN51$ and $SVN36$. The parameters of navigation satellites are shown in Table 6.1. The life of satellite is given in "years (y)", and the duration is given in "minutes (m)".

The system comprises 5 components: the satellite, monitor station, master control station, ground antenna and user. Each subsystem transmits information to objects to which it is connected. The user receives a satellite signal. The satellite receives information from the ground antenna which it then transmits to the monitor station and the user. The monitor station receives information from the satellite and transmits it to the master control station. The master station analyses the data from the monitor station and transmits it to the ground antenna. The ground antenna receives the control commands from the master control station and sends them to the satellite.

Table 6.1: Parameters of navigation satellites.

No	SVN	Launch date	Model	Life (y)	Reliability	Navi. interval	Duration (m)
A	49	24 Mar 2009	Block IIRM	10	0.8	12:00-14:29	149
B	39	26 Jan 1993	Block IIA	7.5	0.7	12:00-13:55	115
C	55	17 Oct 2007	Block IIRM	10	0.8	12:00-13:15	75
D	58	17 Nov 2006	Block IIRM	10	0.8	12:00-14:35	155
E	57	20 Dec 2007	Block IIRM	10	0.8	13:15-14:35	80
F	51	11 May 2000	Block IIR	7.5	0.75	13:55-14:35	40
G	36	10 Mar 1994	Block IIA	7.5	0.7	14:29-14:35	6

The US National Geospatial-Intelligence Agency (NGA) provides GPS satellites' status data available daily[1]. The space segment consists of 7 satellites due to the fact that the navigation mission requires a minimum of 7 satellites, which are identified as A, B, C, D, E, F and G. They receive information from ground stations and transmit navigation information to the user.

6.3 The Formal Approach

Generally, simulation is the common testing and validation approach used for verification of such systems. Given a system, a finite subset of the possible scenarios are selected in a specific simulation environment, and then statistical analysis techniques are applied to obtain probabilistic results on that system. Simulation based verification has been unable to keep pace with the growth in design complexity. As simulation requires the number of scenarios and simulation environments to be restricted, and so one cannot ensure that all conditions have been covered. Formal verification, on the other hand, can be applied to model and verify all scenarios. One automated method of verification is model checking. In particular, probabilistic model checking has proved to be a suitable formal verification technique for exposing errors in satellite systems, mainly due to classical concurrency errors.

Our formal approach mainly consists of four stages as illustrated in Figure 6.5. First, we model in the probabilistic π-calculus the behaviour of the whole mission. This model is composed of two separate models characterising the communication between different segments and their mobility. The latter must be able to be modified without changing the former. Second, the global model is translated into PRISM, and is

[1] `http://www.navcen.uscg.gov/?Do=constellationStatus`

then internally generated into an MDP (stage 1). The availability requirements that the system must satisfy are formalised in Probabilistic Computation Tree Logic (PCTL) [53] properties (stage 2). These formal quantitative properties are then checked with PRISM (stage 3). They can be checked according to our specific flight (from Beijing to Guangzhou) navigation mission. Finally, we analyse the results given by PRISM (stage 4).

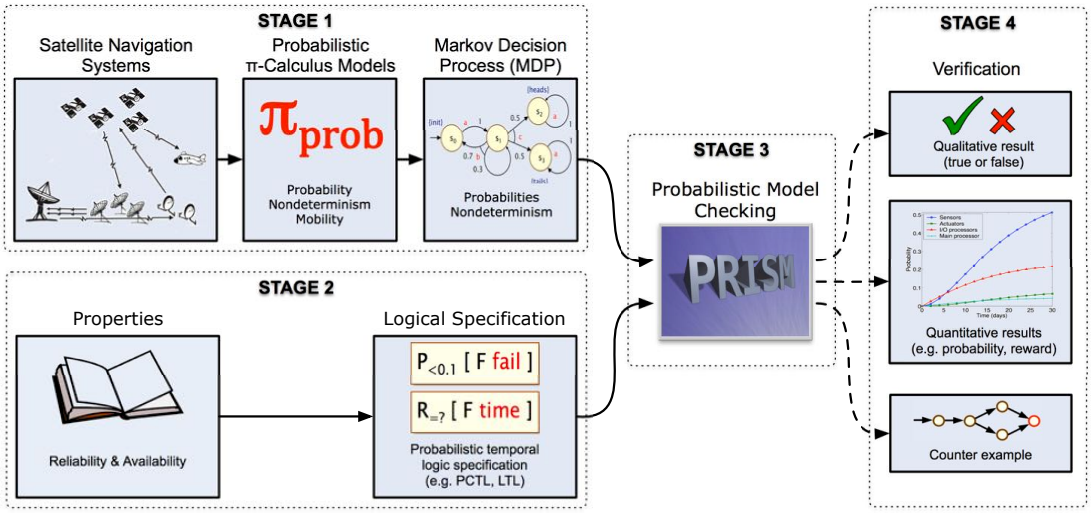


Figure 6.5: Stages in probabilistic verification.

6.3.1 Overview of the Probabilistic π-Calculus

The probabilistic π-calculus is a probabilistic extension of the π-calculus, which is a process algebra. π-calculus are used to model communicating, distributed, and mobile systems, and it provides strong techniques to reason about systems with concurrency at the modelling level. Communication, either inside the system or between the system and its environment, are modelled by synchronous actions on shared channels. It allows channel names to be sent along the channels themselves, thereby enabling one to model dynamically changing networks. Mobility is of central importance in the π-calculus. To appreciate the definition of the probabilistic π-calculus and the mobility our case study in satellite navigation for aviation, we illustrate what kind of mobility that the π-calculus is suitable for.

6.3.1.1 The Probabilistic π-Calculus

For some classes of concurrent and mobile systems, probabilistic behaviours can be key ingredients. Satellite navigation systems, for instance, can exhibit probabilistic behaviours due to either unreliable communication or component failures. We can model such behaviours with the probabilistic π-calculus (π_{prob}). The basic component of probabilistic π-calculus is its syntax as determined by the well-formed combination of operators and more elementary terms. We use "terms" to describe systems, and these are then mapped to labelled transition systems (LTSs). More explicitly, the states of a LTS are just "terms" of the probabilistic π-calculus while the labels of transitions between states represent the actions or the interactions that are possible from a given state and the state that is reached after the action is performed by means of actions.

The probabilistic π-calculus adds a discrete probabilistic choice operator to the classical π-calculus (only non-deterministic choice operator exists). This probabilistic operator associates internal actions with probabilities.

Definition 10. ***(Syntax).*** *We assume P and P_i range over terms and α ranges over actions. We assume a countable set of names $\mathcal{N}$ that range over x, y, x_i, where $i \in \{1,2,...,n\}$. A process P is defined in π_{prob} using the following syntax:*

- $\alpha ::= \tau \mid x(y) \mid \bar{x}\langle y \rangle$
- $P ::= \mathbf{0} \mid \alpha.P \mid \sum_{i\in I} P_i \mid \sum_{i\in I} p_i\tau.P_i \mid P|P \mid \nu xP \mid [x=y]P \mid A(x_1,x_2,...,x_i,...,x_n),$

where I is an index set, $p_i \in (0,1]$ with $\sum_{i\in I} p_i = 1$, and A is a process identifier. We now informally describe the calculus.

The inactive process $\mathbf{0}$ can perform no actions. The process $\alpha.P$ performs action α and then evolves into process P, where α is one of three types: τ is the silent (invisible) action that corresponds to an internal interaction between sub-processes, $x(y)$ is an input action in which a process receives a name y on channel x, and $\bar{x}\langle y \rangle$ is an output action, in which a process sends a name y on channel x. There are two types of summation: nondeterministic choice $\sum_{i\in I} P_i$ and probabilistic choice $\sum_{i\in I} p_i\tau.P_i$. The first is common in the standard π-calculus, and the second is a new operator in π_{prob}. As for π_{prob}, branches of the probabilistic choice operator are normally prefixed with τ actions. Thus, the process $\sum_{i\in I} p_i\tau.P_i$ randomly selects an index $i \in I$ with probability p_i, performs a τ action, and then evolves to P_i. The process $p\tau P_1 \oplus (1-p)\tau P_2$ is a special case of the process $\sum_{i\in I} p_i\tau.P_i$ that only consists 2 branches, such as $0.5\tau.\mathbf{0} \oplus 0.5\tau.\mathbf{0}$.

The parallel composition of processes P_i and P_j is $P_i|P_j$, and it can either proceed in an asynchronous manner or synchronise between P_i and P_j via matching input and output actions. The restriction νxP locally sets the scope of x in process P, so x is treated as a new and unique name within P. The match $[x=y]$ checks whether names x and y are identical, so the process $[x=y]P$ can evolve into process P only if the match $[x=y]$ is satisfied. Finally, $A(x_1,x_2,...,x_i,...,x_n)$ corresponds to a process definition clause and is used in the context $P = A(x_1,x_2,...,x_i,...,x_n)$.

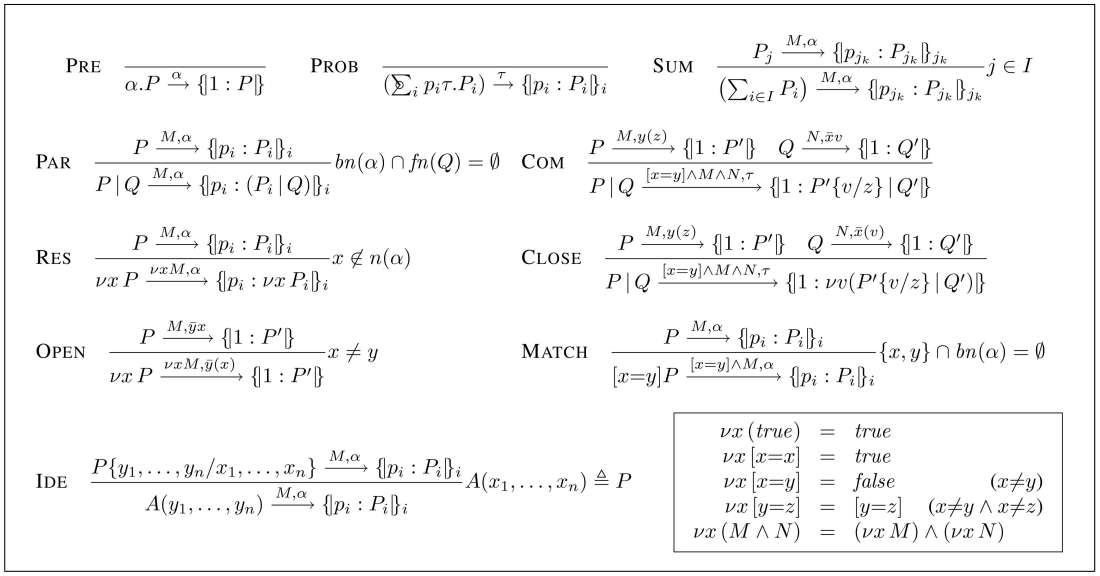


Figure 6.6: The symbolic semantics for π_{prob} (reproduced from [109]).

The operational semantics of π_{prob} are typically expressed in terms of Markov Decision Processes (MDPs) or Probabilistic Automata (PAs). The symbolic semantics of π_{prob} is expressed in terms of *probabilistic symbolic transition graphs* (PSTGs). These are a simple probabilistic extension of the *symbolic transition graphs* introduced in [56].

Definition 11. *Let P be a π_{prob} process. The probabilistic symbolic transition graph (PSTG) representing the semantics of the process P is a tuple (S,s_{init},$\mathcal{T}_{prob}$) where:*

- *S is a finite set of symbolic states, each of which is a term of the probabilistic π-calculus;*
- *$s_{init} \in S$, the initial state, is the term P;*
- *$\mathcal{T}_{prob} \subseteq S \times Cond \times Act \times Dist(S)$ is the probabilistic symbolic transition relation and is the least relation given by the rules in Figure 6.6.*

In the above, *Cond* denotes the set of all conditions (finite conjunctions of matches) over the set of names $\mathcal{N}$. *Act* is a set of actions of basic types: τ, $x(y)$, $\overline{x}\langle y\rangle$, where $x, y \in \mathcal{N}$. $Dist(S)$ is the set of probability distributions over S. The notation $Q_i \xrightarrow{M,\alpha} \{|p_i : R_i^i|\}$ is used for the probabilistic symbolic transition $(Q, M, \alpha, \mu) \in \mathcal{T}_{prob}$, where $\mu(R) = \sum_{Q_i=R} p_i$ for any π_{prob} term R. The multi-sets [109] are used to ensure that processes with duplicate components such as $Q = 0.5\tau.\mathbf{0} \oplus 0.5\tau.\mathbf{0}$ have transition of the form $Q_i \xrightarrow{\tau} \{|0.5 : \mathbf{0}, 0.5 : \mathbf{0}|\}$.

6.3.1.2 An Example of π_{prob} Processes

To illustrate the idea of probabilistic models, we specify a simple π_{prob} model of a set of traffic lights and drivers example. The description of the example is based on paper [123]. The process P_{light} models the traffic lights signalling to drivers, which are are probabilistically red, yellow, or green.

$$P_{light} \triangleq 0.45\tau.\overline{a}\langle Red\rangle.P_{light} \oplus 0.1\tau.\overline{a}\langle Yellow\rangle.P_{light} \oplus 0.45\tau.\overline{a}\langle Green\rangle.P_{light}$$

Here, the traffic light is red with probability 0.45, yellow with probability 0.1, and green with probability 0.45. We distinguish drivers according to how they behave depending on the colours of the lights they see. A cautious driver is modelled by the process P_{c_driver} as follows:

$$\begin{aligned}
P_{c_driver} &\triangleq a(x).([x = red]P_{c_red} + [x = yellow]P_{c_yellow} + [x = green]P_{c_green}) \\
P_{c_red} &\triangleq 0.2\tau.\overline{b}\langle braking\rangle.\mathbf{0} \oplus 0.8\tau.\overline{b}\langle stopped\rangle.\mathbf{0} \\
P_{c_yellow} &\triangleq 0.9\tau.\overline{b}\langle braking\rangle.\mathbf{0} \oplus 0.1\tau.\overline{b}\langle driving\rangle.\mathbf{0} \\
P_{c_green} &\triangleq \overline{b}\langle driving\rangle.\mathbf{0}
\end{aligned}$$

A cautious driver sees what colour the light is (through the form of match $[x = y]$) and behaves accordingly (through the probabilistic choice: $\sum_{i \in I} p_i\tau.P_i$). If it is red, he brakes or stops. If it is yellow, mostly likely he brakes. If it is green, he drives on. Similarly, an aggressive driver can be modelled by the process P_{a_driver} as follows:

$$\begin{aligned}
P_{a_driver} &\triangleq a(x).([x = red]P_{a_red} + [x = yellow]P_{a_yellow} + [x = green]P_{a_green}) \\
P_{a_red} &\triangleq 0.3\tau.\overline{b}\langle braking\rangle.\mathbf{0} \oplus 0.6\tau.\overline{b}\langle stopped\rangle..\mathbf{0} \oplus 0.1\tau.\overline{b}\langle driving\rangle.\mathbf{0} \\
P_{a_yellow} &\triangleq 0.1\tau.\overline{b}\langle braking\rangle..\mathbf{0} \oplus 0.9\tau.\overline{b}\langle driving\rangle.\mathbf{0} \\
P_{a_green} &\triangleq \overline{b}\langle driving\rangle.\mathbf{0}
\end{aligned}$$

Therefore, the aggressive driver is more likely to drive on at red and yellow. We may analyse what is the probability of a crash if two different drivers go through a single

traffic light from different streets. The behave process P_{a_driver} is as the following:

$$P_{behave} \triangleq b(y).([y = braking]\mathbf{0} + [y = stopped]\mathbf{0} + [y = driving]\mathbf{0})$$

6.3.2 Translation of a π_{prob} Model into the PRISM Language

We show that for closed and finite processes (i.e., which do not replicate themselves), the semantics of a probabilistic π-calculus process can be represented by an MDP.

6.3.2.1 Translation rules

We assume that the set of all names in the system is $\mathcal{N}$, which is partitioned into disjoint subsets: $\mathcal{N}^{fn}$, the set of all free names appearing in processes $P_1, P_2, ..., P_i, ..., P_n$, and $\mathcal{N}_1^{bn}, \mathcal{N}_2^{bn}, ..., \mathcal{N}_i^{bn}, ..., \mathcal{N}_n^{bn}$, and $P_1, P_2, ..., P_i, ..., P_n$ (the sets of input-bound names for processes). A match is an equality test on names from $\mathcal{N}$ and a condition M is a finite conjunction of matches, i.e., M is of the form $[x_1 = y_1] \wedge ...[x_n = y_n]$. The translation rules of a π_{prob} model into the PRISM language, defined in [109], can be summarised as follows,

- **Rule 1.** Each of the n sub-processes P_i becomes a PRISM *module* with the same name.
- **Rule 2.** Each element Q_j^i of the finite set of terms $S_i = \{Q_1^i, ..., Q_k^i\}$, which is the set of the states of process P_i after each of its transitions (In [109], the set of all these states is called the PSTG of P_i), becomes an integer variable s_i whose values vary from 1 to k.
- **Rule 3.** Module P_i has $|\mathcal{N}_i^{bn}| + 1$ local variables. Each bound name x_j^i of process P_i has a corresponding variable x_j^i with range $0, ..., |\mathcal{N}^{fn}|$ and it is initialised to 0.
- **Rule 4.** The model includes $|\mathcal{N}^{fn}|$ integer constants, one for each free name, which are assigned distinct, consecutive non-zero values. If the value of variable x_j^i is equal to one of these constants, then the corresponding bound name has been assigned the appropriate free name (by an input action). On the contrary, $x_j^i = 0$ means that no input to the bound name has occurred yet.
- **Rule 5 (Probabilistic internal transition).** For $Q_i \xrightarrow{M,\tau} \{|p_1 : R_1^i, ..., p_m : R_m^i|\}$, which a transition, we add the command:

$$[]\ (s_i = Q_i)\ \&\ M \rightarrow p_1 : (s_1' = R_1^i) + ... + p_m : (s_i' = R_m^i).$$

- **Rule 6 (Output on free name).** Process P_i outputs y on free name x to P_j. For a transition $Q_i \xrightarrow{M,\bar{x}\langle y\rangle} R_i$, where $x \in \mathcal{N}^{fn}$, we add, for each $j \in \{1,...,n\}\backslash\{i\}$, the command:

$$[x\_P_i\_P_j\_y]\ (s_i = Q_i)\ \&\ M\ \rightarrow (s_i' = R_i).$$

  The channel x, sender P_i, receiver P_j, and sent name y are all encoded in the action label. See [109] for details.

- **Rule 7 (Output on bound name).** Process P_i outputs y on bound name x to P_j. For a transition $Q_i \xrightarrow{M,\bar{x}\langle y\rangle} R_i$, where $x \in \mathcal{N}_i^{bn}$, we add, for each $a \in \mathcal{N}^{fn}$ and $j \in \{1,...,n\}\backslash\{i\}$, the command:

$$[a\_P_i\_P_j\_y]\ (s_i = Q_i)\ \&\ M\ \&\ (x = a) \rightarrow (s_i' = R_i).$$

  This is similar to Rule 6 except that it includes a command for each possible value a of x.

- **Rule 8 (Input on free name).** Process P_j inputs z on free name x from P_i. For a transition $Q_i \xrightarrow{M,x(z)} R_i$, where $x \in \mathcal{N}^{fn}$, we add, for each $y \in \mathcal{N}\backslash\mathcal{N}_i^{bn}$ and $j \in \{1,...,n\}\backslash\{i\}$, the command:

$$[x\_P_j\_P_i\_y]\ (s_i = Q_i)\ \&\ M \rightarrow (s_i' = R_i)\ \&\ (z' = y).$$

  For input actions, an extra assignment $(z' = y)$ is added to consider each possible received name y. It models the update of the bound name z to y.

- **Rule 9 (Input on bound name).** Process P_j inputs z on bound name x from P_i. For a transition $Q_i \xrightarrow{M,x(z)} R_i$, where $x \in \mathcal{N}_i^{bn}$, we add, for each $a \in \mathcal{N}^{fn}$, $y \in \mathcal{N}\backslash\mathcal{N}_i^{bn}$ and $j \in \{1,...,n\}\backslash\{i\}$, the command:

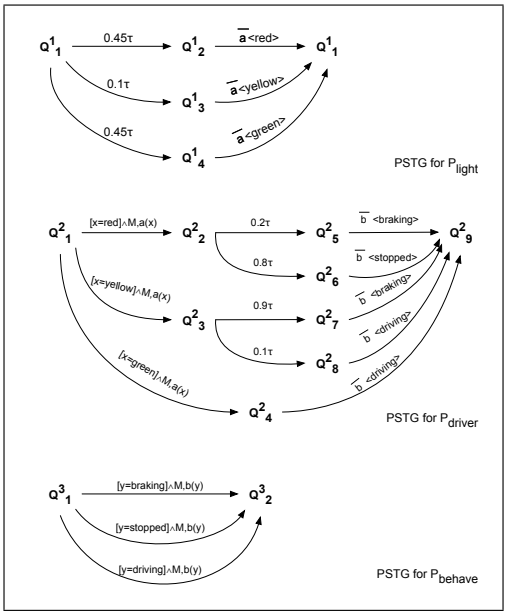
$$[a\_P_j\_P_i\_y]\ (s_i = Q_i)\ \&\ M\ \&\ (x = a) \rightarrow (s_i' = R_i)\ \&\ (z' = y).$$

  This rule combines elements of Rules 8 and 9, since a command is added to consider each possible pairing of channel a that x may represent and name y that may be received.

In addition, some incorrect commands added in Rules 8 and 9 need to be removed. This is due to that these commands correspond to input actions which will never occur. Particularly, the action label $x\_P_i\_P_j\_y$ appears on a command of each module P_j, but does not appear in any of the commands in module P_i. Therefore, according to [109], commands with such action labels are removed from P_j.

6.3.2.2 Translation of the example

In Figure 6.7, we show an example translation for the traffic light example in Section 3.3.2. The intermediate PTSG is illustrated in Figure 6.7 (a), and the PRISM model is shown in Figure 6.7 (b).



(a) PTSGs

```
mdp

const int red = 1; const int yellow = 2; const int green = 3;
const int braking = 4; const int stopped = 5; const int driving = 6;
const int a = 7; const int b = 8;

module Light
        s1 : [1..4] init 1;
        [] (s1=1) -> 0.45 : (s1'=2) + 0.1 : (s1'=3) + 0.45 : (s1'=4);
        [a_Light_Driver_red] (s1=2) -> (s1'=1);
        [a_Light_Driver_yellow] (s1=3) -> (s1'=1);
        [a_Light_Driver_green] (s1=4) -> (s1'=1);
endmodule

module Driver
        s2 : [1..9] init 1;
        x : [0..8] init 0;
        [a_Light_Driver_red] (s2=1) -> (s2'=2) & (x'=red);
        [a_Light_Driver_yellow] (s2=1) -> (s2'=3) & (x'=yellow);
        [a_Light_Driver_green] (s2=1) -> (s2'=4) & (x'=green);
        [] (s2=2) -> 0.2 : (s2'=5) + 0.8 : (s2'=6);
        [] (s2=3) -> 0.9 : (s2'=7) + 0.1 : (s2'=8);
        [b_Driver_Behave_braking] (s2=5) -> (s2'=1);
        [b_Driver_Behave_stopped] (s2=6) -> (s2'=1);
        [b_Driver_Behave_braking] (s2=7) -> (s2'=1);
        [b_Driver_Behave_driving] (s2=8) -> (s2'=1);
        [b_Driver_Behave_driving] (s2=4) -> (s2'=1);
endmodule

module Behave
        s3 : [1..2] init 1;
        y : [0..8] init 0;
        [b_Driver_Behave_braking] (s3=1) -> (s3'=2) & (y'=braking);
        [b_Driver_Behave_stopped] (s3=1) -> (s3'=2) & (y'=stopped);
        [b_Driver_Behave_driving] (s3=1) -> (s3'=2) & (y'=driving);
endmodule
```

(b) PRISM code

Figure 6.7: Model transformation example of traffic light and a cautious driver.

6.4 Specification

6.4.1 The π_{prob} Models

The formal models of the system consist of 12 π_{prob} processes (P_A, P_B, P_C, P_D, P_E, P_F, P_G, MS, MCS, GA, $User$, $Switch$) for different subsystems: each of 7 processes for each of 7 satellites, 1 process for the monitor station, 1 process for the master control station, 1 process for the ground antenna, 1 process for the user, and 1 process for the mobility model. There are also 5 types of channels: a, b, c, d, e, which are used for transmitting messages between subsystems. Table 6.2 below provides the meanings of main variables used in the model.

Table 6.2: Meanings of variables used in the model.

Variables	Meaning
r_c	reliability of transmission from satellite C to the monitor station
m_j	the information sent by a satellite to the monitor station
r_{ms}	reliability of transmission from the monitor to the master control station
p_{ef}	probability of the message is corrupted
v_j	the message has been verified
n_j	the message is corrupted

6.4.1.1 The π_{prob} Models of the Space Segment

The model of the space segment consists of 7 π_{prob} processes of 7 satellites, referred to P_A, P_B, P_C, P_D, P_E, P_F and P_G. These satellites receive information from the ground antenna simultaneously and then transmit the navigation information to the user via the monitor station. In this chapter, the user and the monitor station are assumed to receive navigation signals from the satellites simultaneously.

There are 3 types of channels for each of the 7 satellites, in which d_i (where $i \in \{1,2,...,7\}$) is the channel between the ground antenna and each individual satellite j (where $j \in \{A,B,C,D,E,F,G\}$ for all following denotations), e_i is the channel between the satellite and the aircraft, and a_i the channel between the satellite and the monitor station. The π_{prob} model of satellite C, D, and E of the space segment are given as below, and the π_{prob} models of other satellites can be derived similarly.

$$
\begin{aligned}
P_C &\triangleq r_c\tau.\overline{a_3}\langle m_c\rangle.d_3(x_c).([x_c = v_c]P_C' + [x_c = no]P_C) \oplus (1-r_c)\tau.P_C \\
P_C' &\triangleq r_c\tau.\overline{e_3}\langle m_c\rangle.out_c(y_c).\mathbf{0} \oplus (1-r_c)\tau.P_C' \\
P_D &\triangleq r_d\tau.\overline{a_4}\langle m_d\rangle.d_4(x_d).([x_d = v_d]P_D' + [x_d = no]P_D) \oplus (1-r_d)\tau.P_D \\
P_D' &\triangleq r_d\tau.\overline{e_4}\langle m_d\rangle.\mathbf{0} \oplus (1-r_d)\tau.P_D' \\
P_E &\triangleq in_e(y_e).P_E' \\
P_E' &\triangleq r_e\tau.\overline{a_5}\langle m_e\rangle.d_5(x_e).([x_e = v_e]E'' + [x_e = no]P_E') \oplus (1-r_e)\tau.P_E' \\
P_E'' &\triangleq r_e\tau.\overline{y_e}\langle m_e\rangle.\mathbf{0} \oplus (1-r_e)\tau.P_E''
\end{aligned}
$$

In the above, r_j denotes the reliability of transmission from the corresponding satellite to the monitor station, which is represented by reliability (probability) as shown in Table 6.1. For communication, m_j is the information sent by a satellite to the monitor station via channel a_i. Afterwards this message is relayed to the master control station and verified there, so, v_j denotes that the message has been verified, otherwise it will

be *no* for the message that has been corrupted due to the influence of environmental factors. The satellite then sends the verified message to the aircraft via channel e_i for the purpose of continuous navigation.

6.4.1.2 The π_{prob} Models of the Control Segment

Here, navigation information mainly refers to the data of the on-orbit state of satellites that are transmitted by the navigation satellites. Satellites in this system do not exchange information with one another. In the real world, all GPS satellites are monitored by a set of 6 monitor stations. In this chapter, we make the simplifying assumption that there is a single monitor station, which is essentially a combination of the 6 stations. As a result, each satellite transmits information to the monitor station independently and simultaneously. The π_{prob} model of the monitor stations is MS.

$$\begin{aligned} MS &\triangleq a_1(x).MS_1 + a_2(x).MS_2 + a_3(x).MS_3 + a_4(x).MS_4 \\ &\quad + a_5(x).MS_5 + a_6(x).MS_6 + a_7(x).MS_7 \\ MS_i &\triangleq r_{ms}\tau.\overline{b}\langle x\rangle.MS \oplus (1 - r_{ms})\tau.MS_i \ (1 \leq i \leq 7) \end{aligned}$$

In the above, MS_i denotes the processes for communication between satellites A, B, C, D, E, F, and G and the monitor station respectively. The direct summation $+$ is used due to the fact that in our assumption the single monitor station (MS) is unable receive simultaneous transmissions, so there will be a nondeterministic choice between simultaneous transmissions from different satellites to the monitor station. Then, r_{ms} denotes reliability of transmission from the monitor station to the master control station, which is a probability ($r_{ms} = 0.99999$ as default) as shown in Table 6.5. For communication, x is the message received from the satellite and relayed to the master control station via channel b.

The master control station receives information from the monitor station via channel b, then transmits it to the ground antenna via channel c. Further, MCS_i represents the sub-process for verifying the relayed message from a satellite via the monitor station. Its π_{prob} model is MCS, defined as the following process:

$$\begin{aligned} MCS &\triangleq b(x).([x = m_a]MCS_1 + [x = m_b]MCS_2 + [x = m_c]MCS_3 \\ &\quad + [x = m_d]MCS_4 + [x = m_e]MCS_5 + [x = m_f]MCS_6 + [x = m_g]MCS_7) \\ MCS_i &\triangleq r_{mcs} \cdot p_{ef}\tau.\overline{c}\langle v_j\rangle.MCS \oplus r_{mcs} \cdot (1 - p_{ef})\tau.\overline{c}\langle n_j\rangle.MCS \oplus (1 - r_{mcs})\tau.MCS_i \\ &\quad ((i,j) \in \{(1,a),(2,b),(3,c),(4,d),(5,e),(6,f),(7,g)\}) \end{aligned}$$

In the above, r_{mcs} denotes the reliability of transmission from the master control station to the ground antenna, which is a probability ($r_{mcs} = 0.99999$ as default) as

shown in Table 6.5. p_{ef} is the probability of whether the message is corrupted due to the influence of environmental factors. For communication, the master control station sends the verified result back to the corresponding satellite through the ground antenna via channel c. The nondeterministic choice $+$ is used for name matching of messages sent from the monitor station to the master control station.

Similar to the monitor station, the ground antenna communicates with the 7 satellites simultaneously. There are 4 ground antennas worldwide that perform the daily routine of transmitting commands to each satellite. We also make a similar abstraction that we use a single ground antenna instead of the 4 original ground antennas. The π_{prob} model of the ground antenna is GA, defined as the following process:

$$\begin{aligned} GA &\triangleq c(y).([y=v_a]GA_1+[y=n_a]GA_1+[y=v_b]GA_2 \\ &\quad +[y=n_b]GA_2+[y=v_c]GA_3+[y=n_c]GA_3+[y=v_d]GA_4 \\ &\quad +[y=n_d]GA_4+[y=v_e]GA_5+[y=n_e]GA_5+[y=v_f]GA_6 \\ &\quad +[y=n_f]GA_6+[y=v_g]GA_7)+[y=n_g]GA_7 \\ GA_i &\triangleq r_{ga}\tau.\overline{d_i}\langle y\rangle.GA \oplus (1-r_{ga})\tau.GA_i \ (1\leq i\leq 7) \end{aligned}$$

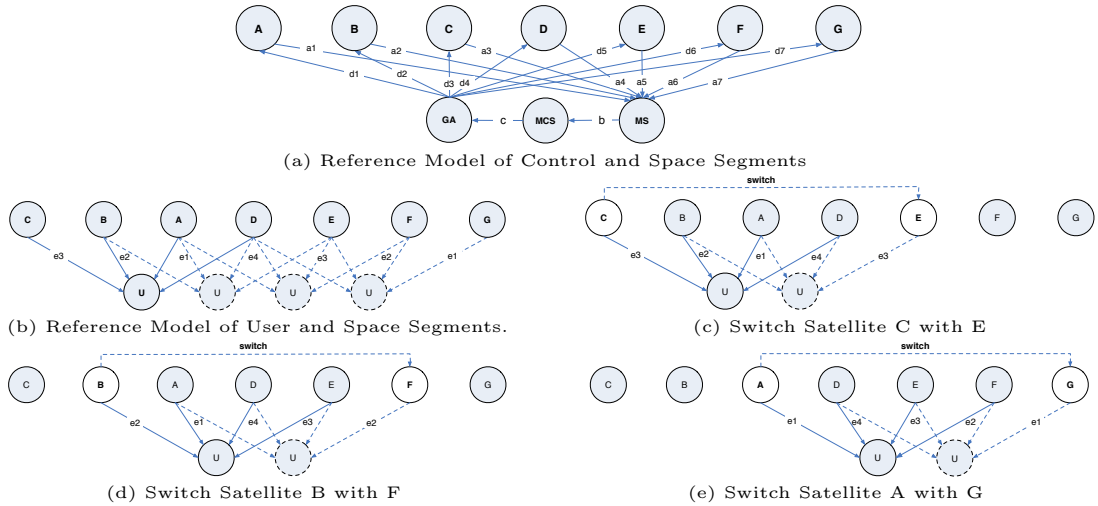
In the above, GA_i denotes the processes for communication between the ground antenna and a satellite. Then, r_{ga} denotes the reliability of transmission from the ground antenna to the corresponding satellite, which is a probability ($r_{ga}=0.99999$ as default) as shown in Table 6.5. For communication, the ground antenna receives the verified result from the master control station via channel c, and then sends the message to the different satellites based on the verified result via channel d_i respectively. Similarly, the nondeterministic choice $+$ is used for name matching of messages sent from the master control station to the ground antenna.

6.4.1.3 The π_{prob} Models of the User Segment

The user segment usually refers to the "GNSS receivers" that capture, process and track L-band signals from visible satellites to calculate the airplane's PVT (see Section 2.3). The navigation mission of the specific flight from Beijing to Guangzhou was used to study the availability of navigation satellites to accomplish the mission during a specific segment of the flight. The 7 satellites were used for navigation during the flight. Due to the coverage limitation of satellites, the aircraft needs to switch to different satellites for navigation guidance during the flight. Figure 6.8 gives the schema of the satellite navigation switching that occurred during the entire flight. As a result, there are 4 satellite groups available for navigation during the entire flight:

$\{A,B,C,D\}$, $\{A,B,D,E\}$, $\{A,D,E,F\}$ and $\{D,E,F,G\}$.

There are two kinds of independent movement: the physical movement of satellites A, B,..., G and the aircraft Usr, and the virtual movement of communication links between them. Their combined physical movement gives rise to the virtual movement of the link between them[2].



(a) Reference Model of Control and Space Segments

(b) Reference Model of User and Space Segments.

(c) Switch Satellite C with E

(d) Switch Satellite B with F

(e) Switch Satellite A with G

Figure 6.8: Reference Model of GNSS Segments.

For mobility models, switching occurred between satellite pairs: C and E, B and F, and A and G. The switch from C to E occurs at 13:15, as shown in Figure 6.8 (c). The switch from B to F occurs at 13:55, as shown in Figure 6.8 (d). The switch from A to G occurs at 14:29, as shown in Figure 6.8 (e). In Figure 6.8 (c), the airplane sequentially uses satellite groups $\{A,B,C,D\}$ and $\{A,B,D,E\}$ for navigation. First, the aircraft uses satellites C, B, A and D; the linking channels between these 4 satellites and the airplane are $e1$, $e2$, $e3$ and $e4$. When the aircraft uses satellites B, A, D and E for navigation, E replaces C at the last stage and the channel of C is replaced by that of E. Figure 6.8 (d) shows the scenario when the aircraft changes from using satellite group $\{A,B,D,E\}$ to group $\{A,D,E,F\}$, and in Figure 6.8 (e), the aircraft changes from using satellite group $\{A,D,E,F\}$ to group $\{D,E,F,G\}$. Similarly, when satellites $\{A,D,E,F\}$ or $\{D,E,F,G\}$ are used.

$$
\begin{aligned}
Usr \quad &\triangleq \quad e_1(z).Usr + e_2(z).Usr + e_3(z).Usr + e_4(z).Usr \\
&\quad\quad + e_5(z).Usr + e_6(z).Usr + e_7(z).Usr \\
Switch \quad &\triangleq \quad \overline{out_c}\langle e_3\rangle.\overline{in_e}\langle e_3\rangle.\overline{out_b}\langle e_2\rangle.\overline{in_f}\langle e_2\rangle.\overline{out_a}\langle e_1\rangle.\overline{in_a}\langle e_1\rangle.\mathbf{0}
\end{aligned}
$$

[2]The links and their movement are obtained using the modelling, simulation, analysis, and operations software Satellite Tool Kit (STK).

6.4.2 Translation from π_{prob} Models to PRISM Language

The π_{prob} processes must be translated to PRISM in order to perform probabilistic verification using the model checker. Translation from π_{prob} models of the satellite navigation system to their representation in PRISM follows the translation rules given in Section 3.3.1. We use the process P_A of satellite A to illustrate the procedure of the translation. The π_{prob} model of the communication between satellite A and the monitor station is:

$$\begin{aligned} P_A &\triangleq r_a\tau.\overline{a_1}\langle m_a\rangle.d_1(x_a).([x_a = v_a]P_A' + [x_a = no]P_A) \oplus (1-r_a)\tau.P_A \\ P_A' &\triangleq r_a\tau.\overline{e_1}\langle m_a\rangle.out_a(y_a).\mathbf{0} \oplus (1-r_a)\tau.P_A' \end{aligned}$$

Then, the process is converted into a graphical representation, namely a PSTG. For comparison, the converted PSTGs of processes P_A and P_E are both shown in Figure 6.9.

Finally, the PSTG of the system is translated into the PRISM modules according to the transition rules, and the corresponding module of π_{prob} model of satellite A can be derived, as shown in Figure 6.10.

The translation of π_{prob} models of the remaining 6 satellites, the monitor station, the master control station, the ground antenna, the aircraft, and the mobility model can be derived similarly using the translation rules. The entire PRISM code is shown in Appendix.

We built a small, but detailed, model of the system. The satellite navigation systems exhibit both probabilistic behaviour (re-transmission due to unreliability of space segment and control segment) and nondeterministic behaviour (scheduling of transmission of satellites by control segment within the mission) and can be naturally modelled as an MDP. We translated a constructed process algebraic model based on the underlying reference model in PRISM. The state space for different number of satellites is shown in Table 6.3.

Table 6.3: State space for different number of satellites.

N	States	Transitions	Time for constructing the MDP (s)
4	153,824	750,368	1.999
5	331,120	1,625,059	9.074
6	501,290	2,466,627	18.153
7	659,252	3,249,969	33.051
8	724,230	3,554,991	61.741

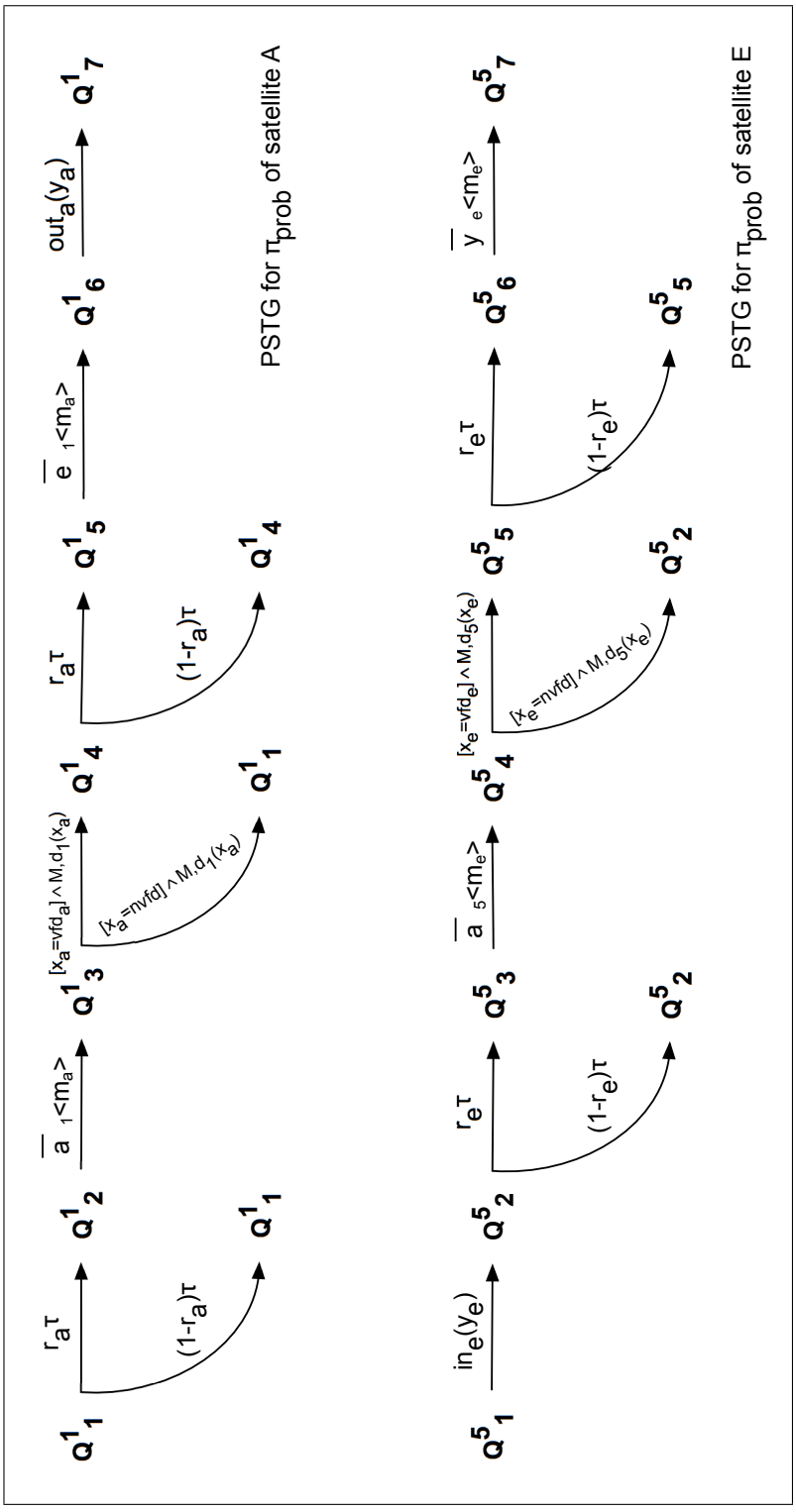


Figure 6.9: PTSGs of π_{prob} process of satellites A and E.

Because of the detailed nature of the model and the corresponding state space size, we first consider a small number of satellites (N=4, 5, 6, 7, 8). It is possible, though,

```
module SA     // module for satellite A
        sa : [1..7] init 1;
        xa : [0..18] init 0;
        [] (sa=1) -> ra : (sa'=2) + (1-ra) : (sa'=1);
        [a1_SA_MS_ma] (sa=2) -> (sa'=3);
        [d1_GA_SA_z] (sa=3) & (z=va) -> (sa'=4) & (xa'=z);
        [d1_GA_SA_z] (sa=3) & (z!=va) -> (sa'=1) & (xa'=z);
        [] (sa=4) -> ra : (sa'=5) + (1-ra) : (sa'=4);
        [e1_SA_Usr_ma] (sa=5) -> (sa'=6);
        [outa_S_SA_e1] (sa=6) -> (sa'=7);
endmodule
```

Figure 6.10: The PRISM module of satellite A.

that availability properties analysed in these small models will also be exhibited by a more large size (e.g., 17 satellites). With regards to the initial configuration of the model, we assume that the control segment and the user segment communicate with one satellite at a time. This configuration is suitably realistic and ensures that the mobility is possible. For N=7, the model has 659,252 states and 3,249,969 transitions; for N=8, it has 724,230 states and 3,554,991 transitions.

These MDPs are constructed by PRISM on a 2.4GHz Mac with 8GB RAM in 33.051 seconds and 61.741 seconds respectively. Note that the increase of states and transitions between N=7 and N=8 is much lower than the other increases. This is because the flight navigation mission typically requires a minimum of 7 satellites, and the 8th satellite is only used as backup satellite, so there is much less communication between the satellite and the other segments, reducing the transitions accordingly.

6.5 Verification

6.5.1 Availability Parameters

During signal transmission from the monitor station to the master control station and from the master control station to the ground antenna, abnormal signal transmission may occur, resulting in errors in information and corresponding anomalies in the subsequent update information for the satellites. This can affect the navigation safety of users if the situation is severe. If anomalies occur in signal transmission, the master control station can correct the signal after a certain period of time. The reliability of

space and control segments based on MTBF and MTTR is given in Table 6.4.

Table 6.4: Reliability of space and control segments.

Systems	MTBF (hours)	MTTR
Satellite	model	6 months
Monitor Station	156000	25.2 minutes
Master Control Station	1248	52.3 minutes
Ground Antenna	2310	4.2 hours

Furthermore, we propose a modified concept for the GNSS availability properties associated with the underlying specification. The current approach involves prediction of the "mean" availability over the system lifetime, assuming that the system is in a steady state. This approach is not suited to the specification of GNSS positioning systems, where the objective is to guarantee what can be obtained from the system during short periods of time that are meaningful to users, and that this short term availability will be maintained during the lifetime of the system. This requires a modification of the availability concept, as it is currently understood.

Based on a preliminary investigation, it is assumed in our analysis that the information exchange among the satellites, monitor station and ground antenna does not itself generate information anomalies, but its reliability is a direct consequence of the reliabilities of the satellites and ground antenna. It is additionally assumed that information anomalies can occur in the signal transmission between satellites, master control station, monitor station, and ground antenna, and environmental factors as well. These assumptions and related data are based on relevant reports[3] on GPS, as summarised in Table 6.5.

Where available, the data used for quantitative analysis in this study were collected from the official published data [129, 65]. In other cases we used data for similar systems. The satellite models involved in the navigation satellite availability analysis of this section are Block-IIA, Block-IIR and Block-IIRM.

6.5.2 Best and Worst Case Availability

In this section, we perform quantitative analysis of satellite availability and channel availability of the satellite navigation system using PRISM respectively. Some typical

[3]Global Positioning System (GPS) Performance Quarterly Report

Table 6.5: Transmission reliability of satellite navigation systems.

Systems	Transmission reliability
Satellite→MS	reliability of satellites
MS→MCS	0.99999
MCS→GA	0.99999
GA→Satellite	0.99999
environmental factors	0.9

examples of availability properties formalised with PCTL are given in Table 6.6 and Table 6.7.

Table 6.6: Summary of PRISM properties used in the chapter (Part I).

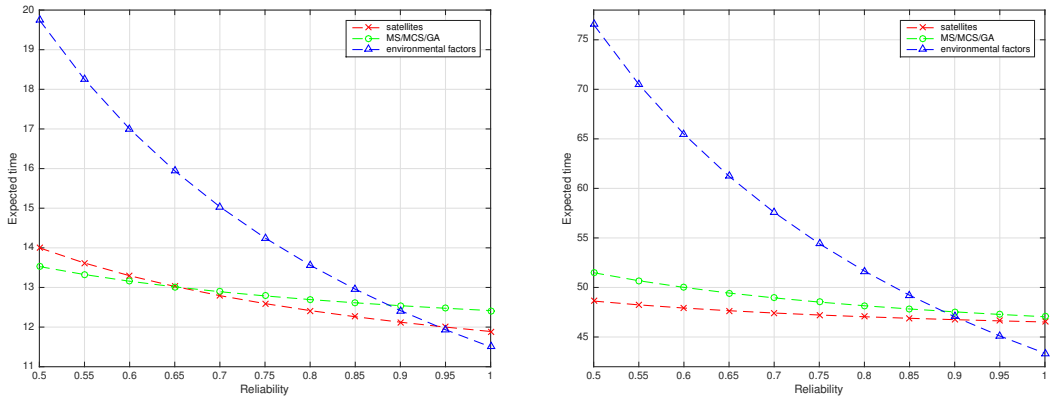
No.	Name	PRISM notation
1	"*available satellite*"	$\mathbf{P}_{min\geq 1}[F\ (sc=7)]$
2	"*minimum available satellite*"	$\mathbf{R}_{min=?}[F\ (sc=6)]$
3	"*maximum available satellite*"	$\mathbf{R}_{max=?}[F\ (s4=4)]$
4	"*minimum unavailable channel*"	$\mathbf{R}_{min=?}[F\ (s5=3)]-\mathbf{R}_{min=?}[F\ (s5=2)]$
5	"*maximum unavailable channel*"	$\mathbf{R}_{max=?}[F\ (s5=6)]-\mathbf{R}_{max=?}[F\ (s5=5)]$
6	"*minimum available time bound satellite*"	$\mathbf{P}_{min=?}[F\leq T\ (sc=6)]$
7	"*maximum available time bound satellite*"	$\mathbf{P}_{max=?}[F\leq T\ (se=7)]$

Table 6.7: Summary of PRISM properties used in the chapter (Part II).

No.	Meaning
1	Whether satellite C is available during the navigation?
2	The minimum available time of satellite C
3	The maximum expected time of navigation mission
4	The minimum unavailable time of channel e3
5	The maximum unavailable time of channel e1
6	The min. probability that C done transmission with U within T
7	The minimum probability that E done transmission with U within T

We first study how the longest expected time of satellite navigation mission for air-

craft varies over the execution of the mission. To consider the probability of some behaviour of our MDP, the nondeterministic choices need to be resolved first. PRISM provides us an exhaustive search and exact quantitative results of all possible behaviours of the system, including both best case and worst case scenarios. This is done using PCTL properties of the form: $\mathbf{R}_{\{"time"\}min=?}[F\ (s4 = 4)]$ and $\mathbf{R}_{\{"time"\}max=?}[F\ (s4 = 4)]$, which represent the minimum (best case) and maximum (worst case) expected value of time that is from the beginning until the end of the mission (at the time instant $F\ (s4 = 4)$). Since we have added a reward structure called "time" to the PRISM model, it associates with each state of the MDP a value representing the longest expected time between any two components at that point. The obtained result of the minimum and maximum expected time that depends on reliability of different components is depicted in Figure 6.11.



(a) Minimum expected time

(b) Maximum expected time

Figure 6.11: Expected time results for different reliability of components.

We see that as reliability increases, the expected time decreases. In Figure 6.11(a) for best case scenario, if the reliability of satellites is larger than 0.65, it will have less influence for satellites than different components of control segment (MS, MCS, and GA) on finishing the navigation mission. However, it is clear that the environmental factor has greatest influence on the total execution time. The reliability of environment is to what extent the environmental factors can jeopardise the transmission reliability between satellites and the control segment, so higher reliability means more reliable transmission. The default value of environmental factors is considered to be 0.9, but we can see that it has less influence on mission time when it is larger than 0.95. So, we should design the course of movement of satellites in the environment as gentle and stable (e.g., less solar radiation) as possible. From Figure 6.11(b), we see that

the curves of satellites and control segment are similar for worst case scenario. But, when the reliability of environment is larger than 0.9 (which is compared with 0.95 in minimum case), the environment influence on mission time can be neglected.

The properties $\mathbf{P}_{min=?}[F \leq T \ (sc = 6)]$ and $\mathbf{P}_{max=?}[F \leq T \ (sc = 6)]$ enable us to compute the minimum and maximum probability that satellite C finishes signal transmission with the aircraft within T time steps. The form of "$\leq t$" or "$< t$" (where t is a PRISM expression evaluating to a constant, non-negative value) is the upper time bound.

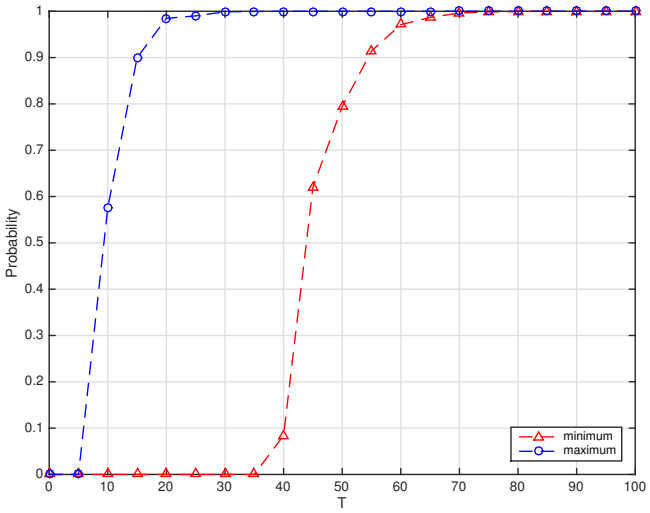
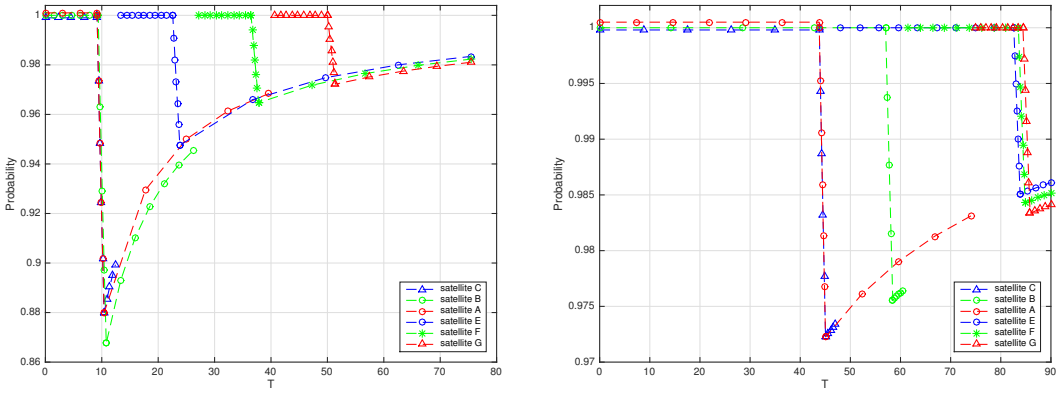


Figure 6.12: Probability of for satellite C within time T.

As shown in Figure 6.12, we have $t = T$ in our case, where T is a constant value between 0 and 100. We see that the probability increases as T increases after the probability equals to 0 and before it reaches to 1. For both cases, the satellite C eventually sent signal to the aircraft. However, for the minimum case, it takes much less time for the probability to reach 0.9 from 0 (about 10 time steps), compared to the maximum case (about 55 time units). We should be aware of that in realistic cases, the probability distribution is between the two of them.

The minimum availability of satellite C varies on time, and it can be derived by the following formula, where $\mathbf{R}_{\{"time"\}min=?}[F \ (sc = 5)] - \mathbf{R}_{\{"time"\}min=?}[F \ (sc = 4)])$ is the unavailable time of satellite C in the minimum case.

$$Avail^{C}_{min}(T) = \begin{cases} 1 & C1 \\ \mathbf{R}_{\{"time"\}min=?}[F \ (sc = 4)]/T & C2 \\ T - (\mathbf{R}_{\{"time"\}min=?}[F \ (sc = 5)] - \mathbf{R}_{\{"time"\}min=?}[F \ (sc = 4)])/T & C3 \end{cases}$$



(a) Minimum availability (b) Maximum availability

Figure 6.13: Minimum and maximum availability of satellites.

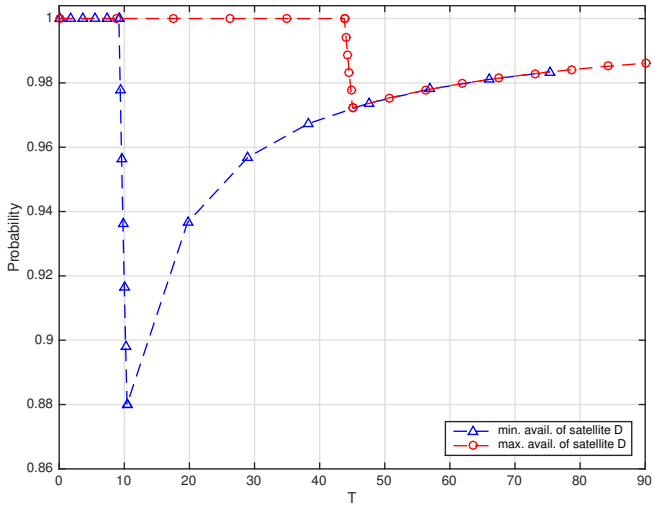


Figure 6.14: Availability of satellite D.

Table 6.8: Conditions (Part I).

Conditions	Meaning
C1	$0 \leq T < \mathbf{R}_{\{''time''\}min=?}[F\ (sc = 4)]$
C2	$\mathbf{R}_{\{''time''\}min=?}[F\ (sc = 4)] \leq T \leq \mathbf{R}_{\{''time''\}min=?}[F\ (sc = 5)]$
C3	$\mathbf{R}_{\{''time''\}min=?}[F\ (sc = 5)] < T \leq \mathbf{R}_{\{''time''\}min=?}[F\ (sc = 7)]$

The maximum availability of satellite C and minimum/maximum availability of all other satellites can be derived similarly. The results are illustrated in Figure 6.13 and

Figure 6.14 respectively.

From above figures, we see that satellites of the same model have the same availability distribution. For instance, both the minimum and maximum probability of satellites A, C, and D are all same, except that their individual online duration are difference. They are online at the same time, but satellite D has a long tail than satellites A and C, and satellite A has longer tail than C. This is due to the fact that satellites D is never offline (never being switched) during the mission later than both A and C, and A gets offline (switching to G) later than C. Model Block IIRM (Satellites A, C, D, and E) had the largest satellite availability for navigation, followed by Block IIR (satellite F) and then Block IIA (satellites B and G). The availability curve indicates that the satellite online and offline time instant and the duration of use of a satellite do not have very significant impact on the satellite's availability. Rather, the factor that had the greatest effect on navigation was the design life and reliability of the navigation satellites.

Similar to the definition of satellite availability, we define channel availability. The minimum availability of channel e3 also varies on time, and it can be derived by the following formula, where $\mathbf{R}_{\{''time''\}min=?}[F\ (s5=3)] - \mathbf{R}_{\{''time''\}min=?}[F\ (s5=2)])$ is the unavailable time of channel $e3$ in the minimum case.

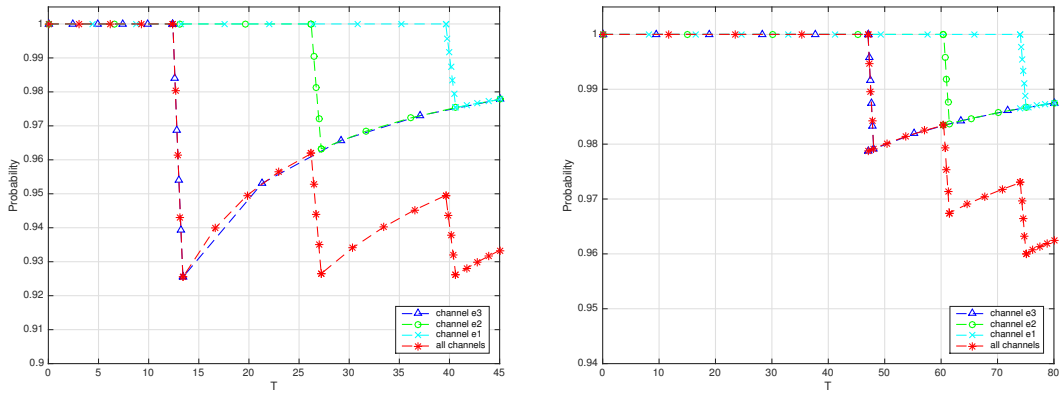
$$Avail^{e_3}_{min}(T) = \begin{cases} 1 & C4 \\ \mathbf{R}_{\{''time''\}min=?}[F\ (s5=2)]/T & C5 \\ T - (\mathbf{R}_{\{''time''\}min=?}[F\ (s5=3)] - \mathbf{R}_{\{''time''\}min=?}[F\ (s5=2)])/T & C6 \end{cases}$$

Table 6.9: Conditions (Part II).

Conditions	Meaning
C4	$0 \leq T < \mathbf{R}_{\{''time''\}min=?}[F\ (s5=2)]$
C5	$\mathbf{R}_{\{''time''\}min=?}[F\ (s5=2)] \leq T \leq \mathbf{R}_{\{''time''\}min=?}[F\ (s5=3)]$
C6	$\mathbf{R}_{\{''time''\}min=?}[F\ (s5=3)] < T \leq \mathbf{R}_{\{''time''\}min=?}[F\ (s5=7)]$

The maximum availability of channel $e3$ and minimum/maximum availability of all other channels can be derived similarly. The results are illustrated in Figure 6.15. The backup satellite of a channel were not considered in this study. Neglecting backup satellite may cause the channel availability to be slightly greater than when it is considered. An actual mission will involve multiple satellites, and each channel has multiple

backup satellites. Thus, once a failure occurs, the channel will be switched to a backup satellite.



(a) Minimum availability (b) Maximum availability

Figure 6.15: Minimum and maximum availability of channels.

Other than using high cost backup satellite for a navigation, a way to improve the channel availability is the addition of new signals. These signals complement the existing signal for navigation service. This additional signal will make GNSS a more robust navigation system for various aviation applications. Thus, the availability of additional signals means that errors that occur in the signals due to disturbances in the ionosphere can be significantly reduced through the simultaneous use of more signals. This will improve the overall system reliability, to increase accuracy and availability, and will allow a robust approach with little or no ground infrastructure.

Therefore, the availability of navigation satellites in the actual process is greater than this value. In general, the impact of environmental factors is small, and thus the availability of satellites for navigation could be larger than 98.5%. Moreover, the presence of multiple satellites will potentially increase the overall availability along an air line, but the increase of available satellites does not necessarily guarantee an improved user-satellites geometry due to the similar orbital arrangement of most GNSS satellites.

The SPS SIS availability is the probability that the slots in the GPS constellation will be occupied by satellites transmitting a trackable and healthy SPS SIS. For this SPS Performance Standard, there are two components of availability as follows: (1) per-slot availability: the fraction of time that a slot in the GPS constellation will be occupied by a satellite that is transmitting a trackable and healthy SPS SIS; (2) constellation availability: the fraction of time that a specified number of slots in the GPS

constellation.

SPS SIS availability is assessed through analysis of the broadcast navigation messages. To evaluate the usefulness of our results for SPS SIS availability, we referred to some official reports from the civil aviation sector. The U.S. Federal Aviation Administration (FAA) releases quarterly reports on the performance analysis of the system based on the operation of the GPS in each quarter to ensure the navigation safety of global aviation. According to the monitoring reports released by the FAA [45] for the period of 1 January 2013 to 31 March 2013, the average service availability of each individual GPS satellite is approximately 99%, and the worst-case service availability is approximately 90%. These numbers approximately are close to those obtained in our study, which are between 88% and 98% for minimum availability and between 97.5% and 98.5% for maximum availability. This supports that, from one line of evidence, the feasibility and applicability of our approach.

6.6 Conclusion and Discussion

In this chapter, we have shown that probabilistic verification can be used to analyse interesting and important reliability and availability properties of GNSS based aircraft guidance systems that would be difficult to discover using alternative analysis techniques such as simulation. We have demonstrated the successful application of probabilistic model checking to the analysis of reliability and availability properties that relate to the dependability and overall performance of the underlying system of satellite navigation for aviation. To do this, we have went through the theory of Markov decision processes (MDPs), process algebras, and probabilistic model checking.

Although using probabilistic model checking limits the number of the satellites in the navigation system that can be analysed, these representative systems can highlight interesting behaviour that may also occur in more realistic configurations for aviation navigation. We have modelled essential aspects (e.g., unreliable signal transmission, component movement, concurrency, nondeterminism) of satellite system for navigating the specific flight (from Beijing to Guangzhou). The results we have obtained demonstrate that modelling unknown choices with randomness causes a variation on the mission execution time and availability: the actual scenario may be different from the best or worst scenarios. The introduction of nondeterminism shows that these measures can take a range of values. As a result, we compute minimum and maximum values, representing both the best case and worst case of mission execution time and

satellite and channel availability under any scheduling of simultaneous transmission between different satellites and control segment.

Although nowadays satellite positioning is commonly used in the aviation sector, it is still to gain a foothold in other industries such as the rail industry. Up to now availability analysis is non-trivial because difficult situations exist on the railways due to the limitations of the GNSS coverage in urban canyons, tunnels, and forest areas. For future work, we plan to add a fourth environment segment that simulates such difficult situations to the GNSS.

Chapter 7

Safety Analysis of Path Planning for Satellite Surveillance

This chapter is based on the paper [114]. We use probabilistic model checking for analysing and verifying satellite surveillance missions. Whereas in Chapter 4 we used CTMCs model to analyse the reliability, availability, and maintainability of the standalone space segment, in Chapter 5 we consider satellite subsystems failures using CTMCs, and in Chapter 6 we model satellite navigation for aviation missions using MDPs. In this chapter, we use PTAs to model the real-time behaviours of the space segment such as a single satellite, and perform safety analysis based on the behaviour models.

In this chapter, we consider three essential parameters have to be taken into consideration simultaneously in path planning, which are: (1) the minimum path following time (2) the maximum observation coverage time, and (3) the minimum path following fuel cost, respectively. We model both classical global and local path planning problems in a single satellite settings in the formalism of probabilistic timed automata, and verify them with some quantitative properties about these parameters using PRISM.

Furthermore, the aim of this chapter is to formally verify three local path planning behaviours (orbit change, high speed, and low speed) that involve a mobile satellite and an aircraft. The paths of the satellite and aircraft cross each other, and their movements have both probabilistic and real-time properties, which is very suitable for using probabilistic timed automata. Thus, the probabilistic models and the logical formulae are first built, and then PRISM is applied again to verify the underlying three strategies.

7.1 Introduction

The operation of a satellite mission of surveillance of an emergency often involves several different phases that must be accomplished in sequence meanwhile continuously. As a result, the whole satellite system involved in surveillance mission can be referred as a phased mission system. During each mission phase, the satellite has to accomplish a specified task and may be subject to different RAMS requirements. Therefore, satellite configuration and component failure behaviour may change from phase to phase [142]. When a satellite follows its whole path, the dynamic behaviour of the satellite usually requires a distinct but continuous-time model for each phase of the mission. This continuous-time model can be used to analyse RAMS properties of the mission.

"On May 12, 2008, an earthquake measuring 8.0 on the Richter scale occurred at Wenchuan City, Sichuan Province, China. It was confirmed that 69,197 people were dead, and 374,176 injured, with 18,222 listed as missing. In this emergency, a significant amount of observation information and pictures about the disaster area were urgently needed. Sufficiently reliable and available satellite surveillance would ensure the successful accomplishment of the rescue and aid mission. In general, path with repeated ground tracks are selected in the design of path planning for the surveillance mission. These paths proved to have better partial coverage properties than those with unrepeated ground tracks" [146]. In general, only the repetition periods of the repeated paths are required in the design and analysis of satellite planning for surveillance mission.

In case of emergencies such as earthquake, forest fire, or railway collision, it is impossible to observe a given target on earth by immediately launching new satellites. Since there is an urgent need for efficient satellite scheduling within a limited time period, existing satellites have to be effectively and efficiently employed to rapidly and continuously cover and monitor the affected area during a short time period.

The main purpose of satellite's path planning for an emergency surveillance is to complete each phase of the mission, and each phase may be very different. In particular, in a phase of the mission, the environment of the mission is also different from the other phases, which leads to the reliability and safety of the satellite in this task is different from others. In order to adapt to the environment and complete the task, the satellite needs to change its configuration according to the actual situation to increase the reliability to complete the task.

The addition to these dynamics and dependencies, phased missions pose unique

challenges to existing analysis techniques. In traditional approaches, the occurrence of the sequence of events has been determined in accordance with advanced planning, but in the actual operation, the different results of events will cause different events, and the events caused may be uncertain to designers. It depends on the specific situation. Another problem is the uncertainty of the time at which an event occurs. Occurrence time of the event is not a fixed value, on the other hand, it may be an interval. As a result, the event will take place at any time in the internal.

In this chapter, we propose a formal analysis technique based on probabilistic timed automata (PTAs), utilising PRISM to evaluate the quantitative properties of the satellite path planning problem. Emphasis will be placed on the uncertainty and real-time natures of satellite global and local path planning during the whole mission. Moreover, the uncertainty includes two aspects: the uncertainty of events sequence, and uncertainty of the time at which events occur.

The rest of this chapter is organised as follows. In Section 7.2, we present the related work on path planning approaches for satellites. In Section 7.3 we introduce satellite surveillance missions. We consider the global path path problem for a single satellite in a surveillance mission of a earthquake in Section 7.4, while in Section 7.5 the local path planning problem and three behaviours for a single satellite is modelled and analysed. Finally, we conclude in Section 7.6.

7.2 Related Work

This formal technique is able to solve phased missions problems that can not be solved using other analysis techniques. In recent years, formal verification has already been used to verify the global path planning problem for autonomous vehicles ([122, 46]). However, there is little work that applied in verifying the local path planning problem in the same domain.

In recent years, several researchers have studied optimal path planning approaches for satellite systems for surveillance missions. In [1], the authors employ a genetic algorithm to solve several path planning problems that are characterised by many local minima, such as space surveillance of a few specified sites. Two types of constraint conditions are considered: (1) the maximum resolution of each observation point with a given imaging sensor; (2) the maximum observation time in the total flight time. Satellite constellations in circular paths are widely used in many applications, and a number of path planning scenarios for satellite constellations have been studied for

local and global coverage of the Earth's surface [138, 82, 28].

A novel approximation approach is developed in [135] for analysing the coverage locations of real time communication systems based on low-earth-orbit (LEO) satellite constellations for path planning. The geometric analytic technique is used to process the statistical parameters of the coverage locations of LEO satellite constellations. Ulybyshev [136] further extends the simple coverage case to a more complex situation associated with full or partial visibility of a geographic area by a satellite from the constellation and proposes an approach. It aims to aid path planning of satellite constellations that search for the solution only in the two-dimensional space missions by combining maps of satellite constellations and the coverage requirements.

Configurations of satellite constellations is studied in [121], aiming to maximise the availability of a specific site and satisfy different mission requirements. A novel approach is developed in order to optimise the constellation configurations of LEO satellites for local observation. The approach is able to satisfy 4 requirements: (1) to minimise the maximum gap (MGap); (2) to maximise the maximum coverage (MCov); (3) to minimise the MGap with a lower bound on the MCov; (4) to maximise the MCov with an upper bound on the MGap.

In [146], the problem of path planning for realising optimal disaster rescue and aid are considered. It considers sun-synchronous path and analyses the results of different parameters in the path planning. The optimal fuel cost is obtained by comparing the primer vector theory with Hohmann transfer, and utilised the results to plan paths for multiple satellites.

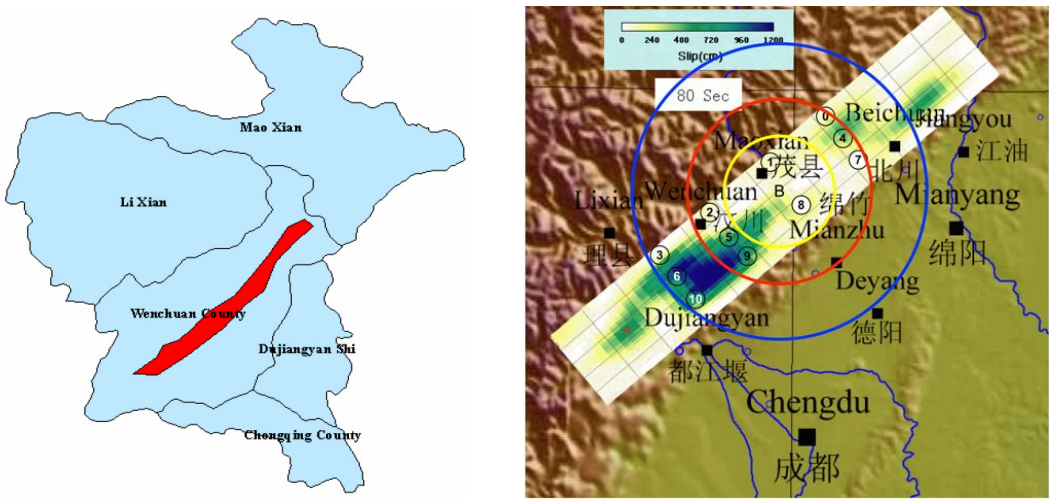
7.3 Surveillance Missions

"The paths of space remote sensing platforms usually control the ground resolution, area coverage, and the frequency of coverage parameters. The path altitude affects the resolution and the swath width [1]. The higher the spacecraft moves, the wider the affected area covered and correspondingly the lower the resolution. Although lower altitudes enable a satellite to get higher resolution, path perturbations due to atmospheric drag should not be overlooked. Ground surveillance can be viewed as the observation of multiple discrete locations (sites) on the surface of the earth. These kinds of missions require that the satellite visits all of the given sites within a given period" [146].

[1] Swath width: the strip of the Earth's surface from which geographic data are collected by a satellite in the course of swath mapping.

Here we consider a single satellite from the China's Beidou satellite system [99]. This set of satellites is Chinese orbiting satellites, which is mainly used for space exploration for science and technology, earth observation, and weather forecasting. These satellites can be scheduled among the low earth orbits, and the range of the semi-major axis is from 6951 to 7182 km. Emergency satellites prove extremely helpful in the Wenchuan earthquake. Satellite terminals were carried by the military units entering the area, providing a desperately needed communications and image capability, in addition to navigation and positioning support. The Satellite Navigation and Positioning Main Station provided 24-hour emergency support during the Wenchuan earthquake.

Path planning of a single satellite must also consider the satellite movement, from the initial site to the target site during a limited time with minimum fuel cost, which is very critical to the entire mission. Many space scientists have considered the optimisation of path planning of multiple satellites [126, 7, 128]. This problem is approximately considered as "a cooperative rendezvous in which both satellites take an active role in order to further reduce propellant consumption considering the common active and passive cases" [146].



(a) Affected earthquake area (the red area).

(b) Distribution of 11 observation sites

Figure 7.1: Affected area in the Wenchuan earthquake and distribution of observation sites.

In order to save the fuel consumption of movement, the operational path must be close to the given sites on condition that the designed path must meet the mission requirements. Along the path at an optimal altitude, the coverage area of the earth's surface can include the whole region. The entire disaster area can be surveilled in one coverage rectangle as shown in Figure 7.1(a), and we can look on the area as a point

for the purpose of path planning.

Generally, satellite path planning must include "an accurate coverage analysis for fixing an optimal set of parameters, such as the number of observed sites and their spatial distribution, according to the required operational purposes. On the other hand, the motion relative to the Earth's surface creates more difficulties because good numerical techniques for calculating and analysing the characteristics of the path coverage are needed in order to obtain the generalised analytical solutions more easily" [146].

7.4 Global Path Planning for a Single Satellite

In this section, we present an illustrative case study of global path planning for a single satellite, analysed using PTAs and probabilistic model checking.

7.4.1 Formal Models

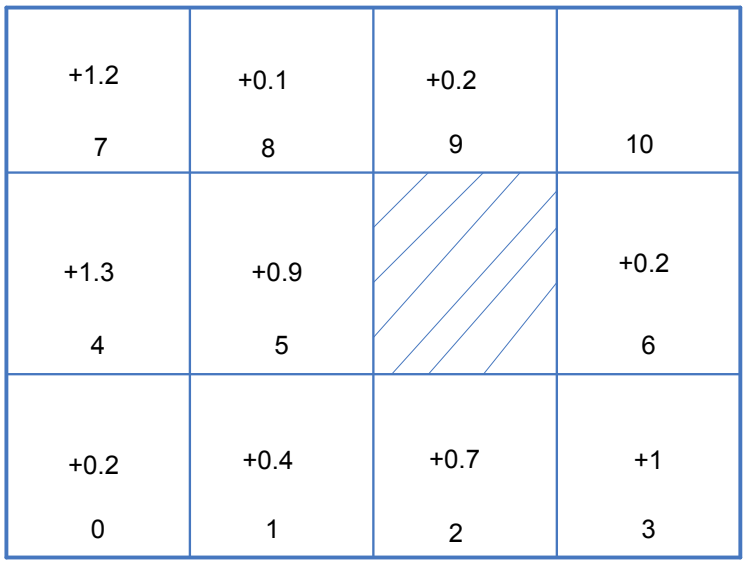


Figure 7.2: Fuel consumption in different area.

We use a case study to illustrate the role of PTAs for the analysis of satellite for surveillance. In this case, a satellite addresses destination by command. The affected area is divided into 12 observation sites, from 0 to 10 (as shown in Figure 7.1(b)). One of the areas is shaded, which denotes a static obstacle area expressing this area cannot be passed by the satellite. The obstacle area is established due to that there maybe a

geostationary satellite[2] in that area.

Assuming a satellite starts to travel from site 0 to site 10, during this process, the satellite operators need to evaluate the condition and environment of the sites, and decide the optimal route to the site 10. Staff control the satellite transfer to the target site along the route. Because of the uncertainty of the control (e.g., solar radiation, human error, etc.) and loss of signal transmission, the satellite does not always transfer to the intended direction. There is a probability of 20% that the satellite will transfer in the wrong direction or remain in the same site. Furthermore, there is a probability of 10% of the deviation to the left site and similarly to the right site.

Table 7.1: Intervals of time-consumption in each area.

Direction	0	1	2	3	4	5	6	7	8	9
East	$3 \sim 5$	$5 \sim 6$	$7 \sim 8$	$3 \sim 5$	$6 \sim 7$	–	–	$7 \sim 8$	$5 \sim 5$	$2 \sim 6$
South	–	–	–	–	$4 \sim 7$	$6 \sim 8$	$3 \sim 5$	$5 \sim 8$	$4 \sim 8$	–
West	–	$4 \sim 6$	$7 \sim 8$	–	–	$5 \sim 8$	–	–	$4 \sim 5$	$4 \sim 6$
North	$2 \sim 5$	$3 \sim 6$	–	$4 \sim 5$	$5 \sim 7$	$6 \sim 8$	$4 \sim 5$	–	–	–

When the satellite does not move according to the planned route, it will have to re-select the path of the route. For example, when the satellite gets the command that it should move ahead to site 4 from site 0, it has 80% probability of completing the mission, 10% probability of remaining in site 0, and 10% probability of moving to site 1.

In this case, there are two other parameters: path following time and path following fuel cost. The time the satellite spends in each observation site transferring to each direction varies. The energy the satellite uses in each site also varies. Figure 7.2 shows the fuel cost in each site. Table 7.1 shows the time spent by the satellite to move between each part of sites.

Figure 7.4 shows a PRISM modelling language description for the PTA models shown in Figure 7.3. It includes a PRISM reward structure, labelled “energy”, to create a priced PTA which assigns different cost rates to states.

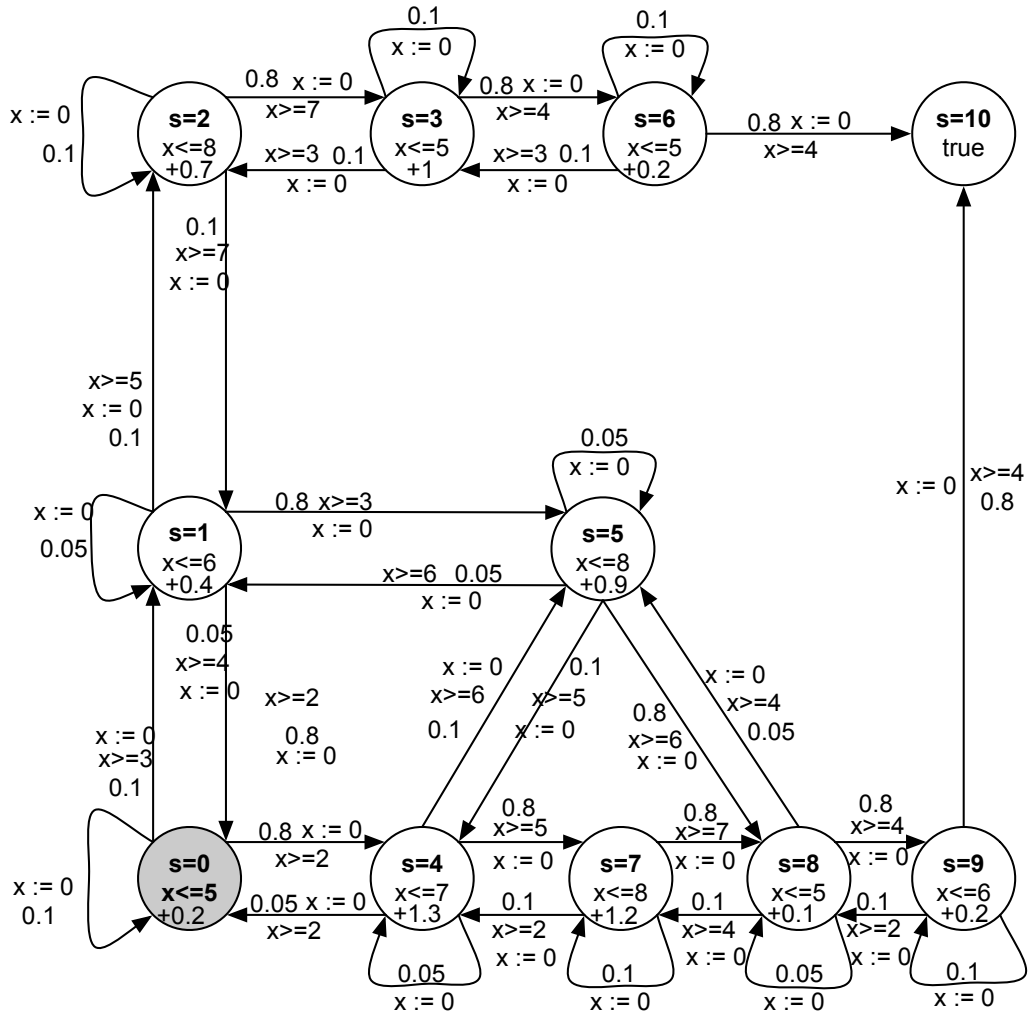


Figure 7.3: Probabilistic timed automata (PTAs) model.

7.4.2 Real Time Verification

Model checking using PRISM shows that the probability of the satellite completing the mission within 30 minutes is 0.84134. The satellite will have 5.184% moving to the area 10 within 15 minutes, and 84.134% within 30 minutes. Figure 7.5 illustrates the relationship between the probability and time to complete the mission.

We models a simple single satellite scheduling in case of the Wenchuan earthquake. The satellite starts in the initial site 0; after between 3 and 5 time units, the satellite attempts to move to site 4:

- with a probability of 0.8, the satellite receives the signal correctly, and starts to

[2]Geostationary satellite: a kind of earth-orbiting satellite that revolves in the same direction and rate the earth rotates.

```
pta
module satellite
        s : [0..10] init 0;
        x : clock;
invariant
        (s=0 => x<=5) & (s=1 => x<=6) & (s=2 => x<=8) & (s=3 => x<=5) &
        (s=4 => x<=7) & (s=5 => x<=8) & (s=6 => x<=5) & (s=7 => x<=8) &
        (s=8 => x<=5) & (s=9 => x<=6) & (s=10 => true)
endinvariant
        [] s=0 & x>=3 -> 0.8:(s'=4)&(x'=0) + 0.1:(s'=0)&(x'=0) +
                         0.1:(s'=1)&(x'=0);
        [] s=4 & x>=6 -> 0.05:(s'=4)&(x'=0) + 0.05:(s'=0)&(x'=0) +
                         0.1:(s'=5)&(x'=0) + 0.8:(s'=7)&(x'=0);
        [] s=1 & x>=5 -> 0.05:(s'=1)&(x'=0) + 0.05:(s'=0)&(x'=0) +
                         0.1:(s'=2)&(x'=0) + 0.8:(s'=5)&(x'=0);
        [] s=2 & x>=7 -> 0.8:(s'=3)&(x'=0) + 0.1:(s'=2)&(x'=0) +
                         0.1:(s'=1)&(x'=0);
        [] s=3 & x>=4 -> 0.8:(s'=6)&(x'=0) + 0.1:(s'=3)&(x'=0) +
                         0.1:(s'=6)&(x'=0);
        [] s=5 & x>=6 -> 0.05:(s'=5)&(x'=0) + 0.05:(s'=1)&(x'=0) +
                         0.1:(s'=4)&(x'=0) + 0.8:(s'=8)&(x'=0);
        [] s=6 & x>=4 -> 0.8:(s'=10)&(x'=0) + 0.1:(s'=6)&(x'=0) +
                         0.1:(s'=3)&(x'=0);
        [] s=7 & x>=7 -> 0.8:(s'=8)&(x'=0) + 0.1:(s'=7)&(x'=0) +
                         0.1:(s'=4)&(x'=0);
        [] s=8 & x>=4 -> 0.05:(s'=5)&(x'=0) + 0.05:(s'=8)&(x'=0) +
                         0.1:(s'=7)&(x'=0) + 0.8:(s'=9)&(x'=0);
        [] s=9 & x>=4 -> 0.8:(s'=10)&(x'=0) + 0.1:(s'=9)&(x'=0) +
                         0.1:(s'=8)&(x'=0);
endmodule
reward ''energy''
        (s=0):0.2; (s=1):0.4; (s=2):0.7; (s=3):1.0; (s=4):1.3;
        (s=5):0.9; (s=6):0.2; (s=7):1.2; (s=8):0.1; (s=9):0.2;
endreward
```

Figure 7.4: The PRISM module for the satellite.

move to site 4, via location 4;

- with a probability of 0.1, data is lost, and the satellite stays in site 0;
- with a probability of 0.1, the satellite receives the signal incorrectly, and starts to

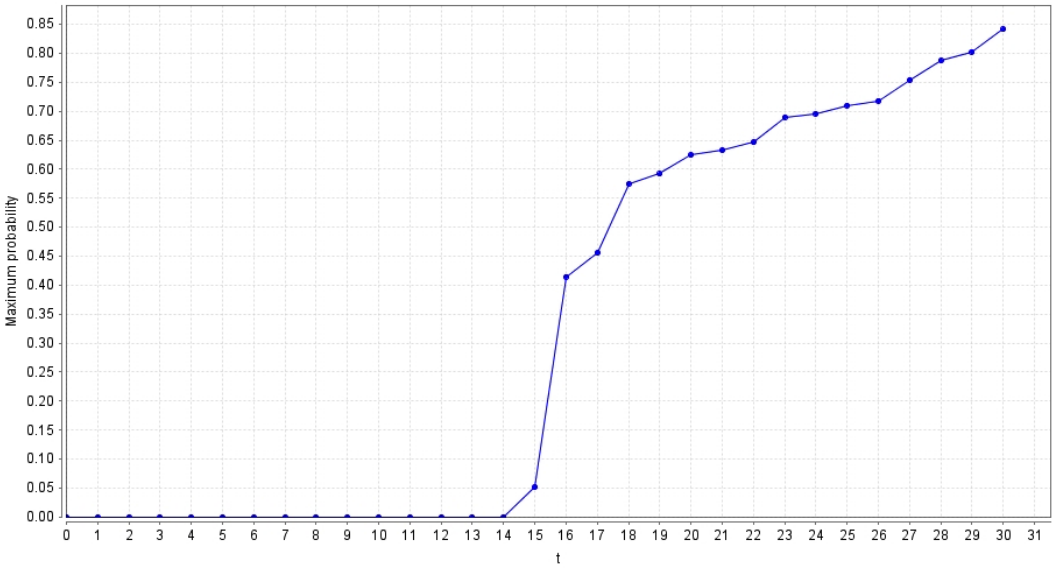
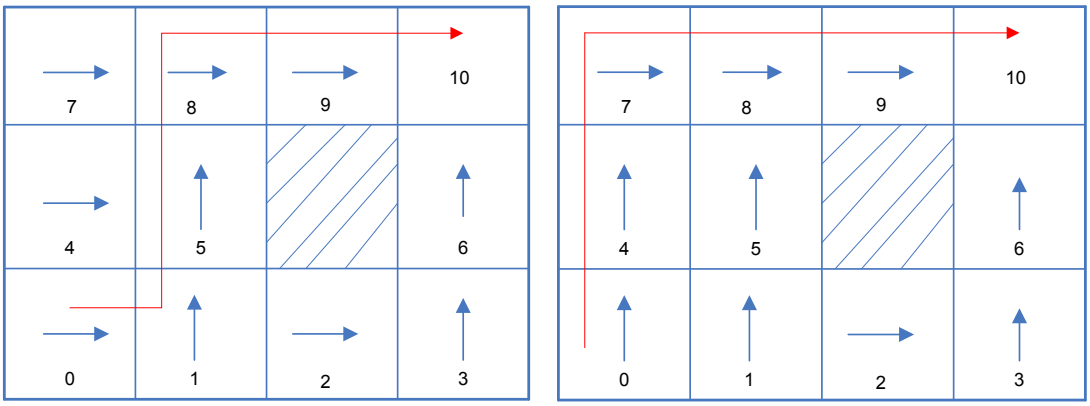


Figure 7.5: The probability curve w.r.t. mission time.

move to the site 1 instead, via location 1.



(a) Minimum fuel consumption.

(b) Minimum time

Figure 7.6: Results of a satellite to reach a given target.

The minimum energy consumption requirement of the satellite moving from site 0 to site 10 is 1.8. The route of the satellite is as shown in Figure 7.6(a) for minimum energy consumption. The best route is: $0 \rightarrow 1 \rightarrow 5 \rightarrow 8 \rightarrow 9 \rightarrow 10$. Thus, the minimum energy consumption can be calculated as: 0.2+0.4+0.9+0.1+0.2=1.8. If the direction of satellite deviates, the satellite should take the best route of each area as shown in Figure 7.6(a), for which the energy consumption is the least.

Similarly, the time the satellite spends transferring from site 0 to site 10 is at least 22.8 minutes. The shortest path is shown in Figure 7.6(b). The best route is: $0 \rightarrow 4 \rightarrow$

$7 \rightarrow 8 \rightarrow 9 \rightarrow 10$. There is one best direction from each area. Once the satellite enters into an area, the satellite will take the least time if the satellite takes the given path.

7.5 Local Path Planning for a Single Satellite

Local path planning (obstacle avoidance) is a key safety requirement for a satellite. However, sometimes its experimental verification is non-trivial, due to the stochastic behaviours of both the satellites and the obstacles. Thus, this section presents a PTA based formalism for three local path planning behaviours of satellites in uncertain dynamic environments, namely orbit change, high speed, and low speed. The PRISM model checker is applied to analyse the underlying models. This work provides a practical application of the probabilistic model checking for decision makings of complicated local path planning behaviours for satellites.

7.5.1 Local Path Planning

Local path planning is a central component for the design, development, and applications of mobile satellites, due to the fact that static or even dynamic obstacles frequently exist in their paths. When several satellites and spacecrafts or other aircrafts move in the same region, they act in fact as obstacles to one another. Research on local path planning has become an active topic in the area of satellite and space systems, and numerous algorithms have been proposed to realise the avoidance of static or dynamic obstacles [126, 98, 97]. In the past, dynamic satellite environments may be known in advance, since obstacles are assumed to have predefined or predicted moving behaviours. However, today's satellites commonly have to work in uncertain circumstances, where the movements of obstacles can not easily to be predicted accurately. Consequently, a number of probabilistic local path planning algorithms have been proposed, in which both the movement of the obstacles and the operations of the satellites are modelled as probabilistic events.

The correctness of local path planning behaviours for satellites is crucial. Simulation and testing have been the most frequently used analysis approach for verifying satellites' behaviours. However, neither of them is by any means the best solution. Their weaknesses are mainly due to two aspects: (1) the results are incomplete, due to the fact that only a subset of all the possible cases can be examined by physical system testing or software simulations; (2) the results are generally obtained from small

sample data that are unsuitable for complex probabilistic analysis.

Formal verification now becomes a very useful alternative approach to traditional analysis approaches such as simulation and testing, because it is not only complete in logic and rigorous in mathematics but adaptable for the description and analysis of probabilistic events. In recent years, formal verification has been used to verify the global path planning problem for satellite and autonomous systems. However, there is little work that applies to verifying the local path planning problem in the same domain.

7.5.2 Probabilistic Behaviours

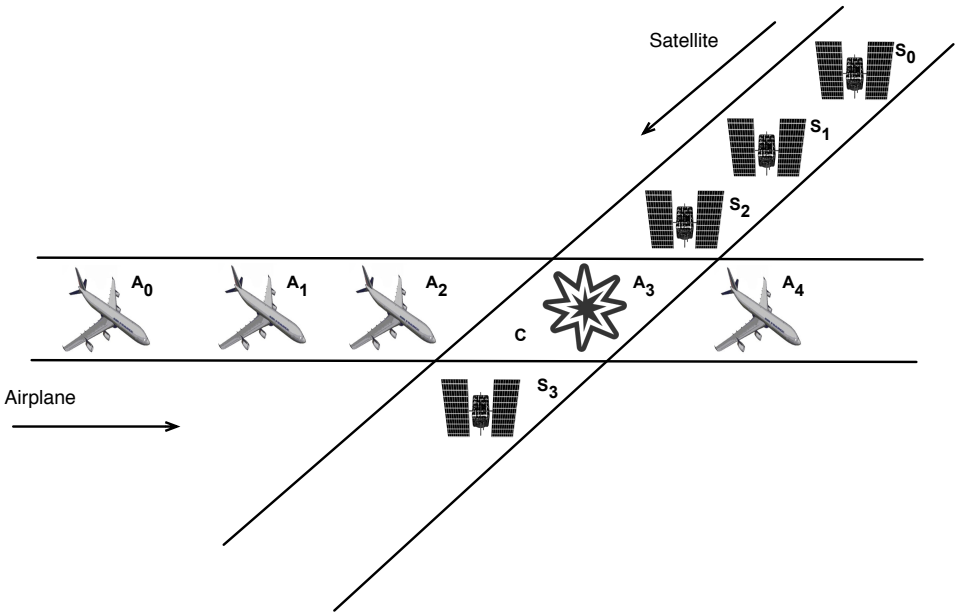


Figure 7.7: The paths of a mobile satellite and an aircraft.

In this section, we assume the dynamic obstacle is an aircraft, in the same region of the satellite. We have also made some assumptions on the movement of the aircraft: (1) the whole moving process of the aircraft can be divided into n steps; (2) the aircraft has a stepwise uniform motion, that is, it has different velocity at different time step, and the minimum and maximum velocities are denoted by v_{min} and v_{max}; (3) the aircraft changes the velocity at every same time interval $\triangle T$; (4) the velocity for a time interval is constant, and independently and randomly selected within the range of $[v_{min}, v_{max}]$.

Based on [107], the probability distribution $p(x;n)$ for the aircraft to reach the

position x after n steps can be expressed as:

$$p(x,n) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-(\frac{(x-\bar{x}_i)^2}{2\sigma_i^2})^{\gamma}}, t \geq 0, \gamma, \alpha > 0 \tag{7.1}$$

where $\sigma^2 = \sigma_0^2 + n\sigma_{step}^2$, $\bar{x}_i = x_0 + n\bar{v}\triangle T$, and $\sigma_{step}^2 = (v_{max} - v_{min})^2/12$. Here, x_0, σ, and $\bar{v}$ are the initial position, variance, average velocity, respectively. Integrating $p(x;n)$ along the practical path of the obstacle, one can obtain the probability of reaching the position x.

In this thesis, three local path planning algorithms are given for a satellite with only a single dynamic obstacle. As shown in Figure 7.7, the former and the latter are represented by a satellite and an aircraft, respectively. Assume that the paths of the satellite and the aircraft intersect at region C with angle θ ($0^\circ < \theta < 180^\circ$). A_1 and S_1 represent their positions when they begin to enter region C, while A_4 and S_3 stand for those when they just leave such a region completely. S_0 is a reference position of the satellite, which can be specified at an arbitrary point not over S_2. A_0 and A_1 are two reference positions of the aircraft. S_1 is an undetermined position of the satellite.

Under these conditions, we consider three local path planning behaviours: high speed, low speed, and orbit change. For the high speed behaviour, the satellite crosses region C before the aircraft does by increasing its velocity; for the low speed case, it passes region C later than the aircraft by decreasing its velocity; for the orbit change case, the satellite avoids the aircraft by changing its movement direction as shown in Figure 7.8. According to its probabilistic behaviours, the satellite may choose either $S_0\_S_4$ or $S_0\_S_5$. We represent $S_0\_S_4$ as the expected path, while $S_0\_S_5$ as the unexpected path. In Figure 7.8, S_4 represents the position where the aircraft begins to enter the path intersection region in the expected orbit change behaviour.

7.5.3 Formal Models

In this section, we give the probabilistic timed automaton (PTA) models for both the aircraft and the satellite with respect to three local path planning behaviours. We make an assumption that all satellites only move in horizontal directions in space relative to the Earth. This is for two reasons: (1) it simplifies the state space when modelling the behaviours, as a result it will reduce the complexity of model checking the underlying models; (2) when the satellite moves, the number of sites that a satellite needs to monitor can be changed.

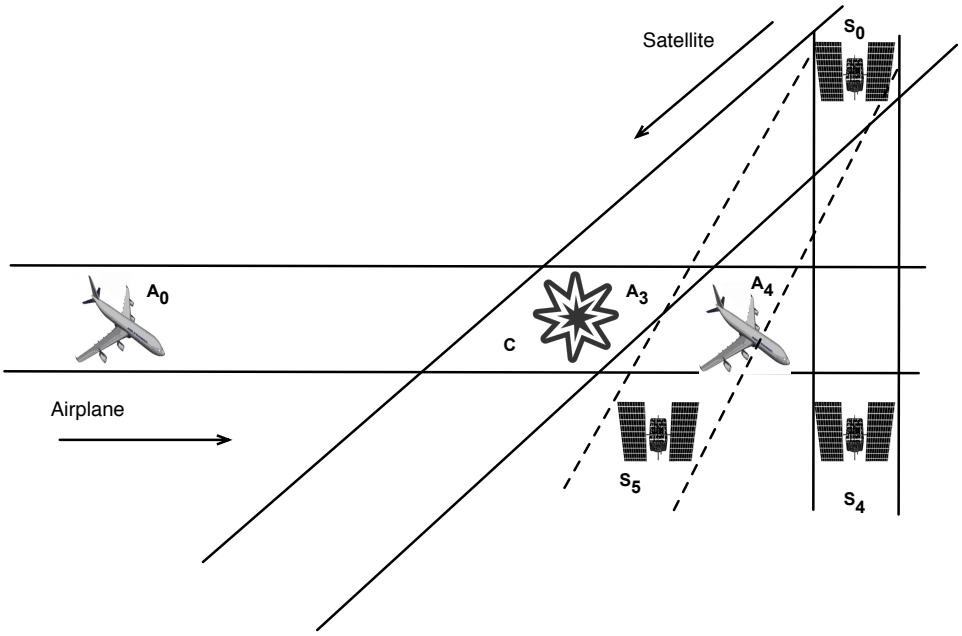


Figure 7.8: Orbit change behaviour of the satellite.

The aim is to formally verify three local path planning algorithms (orbit change, high speed, and low speed) that involves a satellite and an aircraft. The paths of the satellite and aircraft cross each other, and their movements have both probabilistic and real-time properties, which is appropriate for using probabilistic timed automata. The probabilistic models and the logical formulae are built first, and then PRISM is applied to analyse the underlying three local path planning behaviours.

7.5.3.1 PTA Model of the Aircraft

Assume that the aircraft moves along its path from the initial location A_S to the target location A_T. During the process of movement, it passes locations A_0, A_1, A_2, A_3, and A_4 in turn. There are seven corresponding states, and this can be modelled as shown in Figure 7.9. According to the local path planning behaviours, we consider $A_0\_A_1$ to be the dangerous region. As a result, if the aircraft is in state $A_0\_A_2$ or $A_2\_A_1$, the satellite should make a decision on a specific behaviour to avoid collision.

For the purpose of understanding the model, we elaborate three labels for different state transitions: (1) $A_0\_A_2 \xrightarrow{t_A<TA_0\_A_1} A_0\_A_1$, which means when the clock t_A is less than $TA_0\_A_1$, the aircraft stays in the state $A_0\_A_1$. It indicates that the aircraft is moving between the position A_0 and position A_1 (see Figure 7.7). (2) $A_0\_A_1 \xrightarrow{t_A=TA_0\_A_1,reach,t_A:=0}$

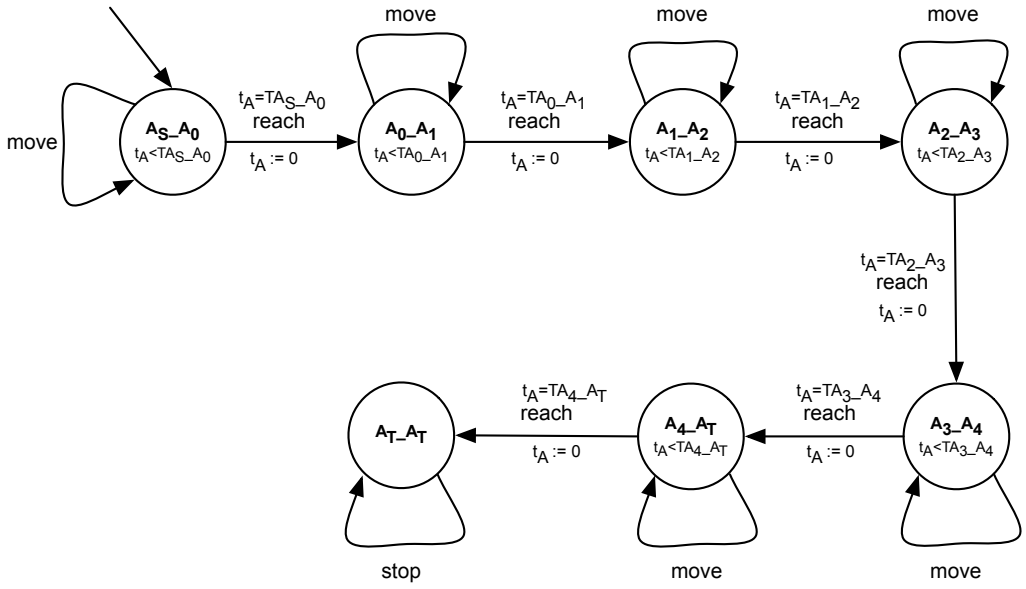


Figure 7.9: Probabilistic timed automaton of the aircraft.

$A_1\_A_2$,which means when the clock t_A is equal to $TA_0\_A_1$, the aircraft is in the state $A_1\_A_2$. It indicates that the aircraft reaches the region between the position A_1 and position A_2 from the region between the position A_0 and position A_1 (see Figure 7.7). (3) $A_T\_A_T \xrightarrow{stop} A_T\_A_T$ which means the aircraft stays in the state $A_T\_A_T$. It indicates that the aircraft reaches the terminal location.

7.5.3.2 PTA Modelling of the Satellite

In this subsection, we provide the formal representation for the state transitions of the satellite (see Figures 7.10, 7.11, and 7.12). We assume that each local path planning behaviour can be performed by the satellite with a certain probability.

By dividing each region in Figures, 7.10, 7.11, and 7.12 into several equal discrete durations in time, we are able to calculate the probability for the aircraft and satellite to reach the location by considering the probability density function in (7.1).

7.5.4 Real Time Verification

In this section, we use a case study to apply probabilistic verification of local path planning behaviours for satellites. Then, probabilistic model checking is performed using PRISM to get the probability for the three local path planning behaviours.

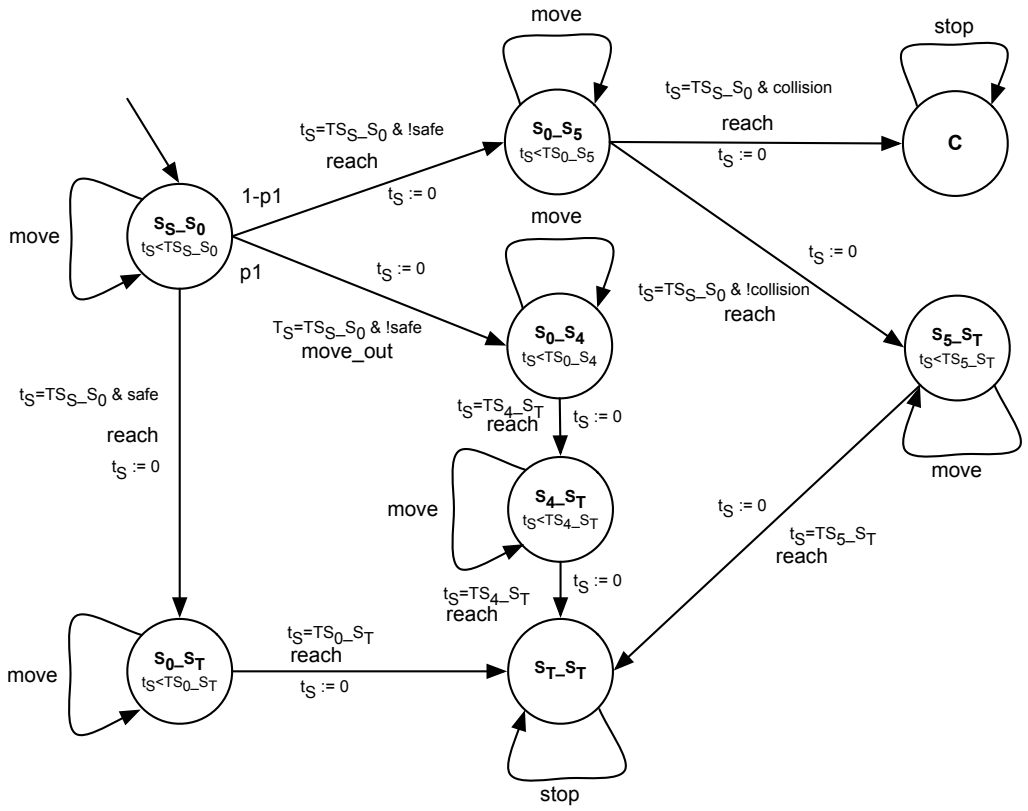


Figure 7.10: Probabilistic timed automaton of orbit change of of the satellite.

7.5.4.1 Expected Behaviour

We consider the cases of the satellite moving through the states $S_0\_S_5$ for orbit change behaviour, $S_0\_S_3$ for acceleration behaviour, and $S_0\_S_1$ for deceleration behaviour. It is assumed that $p1 = p2 = p3 = 1$.

To obtain the probability of local path planning of the satellite using PRISM, we construct the following CSL formula:

$$\mathbf{P}_{max=?}[\mathbf{F}(S_T\_S_T \ \&\ (success\_1 = 1))]$$

where the value 1 of parameter "$success\_1$" means that the satellite succeeds in avoiding the aircraft. The verification results corresponding to the three local path planning behaviours are shown in Figure 7.13.

For orbit change behaviour, if the time for the satellite to pass the region $S_0\_S_4$ is less than or equal to 7s, the aircraft will be successfully avoided. For acceleration behaviour, if the time to go across the region does not exceed 4s, the satellite is able to avoid the aircraft. For deceleration behaviour, if the time to pass the region is more

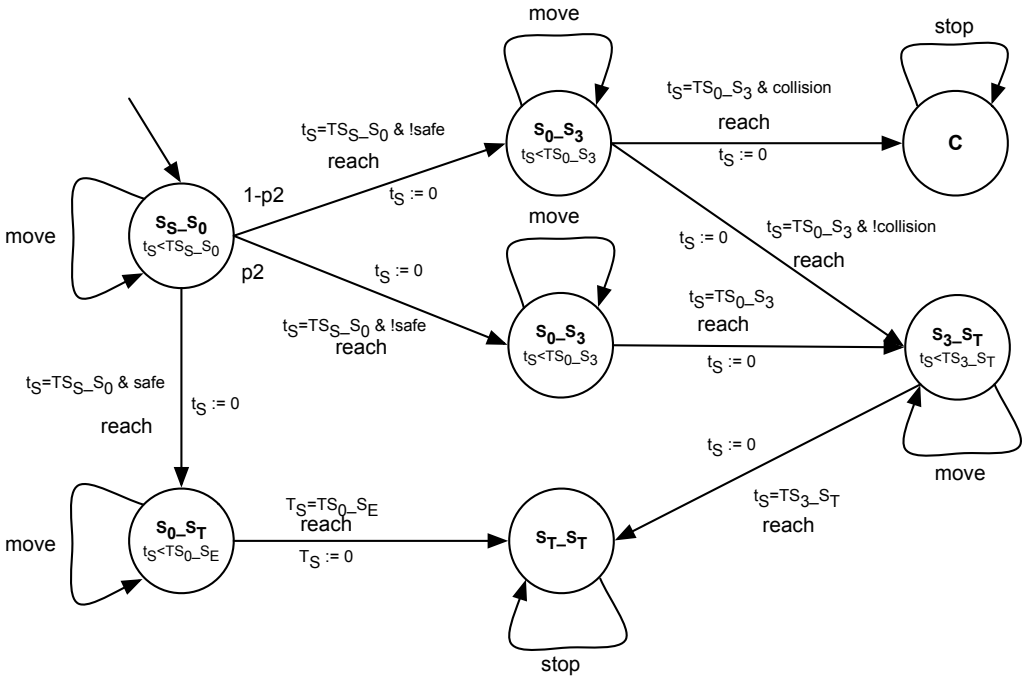


Figure 7.11: Probabilistic timed automaton of the satellite in high speed.

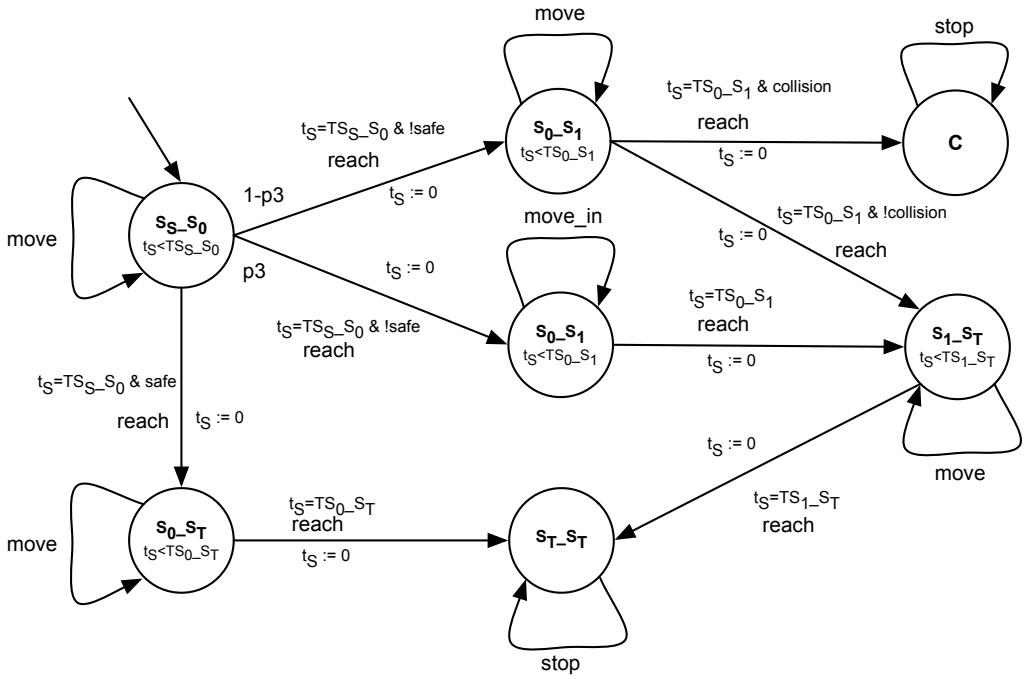


Figure 7.12: Probabilistic timed automaton of the satellite in low speed.

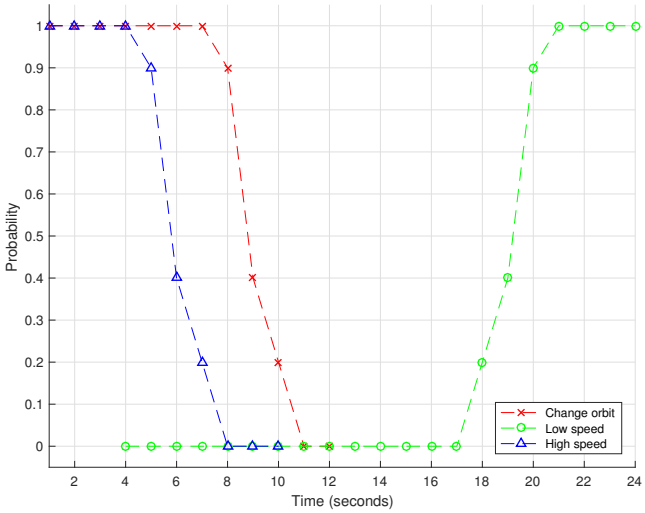


Figure 7.13: The Probability of the local path planning behaviours.

than or equal to 21s, the collision with the aircraft can be prevented.

7.5.4.2 Unexpected Behaviour

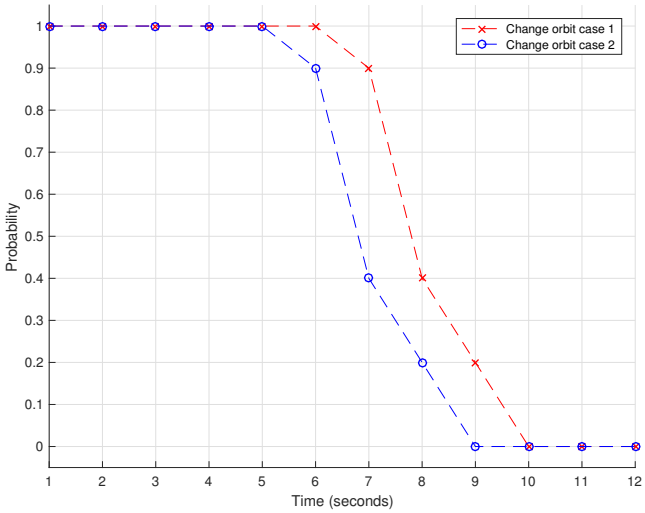


Figure 7.14: The probability of the orbit change behaviour for local path planning.

Due to the probabilistic behaviours of the satellite, it is possible that it may choose the unexpected path $S_0\_S_5$ instead of the expected path $S_0\_S_4$. In this subsection, we consider the cases of the satellite moving through the states $S_0\_S_5$ in orbit change

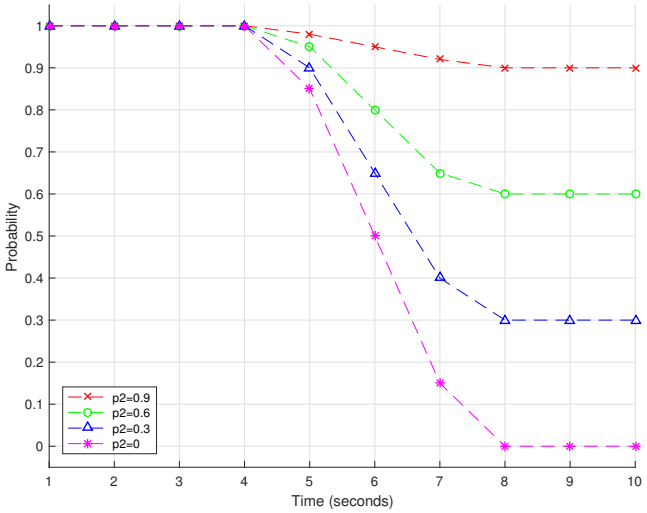


Figure 7.15: The probability of the high speed behaviour for local path planning.

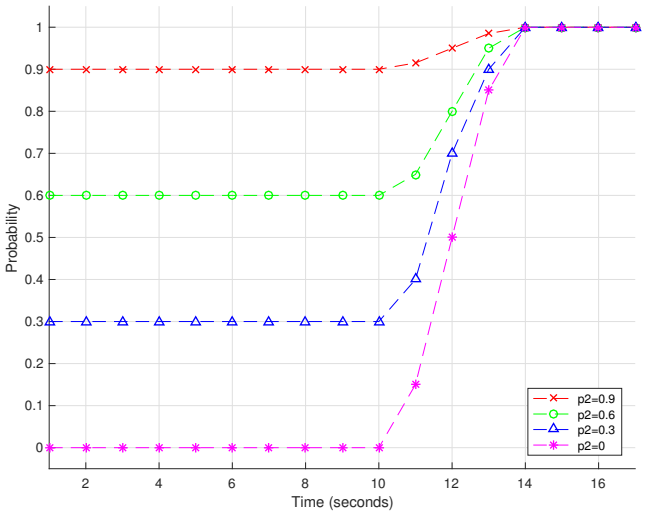


Figure 7.16: The probability of the low speed behaviour for local path planning.

behaviour, $S_0\_S_3$ in acceleration behaviour, and $S_0\_S_1$ in deceleration case. The CSL formula is:

$$\mathbf{P}_{max=?}[\mathbf{F}(S_T\_S_T \ \&\ (success\_2 = 1))]$$

where the value 1 of "*success*_2" means that the satellite succeeds in finally avoiding the aircraft, even if it goes through the unexpected state. The probability curves of the three local path planning algorithms are shown in Figures 7.14, 7.15, and 7.16,

respectively.

For orbit change behaviour, we assume that the expected heading angle is 45°, but the actual one is only 15° or 30°. Under this condition, the satellite will adjust its movement to avoid the aircraft. Figure 7.14 demonstrates that when the actual heading angle is 15° or 30°, the maximum time to successfully avoid the aircraft is 5s or 6s, respectively. For acceleration behaviour, if the time for the satellite to reach S_3 is less than or equal to 4s, the aircraft will be avoided (see Figure 7.15). For deceleration behaviour, if the time to arrive at S_1 is longer than 21s, the satellite is able to avoid the aircraft (see Figure 7.16).

7.6 Conclusion

In this chapter, we have successfully demonstrated how probabilistic model checking and Probabilistic Timed Automata (PTAs) can be used to verify safety properties via two case studies. Both global and local path planning problems of a single surveillance satellite have been considered in an uncertain dynamic environment. We first build PTAs models of the underlying behaviours. We then apply the PRISM model checker to analyse the models. We believe that this work can provide a foundation for the safety analysis and verification of complex decision makings behaviours such as path planning for satellites.

Chapter 8

Conclusions

8.1 Challenge

Satellites are highly valuable and may fail during their life cycle. They are designed to work reliably and safely in very demanding situations with a very low probability of failure. Failures depend on the characteristics of the infrastructure, application scenarios, and the maintenance strategies used. Maintenance strategies must be planned, and the probability of failure is required in order to ensure that satellite systems achieve an acceptable level of reliability. Maintenance actions associated with the repair of these failures such as rebooting, updating, or the replacement of satellites imply unavailability which is difficult to predict. Therefore, feasible approaches for predicting satellite failure are urgently needed.

Because of the mission critical nature of satellite systems, it is essential to guarantee not just qualitative correctness but also a variety of quantitative characteristics, such as reliability, availability, maintainability, and safety (RAMS), and to check if these systems meet the design requirements. RAMS parameters which may be predicted are reliability, availability, failure rate, Mean Time Between Failures (MTBF), Mean Time To Repair (MTTR), among others. RAMS analysis has been indispensable in the design phase of satellites in order to achieve minimum failure rates or to increase MTBF and thus to plan maintainability strategies, optimise reliability, maximise availability, and guarantee safety.

However, using probabilistic model checking poses a unique challenge for the RAMS analysis of satellite systems. In the past, RAMS analysis has been extensively applied to the field of electrical and electronics engineering. It has a high potential for application in the field of space science and engineering. However, it currently lacks

standardisation and suitable procedures for the correct study of RAMS characteristics for satellite systems. RAMS analysis allows system designers and reliability engineers of satellite systems to predict the likelihood of failures from the indication of historical or current operational data.

Verification is a process of analysing whether a system satisfies its specifications. Verification of RAMS requirements for computerised systems has been an active research area for decades. In the context of satellite systems, the verification problem appears to be difficult and not able to be tackled completely by current state of the art verification techniques (e.g., simulation, testing). Simulation may be relatively cheap and easy to implement, but rigorous analysis is not possible without exhaustive simulation and physical experiments. For the scenarios we examined, exhaustive simulation and experimental testing can sometimes be very difficult.

8.2 Contributions

In this thesis, we have presented a quantitative approach using probabilistic verification techniques for the analysis of reliability, availability, maintainability, and safety (RAMS) properties of satellite systems for mission critical industrial applications. A strong case for using probabilistic model checking to support RAMS analysis of satellite systems has been made by our verification results. Specifically, we have demonstrated that it is feasible, useful, and efficient to apply probabilistic verification, particularly probabilistic model checking, during the design phase, to perform predictive analysis of RAMS properties of mission critical systems that are reliant on satellites.

We make two major contributions. One of these is the approach of RAMS analysis to satellite systems. In the past, RAMS analysis has been extensively applied to the field of electrical and electronics engineering. It allows system designers and reliability engineers to predict the likelihood of failures from the indication of historical or current operational data. There is a high potential for the application of RAMS analysis in the field of space science and engineering. However, there is a lack of standardisation and suitable procedures for the correct study of RAMS characteristics for satellite systems. This thesis has considered the promising application of RAMS analysis to the case of satellite design, use, and maintenance, focusing on its system segments. Data collection and verification procedures are discussed, and a number of considerations have also been presented on how to use predict the probability of failure.

Our second contribution is leveraging the power of probabilistic model checking to

analyse satellite systems. We have presented techniques for analysing satellite systems that differ from the more common quantitative approaches based on traditional simulation and testing. These techniques have not been applied in this context before. We have demonstrated the use of probabilistic techniques via a suite of detailed examples, together with their analysis. Our presentation has been done in an incremental manner: in terms of complexity of application domains and system models, and presented a highly detailed PRISM model of each scenario. We have also provided results from practical work together with a discussion about future improvements.

We have presented quantitative verification of various probabilistic models for applying probabilistic model checking and PRISM to satellite based systems. In Chapter 4, CTMC based formal models are constructed for two different kinds of space segment: a single satellite and a constellation of satellites, in an uncertain dynamic environment. In Chapter 5, we have developed novel semi-Markov models that characterise failure behaviours, based on Weibull failure modes inferred from realistic data sources, then we have approximated and encoded these models using CTMCs. In Chapter 6, we have modelled a satellite positioning system for the mission of aircraft guidance in the probabilistic π-calculus, and encoded our model using MDPs. In Chapter 7, we have constructed different PTAs of a single satellite in both cases of global and local path planning. In summary, we have provided an alternative to the current approaches of simulation and testing for analysing RAMS properties of satellite systems.

8.3 Results

The results from the research are useful to system engineers working on the design of satellite systems and similar complex aerospace engineering systems. The key result of our work is the integration of the probabilistic model checking approach into RAMS analysis, which provides a methodology for system engineers to assess and refine their designs to meet strict or complex operational requirements of reliability and safety. We have learned several important lessons from applying this methodology to different mission-critical scenarios. These include: (1) the ability to automate the definition of RAMS properties from mission perspectives using probabilistic model checkers and the ability to abstract and examine these perspectives in probabilistic models, which provides the methodology significant values with respect to its feasibility and usability for system engineers.

Moreover, we believe our methodology and the associated techniques are success-

ful based on two following reasons. First, the expression of reliability, availability, maintainability, and safety requirements can be made easily through formal specification and re-applied automatically when system models are changed or refined. Second, scenarios in how these requirements are met can be demonstrated visually and comprehensively using probabilistic model checkers. These aspects of the methodology provide a highly usable and automated way, in which the RAMS properties of satellite systems in complex and realistic missions can be assessed.

Appendix A

Glossary of Terms, Abbreviations, and Acronyms

Abbreviations	Definition
BDD	Binary Decision Diagram
TTC	Telemetry, Tracking, and Command
CSL	Continuous Stochastic Logic
CTL	Computation Tree Logic
CTMCs	Continuous Time Markov Chains
DTMCs	Discrete Time Markov Chains
FAA	Federal Aviation Administration
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
LEO	Low Earth Orbit
LON	Launch On Need
LOS	Launch On Schedule
LOS	Line Of Sight
LTL	Linear Temporal Logic
LTS	Labelled Transition System
MDPs	Markov Decision Processes
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
NASA	National Aeronautics and Space Administration
NCSS	Navigation-Communication Satellite System

PCTL	Probabilistic Computation Tree Logic
PTAs	Probabilistic Timed Automata
PVT	Position, Velocity, and Time
RAMS	Reliability, Availability, Maintainability, and Safety
SIS	Signal In Space
STK	Satellite Tool Kit
SPS	Standard Positioning Service
SVN	Space Vehicle Number

Appendix B

Reactive Modules in PRISM for Chapter 6

```
mdp  //Makov Decision Processes

const double ra=0.8; const double rb=0.7; const double rc=0.8;
const double rd=0.8; const double re=0.8; const double rf=0.75;
const double rg=0.7;
const double rms=0.99999; const double rmcs=0.99999;
const double pef=0.9; const double rga=0.99999;
const int ma=1; const int mb=2; const int mc=3; const int md=4;
const int me=5; const int mf=6; const int mg=7;
const int va=8; const int vb=9; const int vc=10; const int vd=11;
const int ve=12; const int vf=13; const int vg=14; const int no=15;
const int e1=16; const int e2=17; const int e3=18;

module SC  // module for satellite C
        sc : [1..7] init 1;
        xc : [0..18] init 0;
        [] (sc=1) -> rc : (sc'=2) + (1-rc) : (sc'=1);
        [a3_SC_MS_mc] (sc=2) -> (sc'=3);
        [d3_GA_SC_z] (sc=3) & (z=vc) -> (sc'=4) & (xc'=z);
        [d3_GA_SC_z] (sc=3) & (z!=vc) -> (sc'=1) & (xc'=z);
        [] (sc=4) -> rc : (sc'=5) + (1-rc) : (sc'=4);
        [e3_SC_Usr_mc] (sc=5) -> (sc'=6);
```

```
        [outc_S_SC_e3] (sc=6) -> (sc'=7);
endmodule

// add further processes through renaming
// module for satellite B
module SB = SC[ sc=sb, xc=xb, rc=rb, vc=vb, a3_SC_MS_mc=a2_SB_MS_mb,
                d3_GA_SC_z=d2_GA_SB_z, e3_SC_Usr_mc=e2_SB_Usr_mb,
                outc_S_SC_e3=outb_S_SB_e2 ]
endmodule

// module for satellite A
module SA = SC[ sc=sa, xc=xa, rc=ra, vc=va, a3_SC_MS_mc=a1_SA_MS_ma,
                d3_GA_SC_z=d1_GA_SA_z, e3_SC_Usr_mc=e1_SA_Usr_ma,
                outc_S_SC_e3=outa_S_SA_e1 ]
endmodule

module SD  // module for satellite D
        sd : [1..6] init 1;
        xd : [0..18] init 0;
        [] (sd=1) -> rd : (sd'=2) + (1-rd) : (sd'=1);
        [a4_SD_MS_md] (sd=2) -> (sd'=3);
        [d4_GA_SD_z] (sd=3) & (z=vd) -> (sd'=4) & (xd'=z);
        [d4_GA_SD_z] (sd=3) & (z!=vd) -> (sd'=1) & (xd'=z);
        [] (sd=4) -> rd : (sd'=5) + (1-rd) : (sd'=4);
        [e4_SD_Usr_md] (sd=5) -> (sd'=6);
endmodule

// module for satellite E
module SE
        se : [1..7] init 1;
        xe : [0..18] init 0;
        ye : [0..18] init 0;
        [ine_S_SE_e3] (se=1) -> (se'=2) & (ye'=e3);
        [] (se=2) -> re : (se'=3) + (1-re) : (se'=2);
        [a5_SE_MS_me] (se=3) -> (se'=4);
```

```
        [d5_GA_SE_z] (se=4) & (z=ve) -> (se'=5) & (xe'=z);
        [d5_GA_SE_z] (se=4) & (z!=ve) -> (se'=2) & (xe'=z);
        [] (se=5) -> re : (se'=6) + (1-re) : (se'=5);
        [ye_SE_Usr_me] (se=6) -> (se'=7);
endmodule

// module for satellite F
module SF = SE[ se=sf, xe=xf, ye=yf, re=rf, ve=vf, e3=e2,
                ine_S_SE_e3=inf_S_SF_e2, a5_SE_MS_me=a6_SF_MS_mf,
                d5_GA_SE_z=d6_GA_SF_z, ye_SE_Usr_me=yf_SF_Usr_mf ]
endmodule

// module for satellite G
module SG = SE[ se=sg, xe=xg, ye=yg, re=rg, ve=vg, e3=e1,
                ine_S_SE_e3=ing_S_SG_e1, a5_SE_MS_me=a7_SG_MS_mg,
                d5_GA_SE_z=d7_GA_SG_z, ye_SE_Usr_me=yg_SG_Usr_mg ]
endmodule

module MS  // module for the monitor station MS
        s1 : [1..15] init 1;
        x   : [0..18] init 0;
        [a1_SA_MS_ma] (s1=1) -> (s1'=2) & (x'=ma);
        [a2_SB_MS_mb] (s1=1) -> (s1'=3) & (x'=mb);
        [a3_SC_MS_mc] (s1=1) -> (s1'=4) & (x'=mc);
        [a4_SD_MS_md] (s1=1) -> (s1'=5) & (x'=md);
        [a5_SE_MS_me] (s1=1) -> (s1'=6) & (x'=me);
        [a6_SF_MS_mf] (s1=1) -> (s1'=7) & (x'=mf);
        [a7_SG_MS_mg] (s1=1) -> (s1'=8) & (x'=mg);
        [] (s1=2) -> rms : (s1'=9) + (1-rms) : (s1'=2);
        [] (s1=3) -> rms : (s1'=10) + (1-rms) : (s1'=3);
        [] (s1=4) -> rms : (s1'=11) + (1-rms) : (s1'=4);
        [] (s1=5) -> rms : (s1'=12) + (1-rms) : (s1'=5);
        [] (s1=6) -> rms : (s1'=13) + (1-rms) : (s1'=6);
        [] (s1=7) -> rms : (s1'=14) + (1-rms) : (s1'=7);
        [] (s1=8) -> rms : (s1'=15) + (1-rms) : (s1'=8);
```

```
        [b_MS_MCS_x] (s1=9) -> (s1'=1);
        [b_MS_MCS_x] (s1=10) -> (s1'=1);
        [b_MS_MCS_x] (s1=11) -> (s1'=1);
        [b_MS_MCS_x] (s1=12) -> (s1'=1);
        [b_MS_MCS_x] (s1=13) -> (s1'=1);
        [b_MS_MCS_x] (s1=14) -> (s1'=1);
        [b_MS_MCS_x] (s1=15) -> (s1'=1);
endmodule

module MCS  // module for the master control station MCS
        s2 : [1..22] init 1;
        y : [0..18] init 0;
        vj : [0..18] init 0;
        nj : [0..18] init 0;
        [b_MS_MCS_x] (s2=1) & (x=ma) -> (s2'=2) & (y'=x);
        [b_MS_MCS_x] (s2=1) & (x=mb) -> (s2'=3) & (y'=x);
        [b_MS_MCS_x] (s2=1) & (x=mc) -> (s2'=4) & (y'=x);
        [b_MS_MCS_x] (s2=1) & (x=md) -> (s2'=5) & (y'=x);
        [b_MS_MCS_x] (s2=1) & (x=me) -> (s2'=6) & (y'=x);
        [b_MS_MCS_x] (s2=1) & (x=mf) -> (s2'=7) & (y'=x);
        [b_MS_MCS_x] (s2=1) & (x=mg) -> (s2'=8) & (y'=x);
        [] (s2=2) -> rmcs*pef : (s2'=9) + rmcs*(1-pef) : (s2'=10)
                     + (1-rmcs) : (s2'=2);
        [] (s2=3) -> rmcs*pef : (s2'=11) + rmcs*(1-pef):(s2'=12)
                     + (1-rmcs):(s2'=3);
        [] (s2=4) -> rmcs*pef : (s2'=13) + rmcs*(1-pef):(s2'=14)
                     + (1-rmcs):(s2'=4);
        [] (s2=5) -> rmcs*pef : (s2'=15) + rmcs*(1-pef):(s2'=16)
                     + (1-rmcs):(s2'=5);
        [] (s2=6) -> rmcs*pef : (s2'=17) + rmcs*(1-pef):(s2'=18)
                     + (1-rmcs):(s2'=6);
        [] (s2=7) -> rmcs*pef : (s2'=19) + rmcs*(1-pef):(s2'=20)
                     + (1-rmcs):(s2'=7);
        [] (s2=8) -> rmcs*pef : (s2'=21) + rmcs*(1-pef):(s2'=22)
                     + (1-rmcs):(s2'=8);
```

```
        [c_MCS_GA_va] (s2=9) -> (s2'=1);
        [c_MCS_GA_na] (s2=10) -> (s2'=1);
        [c_MCS_GA_vb] (s2=11) -> (s2'=1);
        [c_MCS_GA_nb] (s2=12) -> (s2'=1);
        [c_MCS_GA_vc] (s2=13) -> (s2'=1);
        [c_MCS_GA_nc] (s2=14) -> (s2'=1);
        [c_MCS_GA_vd] (s2=15) -> (s2'=1);
        [c_MCS_GA_nd] (s2=16) -> (s2'=1);
        [c_MCS_GA_ve] (s2=17) -> (s2'=1);
        [c_MCS_GA_ne] (s2=18) -> (s2'=1);
        [c_MCS_GA_vf] (s2=19) -> (s2'=1);
        [c_MCS_GA_nf] (s2=20) -> (s2'=1);
        [c_MCS_GA_vg] (s2=21) -> (s2'=1);
        [c_MCS_GA_ng] (s2=22) -> (s2'=1);
endmodule

module GA  // module for the ground antenna GA
        s3 : [1..15] init 1;
        z : [0..18] init 0;
        [c_MCS_GA_va] (s3=1) -> (s3'=2) & (z'=va);
        [c_MCS_GA_na] (s3=1) -> (s3'=2) & (z'=no);
        [c_MCS_GA_vb] (s3=1) -> (s3'=3) & (z'=vb);
        [c_MCS_GA_nb] (s3=1) -> (s3'=3) & (z'=no);
        [c_MCS_GA_vc] (s3=1) -> (s3'=4) & (z'=vc);
        [c_MCS_GA_nc] (s3=1) -> (s3'=4) & (z'=no);
        [c_MCS_GA_vd] (s3=1) -> (s3'=5) & (z'=vd);
        [c_MCS_GA_nd] (s3=1) -> (s3'=5) & (z'=no);
        [c_MCS_GA_ve] (s3=1) -> (s3'=6) & (z'=ve);
        [c_MCS_GA_ne] (s3=1) -> (s3'=6) & (z'=no);
        [c_MCS_GA_vf] (s3=1) -> (s3'=7) & (z'=vf);
        [c_MCS_GA_nf] (s3=1) -> (s3'=7) & (z'=no);
        [c_MCS_GA_vg] (s3=1) -> (s3'=8) & (z'=vg);
        [c_MCS_GA_ng] (s3=1) -> (s3'=8) & (z'=no);
        [] (s3=2) -> rga : (s3'=9) + (1-rga) : (s3'=2);
        [] (s3=3) -> rga : (s3'=10) + (1-rga) : (s3'=3);
```

```
        [] (s3=4) -> rga : (s3'=11) + (1-rga) : (s3'=4);
        [] (s3=5) -> rga : (s3'=12) + (1-rga) : (s3'=5);
        [] (s3=6) -> rga : (s3'=13) + (1-rga) : (s3'=6);
        [] (s3=7) -> rga : (s3'=14) + (1-rga) : (s3'=7);
        [] (s3=8) -> rga : (s3'=15) + (1-rga) : (s3'=8);
        [d1_GA_SA_z] (s3=9) -> (s3'=1);
        [d2_GA_SB_z] (s3=10) -> (s3'=1);
        [d3_GA_SC_z] (s3=11) -> (s3'=1);
        [d4_GA_SD_z] (s3=12) -> (s3'=1);
        [d5_GA_SE_z] (s3=13) -> (s3'=1);
        [d6_GA_SF_z] (s3=14) -> (s3'=1);
        [d7_GA_SG_z] (s3=15) -> (s3'=1);
endmodule

module User  // module for the aircraft (User segment) U
        s4 : [1..8] init 1;
        [e1_SA_Usr_ma] (s4=1) -> (s4'=1);
        [e2_SB_Usr_mb] (s4=1) -> (s4'=1);
        [e3_SC_Usr_mc] (s4=1) -> (s4'=1);
        [e4_SD_Usr_md] (s4=1) -> (s4'=1);
        [ye_SE_Usr_me] (s4=1) -> (s4'=2);
        [yf_SF_Usr_mf] (s4=2) -> (s4'=3);
        [yg_SG_Usr_mg] (s4=3) -> (s4'=4);
endmodule

module Switch  // module for the mobility model
        s5 : [1..7] init 1;
        [outc_S_SC_e3] (s5=1) -> (s5'=2);
        [ine_S_SE_e3] (s5=2) -> (s5'=3);
        [outb_S_SB_e2] (s5=3) -> (s5'=4);
        [inf_S_SF_e2] (s5=4) -> (s5'=5);
        [outa_S_SA_e1] (s5=5) -> (s5'=6);
        [ing_S_SG_e1] (s5=6) -> (s5'=7);
endmodule
```

```
// rewards (to calculate expected number of steps)
rewards "steps"
true : 1;
endrewardsdule SD  // module for satellite D
sd : [1..6] init 1;
        xd : [0..18] init 0;
[] (sd=1) -> rd : (sd'=2) + (1-rd) : (sd'=1);
[a4_SD_MS_md] (sd=2) -> (sd'=3);
        [d4_GA_SD_z] (sd=3) & (z=vd) -> (sd'=4) & (xd'=z);
        [d4_GA_SD_z] (sd=3) & (z!=vd) -> (sd'=1) & (xd'=z);
        [] (sd=4) -> rd : (sd'=5) + (1-rd) : (sd'=4);
        [e4_SD_Usr_md] (sd=5) -> (sd'=6);
endmodule

// module for satellite E
module SE
        se : [1..7] init 1;
        xe : [0..18] init 0;
        ye : [0..18] init 0;
        [ine_S_SE_e3] (se=1) -> (se'=2) & (ye'=e3);
        [] (se=2) -> re : (se'=3) + (1-re) : (se'=2);
        [a5_SE_MS_me] (se=3) -> (se'=4);
        [d5_GA_SE_z] (se=4) & (z=ve) -> (se'=5) & (xe'=z);
        [d5_GA_SE_z] (se=4) & (z!=ve) -> (se'=2) & (xe'=z);
        [] (se=5) -> re : (se'=6) + (1-re) : (se'=5);
        [ye_SE_Usr_me] (se=6) -> (se'=7);
endmodule

// module for satellite F
module SF = SE[ se=sf, xe=xf, ye=yf, re=rf, ve=vf, e3=e2,
                ine_S_SE_e3=inf_S_SF_e2, a5_SE_MS_me=a6_SF_MS_mf,
                d5_GA_SE_z=d6_GA_SF_z, ye_SE_Usr_me=yf_SF_Usr_mf ]
endmodule

// module for satellite G
```

```
module SG = SE[ se=sg, xe=xg, ye=yg, re=rg, ve=vg, e3=e1,
                ine_S_SE_e3=ing_S_SG_e1, a5_SE_MS_me=a7_SG_MS_mg,
                d5_GA_SE_z=d7_GA_SG_z, _SE_Usr_me=yg_SG_Usr_mg ]
endmodule

module MS  // module for the monitor station MS
        s1 : [1..15] init 1;
        x   : [0..18] init 0;
        [a1_SA_MS_ma] (s1=1) -> (s1'=2) & (x'=ma);
        [a2_SB_MS_mb] (s1=1) -> (s1'=3) & (x'=mb);
        [a3_SC_MS_mc] (s1=1) -> (s1'=4) & (x'=mc);
        [a4_SD_MS_md] (s1=1) -> (s1'=5) & (x'=md);
        [a5_SE_MS_me] (s1=1) -> (s1'=6) & (x'=me);
        [a6_SF_MS_mf] (s1=1) -> (s1'=7) & (x'=mf);
        [a7_SG_MS_mg] (s1=1) -> (s1'=8) & (x'=mg);
        [] (s1=2) -> rms : (s1'=9) + (1-rms) : (s1'=2);
        [] (s1=3) -> rms : (s1'=10) + (1-rms) : (s1'=3);
        [] (s1=4) -> rms : (s1'=11) + (1-rms) : (s1'=4);
        [] (s1=5) -> rms : (s1'=12) + (1-rms) : (s1'=5);
        [] (s1=6) -> rms : (s1'=13) + (1-rms) : (s1'=6);
        [] (s1=7) -> rms : (s1'=14) + (1-rms) : (s1'=7);
        [] (s1=8) -> rms : (s1'=15) + (1-rms) : (s1'=8);
        [b_MS_MCS_x] (s1=9) -> (s1'=1);
        [b_MS_MCS_x] (s1=10) -> (s1'=1);
        [b_MS_MCS_x] (s1=11) -> (s1'=1);
        [b_MS_MCS_x] (s1=12) -> (s1'=1);
        [b_MS_MCS_x] (s1=13) -> (s1'=1);
        [b_MS_MCS_x] (s1=14) -> (s1'=1);
        [b_MS_MCS_x] (s1=15) -> (s1'=1);
endmodule

module MCS  // module for the master control station MCS
        s2 : [1..22] init 1;
        y : [0..18] init 0;
        vj : [0..18] init 0;
```

```
nj : [0..18] init 0;
[b_MS_MCS_x] (s2=1) & (x=ma) -> (s2'=2) & (y'=x);
[b_MS_MCS_x] (s2=1) & (x=mb) -> (s2'=3) & (y'=x);
[b_MS_MCS_x] (s2=1) & (x=mc) -> (s2'=4) & (y'=x);
[b_MS_MCS_x] (s2=1) & (x=md) -> (s2'=5) & (y'=x);
[b_MS_MCS_x] (s2=1) & (x=me) -> (s2'=6) & (y'=x);
[b_MS_MCS_x] (s2=1) & (x=mf) -> (s2'=7) & (y'=x);
[b_MS_MCS_x] (s2=1) & (x=mg) -> (s2'=8) & (y'=x);
[] (s2=2) -> rmcs*pef : (s2'=9) + rmcs*(1-pef) : (s2'=10)
             + (1-rmcs) : (s2'=2);
[] (s2=3) -> rmcs*pef : (s2'=11) + rmcs*(1-pef):(s2'=12)
             + (1-rmcs):(s2'=3);
[] (s2=4) -> rmcs*pef : (s2'=13) + rmcs*(1-pef):(s2'=14)
             + (1-rmcs):(s2'=4);
[] (s2=5) -> rmcs*pef : (s2'=15) + rmcs*(1-pef):(s2'=16)
             + (1-rmcs):(s2'=5);
[] (s2=6) -> rmcs*pef : (s2'=17) + rmcs*(1-pef):(s2'=18)
             + (1-rmcs):(s2'=6);
[] (s2=7) -> rmcs*pef : (s2'=19) + rmcs*(1-pef):(s2'=20)
             + (1-rmcs):(s2'=7);
[] (s2=8) -> rmcs*pef : (s2'=21) + rmcs*(1-pef):(s2'=22)
             + (1-rmcs):(s2'=8);
[c_MCS_GA_va] (s2=9) -> (s2'=1);
[c_MCS_GA_na] (s2=10) -> (s2'=1);
[c_MCS_GA_vb] (s2=11) -> (s2'=1);
[c_MCS_GA_nb] (s2=12) -> (s2'=1);
[c_MCS_GA_vc] (s2=13) -> (s2'=1);
[c_MCS_GA_nc] (s2=14) -> (s2'=1);
[c_MCS_GA_vd] (s2=15) -> (s2'=1);
[c_MCS_GA_nd] (s2=16) -> (s2'=1);
[c_MCS_GA_ve] (s2=17) -> (s2'=1);
[c_MCS_GA_ne] (s2=18) -> (s2'=1);
[c_MCS_GA_vf] (s2=19) -> (s2'=1);
[c_MCS_GA_nf] (s2=20) -> (s2'=1);
[c_MCS_GA_vg] (s2=21) -> (s2'=1);
```

```
        [c_MCS_GA_ng] (s2=22) -> (s2'=1);
endmodule

module GA  // module for the ground antenna GA
        s3 : [1..15] init 1;
        z : [0..18] init 0;
        [c_MCS_GA_va] (s3=1) -> (s3'=2) & (z'=va);
        [c_MCS_GA_na] (s3=1) -> (s3'=2) & (z'=no);
        [c_MCS_GA_vb] (s3=1) -> (s3'=3) & (z'=vb);
        [c_MCS_GA_nb] (s3=1) -> (s3'=3) & (z'=no);
        [c_MCS_GA_vc] (s3=1) -> (s3'=4) & (z'=vc);
        [c_MCS_GA_nc] (s3=1) -> (s3'=4) & (z'=no);
        [c_MCS_GA_vd] (s3=1) -> (s3'=5) & (z'=vd);
        [c_MCS_GA_nd] (s3=1) -> (s3'=5) & (z'=no);
        [c_MCS_GA_ve] (s3=1) -> (s3'=6) & (z'=ve);
        [c_MCS_GA_ne] (s3=1) -> (s3'=6) & (z'=no);
        [c_MCS_GA_vf] (s3=1) -> (s3'=7) & (z'=vf);
        [c_MCS_GA_nf] (s3=1) -> (s3'=7) & (z'=no);
        [c_MCS_GA_vg] (s3=1) -> (s3'=8) & (z'=vg);
        [c_MCS_GA_ng] (s3=1) -> (s3'=8) & (z'=no);
        [] (s3=2) -> rga : (s3'=9) + (1-rga) : (s3'=2);
        [] (s3=3) -> rga : (s3'=10) + (1-rga) : (s3'=3);
        [] (s3=4) -> rga : (s3'=11) + (1-rga) : (s3'=4);
        [] (s3=5) -> rga : (s3'=12) + (1-rga) : (s3'=5);
        [] (s3=6) -> rga : (s3'=13) + (1-rga) : (s3'=6);
        [] (s3=7) -> rga : (s3'=14) + (1-rga) : (s3'=7);
        [] (s3=8) -> rga : (s3'=15) + (1-rga) : (s3'=8);
        [d1_GA_SA_z] (s3=9) -> (s3'=1);
        [d2_GA_SB_z] (s3=10) -> (s3'=1);
        [d3_GA_SC_z] (s3=11) -> (s3'=1);
        [d4_GA_SD_z] (s3=12) -> (s3'=1);
        [d5_GA_SE_z] (s3=13) -> (s3'=1);
        [d6_GA_SF_z] (s3=14) -> (s3'=1);
        [d7_GA_SG_z] (s3=15) -> (s3'=1);
endmodule
```

```
module User  // module for the aircraft (User segment) U
        s4 : [1..4] init 1;
        [e1_SA_Usr_ma] (s4=1) -> (s4'=1);
        [e2_SB_Usr_mb] (s4=1) -> (s4'=1);
        [e3_SC_Usr_mc] (s4=1) -> (s4'=1);
        [e4_SD_Usr_md] (s4=1) -> (s4'=1);
        [ye_SE_Usr_me] (s4=1) -> (s4'=2);
        [yf_SF_Usr_mf] (s4=2) -> (s4'=3);
        [yg_SG_Usr_mg] (s4=3) -> (s4'=4);
endmodule

module Switch  // module for the mobility model
        s5 : [1..7] init 1;
        [outc_S_SC_e3] (s5=1) -> (s5'=2);
        [ine_S_SE_e3] (s5=2) -> (s5'=3);
        [outb_S_SB_e2] (s5=3) -> (s5'=4);
        [inf_S_SF_e2] (s5=4) -> (s5'=5);
        [outa_S_SA_e1] (s5=5) -> (s5'=6);
        [ing_S_SG_e1] (s5=6) -> (s5'=7);
endmodule
```

Bibliography

[1] O. Abdelkhalik and A. Gad. Optimization of space orbits design for earth orbiting missions. *Acta Astronautica*, 68(7-8):1307–1317, 2011.

[2] R. Abo, K. Barkaoui, and K. Djouani. Verification and Performance Evaluation of S-MAC Protocol Based on Process Calculi. In *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2010 Workshops)*, pages 189–198. IEEE, 2010.

[3] W. Ahmed, O. Hasan, and S. Tahar. Formal Dependability Modeling and Analysis: A Survey. In M. Kohlhase, M. Johansson, B. Miller, L. de Moura, and F. Tompa, editors, *Intelligent Computer Mathematics*, volume 9791 of *Lecture Notes in Computer Science*, pages 132–147. Springer International Publishing, 2016.

[4] R. Alur, C. Courcoubetis, and D. Dill. Model-Checking for Real-Time Systems. In *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science (LICS 1990)*, pages 414–425, Philadelphia, PA, 1990. IEEE.

[5] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

[6] R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[7] G. S. Aoude, J. P. How, and I. M. Garcia. Two-Stage Path Planning Approach for Solving Multiple Spacecraft Reconfiguration Maneuvers. *The Journal of the Astronautical Sciences*, 56(4):515–544, October–December 2008.

[8] H. Arabian-Hoseynabadi, H. Oraee, and P. Tavner. Failure Modes and Effects Analysis (fmea) for wind turbines. *International Journal of Electrical Power & Energy Systems*, 32(7):817–824, September 2010.

[9] M. Arapinis, M. Calder, L. Dennis, M. Fisher, P. Gray, S. Konur, A. Miller, E. Ritter, M. Ryan, S. Schewe, C. Unsworth, and R. Yasmin. Towards the Verification of Pervasive Systems. *Electronic Communications of the EASST*, 22:1–15, 2009.

[10] S. Arrizabalaga, J. Mendizabal, S. Pinte, J. Sánchez, J. González, J. Bauer, M. Themistokleous, and D. Lowe. Development of an Advanced Testing System and Smart Train Positioning System for ETCS applications. In *Proceedings of the 5th Transport Research Arena Conference (TRA 2015)*, 2014.

[11] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying Continuous Time Markov Chains. In R. Alur and T. A. Henzinger, editors, *Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 269–276. Springer Berlin Heidelberg, 1996.

[12] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, Cambridge, MA, 2008.

[13] C. Baier, J.-P. Katoen, and H. Hermanns. Approximative Symbolic Model Checking of Continuous-Time Markov Chains. In J. C. M. Baeten and S. Mauw, editors, *CONCUR'99 Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–161. Lecture Notes in Computer Science, 1999.

[14] M. Beccuti, G. Franceschinis, and J. Sproston. Modeling and Verification of Distributed Systems Using Markov Decision Processes. In D. Bruneo and S. Distefano, editors, *Quantitative Assessments of Distributed Systems: Methodologies and Techniques*, chapter 1, pages 3–26. Wiley, 2015.

[15] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi. UPPAAL - a Tool Suite for Automatic Verification of Real-Time Systems. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 232–243. Springer Berlin Heidelberg, 1996.

[16] J. Beugin and J. Marais. Simulation-based evaluation of dependability and safety properties of satellite technologies for railway localization. *Transportation Research Part C*, 22(June):42–57, 2012.

[17] A. Birolini. *Reliability Engineering: Theory and Practice*. Springer, 6th edition, September 2010.

[18] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A Model-Checking Tool for Real-Time Systems. In A. J. Hu and M. Y. Vardi, editors, *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 546–550. Springer Berlin Heidelberg, 2005.

[19] M. Bozzano, A. Cimatti, J.-P. Katoen, P. Katsaros, K. Mokos, V. Y. Nguyen, T. Noll, B. Postma, and M. Roveri. Spacecraft early design validation using formal methods. *Reliability Engineering & System Safety*, 132:20–35, December 2014.

[20] G. Brat, E. Denney, D. Giannakopoulou, J. Frank, and A. Jonsson. Verification of Autonomous Systems for Space Applications. In *Proceedings of the 2006 IEEE Aerospace Conference*, pages 1–10, Big Sky, MT, 2006. IEEE.

[21] J.-F. Castet and J. H. Saleh. Satellite and satellite subsystems reliability- statistical data analysis and modeling. *Reliability Engineering and System Safety*, 94(11):1718–1728, November 2009.

[22] J.-F. Castet and J. H. Saleh. Satellite reliability: Statistical data analysis and modeling. *Journal of Spacecraft and Rockets*, 46(5):1065–1076, 2009.

[23] J.-F. Castet and J. H. Saleh. Beyond reliability, multi-state failure analysis of satellite subsystems: A statistical approach. *Reliability Engineering & System Safety*, 95(4):311–322, April 2010.

[24] F. Cavaliere, F. Mari, I. Melatti, G. Minei, I. Salvo, E. Tronci, G. Verzino, and Y. Yushtein. Model Checking Satellite Operational Procedures. In *Proceedings of The International Space System Engineering Conference (DASIA 2011)*, pages 1–8, Malta, 2011.

[25] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In E. Brinksma and K. G. Larsen, editors, *Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364. Springer Berlin Heidelberg, 2002.

[26] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, March 2000.

[27] G. Ciobanu and A. Rotaru. Phase-Type Approximations for Non-Markovian Systems: A Case Study. In C. Canal and A. Idani, editors, *Software Engineering and Formal Methods*, volume 8938 of *Lecture Notes in Computer Science*, pages 323–334. Springer International Publishing, 2015.

[28] C. Circi, E. Ortore, and F. Bunkheila. Satellite constellations in sliding ground track orbits. *Aerospace Science and Technology*, 39:395–402, December 2014.

[29] E. Clarke, S. C. K. McMillan, and V. Hartonas-Garmhausen. Symbolic Model Checking. In R. Alur and T. A. Henzinger, editors, *Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 419–422. Springer Berlin Heidelberg, 2005.

[30] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.

[31] H. Cui and C. Han. Satellite Constellation Configuration Design with Rapid Performance Calculation and Ordinal Optimization. *Chinese Journal of Aeronautics*, 24(5):631–639, 2011.

[32] H. Czichos. Scope of Technical Diagnostics. In H. Czichos, editor, *Handbook of Technical Diagnostics*, pages 3–9. Springer Berlin Heidelberg, 2013.

[33] L. A. Dennis, M. Fisher, N. K. Lincoln, A. Lisitsa, and S. M. Veres. Practical verification of decision-making in agent-based autonomous systems. *Automated Software Engineering*, 23(3):305–359, 2016.

[34] Department of Defense, USA. *Global Positioning System Standard Positionning Service Performance Standard*, September 2008.

[35] L. Devauchelle, P. G. Larsen, and H. Voss. PICGAL: Practical Use of Formal Specification to Develop a Complex Critical System. In J. Fitzgerald, C. B. Jones, and P. Lucas, editors, *FME '97: Industrial Applications and Strengthened Foundations of Formal Methods*, volume 1313 of *Lecture Notes in Computer Science*, pages 221–236. Springer Berlin Heidelberg, 1997.

[36] D. L. Dill, A. J. Drexler, A. J. Hu, and C. H. Yang. Protocol Verification as a Hardware Design Aid. In *Proceedings of the 10th IEEE International Conference on Computer Design on VLSI in Computer and Processors (ICCD 1992)*, pages 522–525, Cambridge, MA, 1992. IEEE.

[37] S. Distefano and A. Puliafito. Dependability Evaluation with Dynamic Reliability Block Diagrams and Dynamic Fault Trees. *IEEE Transactions on Dependable and Secure Computing*, 6(1):4–17, January-March 2009.

[38] M. Duflot, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic Model Checking of the CSMA/CD Protocol Using PRISM and APMC. *Electronic Notes in Theoretical Computer Science*, 128(6):195–214, May 2005.

[39] M. Duflot, M. Kwiatkowska, G. Norman, D. Parker, S. Peyronnet, C. Picaronny, and J. Sproston. Practical Applications of Probabilistic Model Checking to Communication Protocols. In S. Gnesi and T. Margaria, editors, *Formal Methods for Industrial Critical Systems: A Survey of Applications*, chapter 7, pages 133–150. Wiley, 2012.

[40] J.-M. Durand and A. Caseau. GPS Availability, Part II: Evaluation of State Probabilities for 21 Satellite and 24 Satellite Constellations. *Navigation*, 37(3):285–296, 1990.

[41] J.-M. Durand, T. Michal, and J. Bouchard. GPS Availability, part I: Availability of Service Achievable for Different Categories of Civil Users. *Navigation*, 37(2):123–139, 1990.

[42] B. Dutertre. Probabilistic Analysis of Distributed Fault-Tolerant Systems. Technical Report NASA/CR-2011-217090, NASA, May 2011.

[43] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Formal Models and Semantics*, pages 995, 997–1072. Elsevier B.V., 1990.

[44] M.-A. Esteve, J.-P. Katoen, V. Y. Nguyen, B. Postma, and Y. Yushtein. Formal Correctness, Safety, Dependability, and Performance Analysis of a Satellite. In *Proceedings of the 34th International Conference on Software Engineering (ICSE 2012)*, pages 1022–1031. IEEE, 2012.

[45] FAA. Global Positoning System (GPS) Standard Positioning Service (SPS) Performance Analysis Report. 2013.

[46] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Temporal Logic Motion Planning for Mobile Robots. In *Proceedings of ICRA 2005*, pages 2020–2025. IEEE, 2005.

[47] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 31(3-4):245–279, 1998.

[48] X. Gan, J. Dubrovin, and K. Heljanko. A symbolic model checking approach to verifying satellite onboard software. *Science of Computer Programming*, 82:44–55, March 2014.

[49] V. Giovanni, F. Cavaliere, F. Mari, I. Melatti, G. Minei, I. Salvo, Y. Yushtein, and E. Tronci. Model Checking Driven Simulation of Sat Procedures. In *Proceedings of 12th International Conference on Space Operations (SpaceOps 2012)*, pages 1–13, 2012.

[50] A. Gomes, A. Mota, A. Sampaio, F. Ferri, and J. Buzzi. Systematic Model-Based Safety Assessment Via Probabilistic Model Checking. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification, and Validation*, volume 6415 of *Lecture Notes in Computer Science*, pages 625–639. Springer Berlin Heidelberg, 2010.

[51] K. Gopinath, J. Elerath, and D. Long. Reliability Modelling of Disk Subsystems with Probabilistic Model Checking. Technical Report UCSC-SSRC-09-05, University of California, Santa Cruz, 2009.

[52] M. S. Grewal and A. P. Andrews. Applications of Kalman Filtering in Aerospace 1960 to the Present. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.

[53] H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[54] K. Havelund, M. Lowry, and J. Penix. Formal Analysis of a Space-Craft Controller using SPIN. *IEEE Transactions on Software Engineering*, 27(8):749–765, 2001.

[55] K. Havelund and T. Pressburger. Model checking JAVA programs using JAVA Pathfinder. *International Journal on Software Tools for Technology Transfer*, 2(4):366–381, March 2000.

[56] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

[57] D. Henriques, J. G. Martins, P. Zuliani, A. Platzer, and E. M. Clarke. Statistical Model Checking for Markov Decision Processes. In *Proceedings of the 9th International Conference on Quantitative Evaluation of Systems (QEST 2012)*, pages 84–93. IEEE, 2012.

[58] H. Hermanns and J.-P. Katoen. Performance Evaluation:= (Process Algebra + Model Checking) x Markov Chains. In K. G. Larsen and M. Nielsen, editors, *CONCUR 2001 — Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 59–81. Springer Berlin Heidelberg, 2001.

[59] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. ETMCC: Model Checking Performability Properties of Markov Chains. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2003)*, page 1. IEEE, 2003.

[60] M. G. Hinchey and J. P. Bowen. *Applications of Formal Methods*. Prentice Hall, 1 edition edition, 1995.

[61] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle. *GNSS - Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Springer-Verlag, Vienna, Austria, 2007.

[62] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997.

[63] G. J. Holzmann. *The SPIN Model Checker*. Addison-Wesley, Boston, MA, 2004.

[64] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2nd edition edition, 2004.

[65] K. Jackson. Global Positioning System (GPS), 2013.

[66] C. W. Johnson. The Natural History of Bugs: Using Formal Methods to Analyse Software Related Failures in Space Missions. In J. Fitzgerald, I. J. Hayes, and A. Tarlecki, editors, *FM 2005: Formal Methods*, volume 3582 of *Lecture Notes in Computer Science*, pages 9–25. Springer-Verlag, 2005.

[67] C. W. Johnson. Innovation vs Safety: Hazard Analysis Techniques to Avoid Premature Commitment in the Early Stage Development of National Critical

Infrastructures. In *Proceedings of 32nd International Systems Safety Conference*, 2014.

[68] J. P. Katoen, M. Khattri, and I. S. Zapreevt. A Markov Reward Model Checker. In *Proceedings of the 2nd International Conference on the Quantitative Evaluation of Systems (QEST 2005)*, pages 243–244. IEEE, 2005.

[69] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker mrmc. *Performance Evaluation*, 68(2):90–104, February 2011.

[70] C. Kelley and M. Dessouky. Minimizing the Cost of Availability of Coverage from a Constellation of Satellites: Evaluation of Optimization Methods. *Systems Engineering*, 7(2):113–122, 2004.

[71] C. Kelley and M. Dessouky. Minimizing the Cost of Availability of Coverage from a Constellation of Satellites: Evaluation of Optimization Methods. *Systems Engineering*, 7(2):113–122, 2004.

[72] K. Kubik, Y. Feng, and T. Tang. An Availability Study for a Nav-Com Satellite System (NCSS) in Australia. In *Proceedings of the 9th National Space Engineering Symposium*, pages 59–66, 1994.

[73] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, April 2004.

[74] M. Kwiatkowska, G. Norman, and D. Parker. Symmetry Reduction for Probabilistic Model Checking. In *Computer Aided Verification*, volume 4144 of *Lecture Notes in Computer Science*, pages 234–248. Springer Berlin Heidelberg, 2006.

[75] M. Kwiatkowska, G. Norman, and D. Parker. Controller dependability analysis by probabilistic model checking. *Control Engineering Practice*, 15(11):1427–1434, 2007.

[76] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic Model Checking. In M. Bernardo and J. Hillston, editors, *Formal Methods for Performance Evaluation*, number 4486 in Lecture Notes in Computer Science, pages 220–270. Springer Berlin Heidelberg, 2007.

[77] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic Model Checking for Performance and Reliability Analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.

[78] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In G. Gopalakrishnan and S. Qadeer, editors, *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer Berlin Heidelberg, 2011.

[79] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29(1):33–78, 2006.

[80] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying Quantitative Properties of Continuous Probabilistic Timed Automata. In C. Palamidessi, editor, *CONCUR 2000 — Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 123–137. Springer Berlin Heidelberg, 2000.

[81] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic Model Checking of Deadline Properties in the IEEE 1394 FireWire Root Contention Protocol. *Formal Aspects of Computing*, 14(3):295–318, April 2003.

[82] T. J. Lang and W. S. Adams. A Comparison of Satellite Constellations for Continuous Global Coverage. In J. C. van der Ha, editor, *Mission Design & Implementation of Satellite Constellations*, volume 1 of *Space Technology Proceedings*, pages 51–62. Springer Netherlands, 1998.

[83] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1):134–152, December 1997.

[84] Y.-W. Lee, Y.-C. Suh, and R. Shibasaki. A simulation system for GNSS multipath mitigation using spatial statistical methods. *Computers & Geosciences*, 34(11):1597–1609, 2008.

[85] Lockheed Martin. GPS III. The Next Generation Global Positioning System, 2011.

[86] G. G. I. López, H. Hermanns, and J.-P. Katoen. Beyond Memoryless Distributions: Model Checking Semi-Markov Chains. In L. de Alfaro and S. Gilmore,

editors, *Process Algebra and Probabilistic Methods. Performance Modelling and Verification*, volume 2165 of *Lecture Notes in Computer Science*, pages 57–70. Springer Berlin Heidelberg, 2001.

[87] D. Lu and E. Schnieder. Performance Evaluation of GNSS for Train Localization. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):1054–1059, April 2015.

[88] D. Lu, F. G. Toro, and E. Schnieder. RAMS Evaluation of GNSS for Railway Localisation. In *Proceedings of the IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, pages 209–214. IEEE, 2013.

[89] Y. Lu, A. A. Miller, R. Hoffmann, and C. W. Johnson. Towards the Automated Verification of Weibull Distributions for Systems Failure Rates. In M. H. ter Beek, S. Gnesi, and A. Knapp, editors, *Critical Systems: Formal Methods and Automated Verification*, volume 9933 of *Lecture Notes in Computer Science*. Springer International Publishing, 2016.

[90] Y. Lu, A. A. Miller, and C. W. Johnson. Automatic Verification for Satellite Subsystems failures with Weibull Distributions. In Preparation.

[91] Y. Lu, A. A. Miller, C. W. Johnson, Z. Peng, and T. Zhao. Availability Analysis of Satellite Positioning Systems for Aviation using the PRISM Model Checker. In *Proceedings of the 17th IEEE International Conference on Computational Science and Engineering (CSE 2014)*, pages 704–713. IEEE, 2014.

[92] Y. Lu, Z. Peng, A. A. Miller, T. Zhao, and C. W. Johnson. Timed Fault Tree Models of the China Yongwen Railway Accident. In *Proceedings of the 8th Asia Modelling Symposium (AMS 2014)*, pages 128–133. IEEE, 2014.

[93] Y. Lu, Z. Peng, A. A. Miller, T. Zhao, and C. W. Johnson. How reliable is satellite navigation for aviation? checking availability properties with probabilistic verification. *Reliability Engineering & System Safety*, 144:95–116, December 2015.

[94] M. Malhotra and A. Reibman. Selecting and implementing phase approximations for semi-Markov models. *Communications in Statistics. Stochastic Models*, 9(4):473–506, 1993.

[95] T. Mancini, F. Mari, A. Massini, I. Melatti, F. Merli, and E. Tronci. System Level Formal Verification via Model Checking Driven Simulation. In N. Sharygina and H. Veith, editors, *Computer Aided Verification*, volume 8044 of *LNCS*, pages 296–312. Springer Berlin Heidelberg, 2013.

[96] T. Mancini, F. Mari, A. Massini, I. Melatti, and E. Tronci. System Level Formal Verification via Distributed Multi-Core Hardware in the Loop Simulation. In *Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2014)*, pages 734–742, Torino, Italy, 2014. IEEE.

[97] N. Martinson. Obstacle avoidance guidance and control algorithm for spacecraft maneuvers. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pages 1–9. AIAA, 2009.

[98] S. B. McCamish. *Distributed autonomous control of multiple spacecraft during close proximity operations*. PhD thesis, Naval Postgraduate School, 2007.

[99] K. N. McCauley. Putting Precision in Operations: Beidou Satellite Navigation System. *China Brief*, 14(6), 2014.

[100] K. L. McMillan. *Symbolic Model Checking*. Springer US, 1993.

[101] J. Mencik. Concise reliability for engineers. In *Failure Modes and Effects Analysis*, chapter 12, pages 89–95. InTech, 2016.

[102] J. Mencik. Fault tree analysis and reliability block diagrams. In *Concise Reliability for Engineers*, chapter 13, pages 97–100. InTech, 2016.

[103] A. Miller, A. Donaldson, and M. Calder. Symmetry in temporal logic model checking. *ACM Computing Surveys*, 38(3):Article No. 8, 2006.

[104] A. A. Miller and A. F. Donaldson. Property Preservation in Quotient Structures. Technical Report TR-2008-270, University of Glasgow, 2008.

[105] R. Milner. A Calculus of Mobile Processes, I. *Information and Computation*, 100(1):1–40, 1992.

[106] R. Milner. A Calculus of Mobile Processes, II. *Information and Computation*, 100(1):41–77, 1992.

[107] J. Miura and Y. Shirai. Modeling Motion Uncertainty of Moving Obstacles for Robot Motion Planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2000)*, pages 2258–2263. IEEE, 2000.

[108] NOAA. National Coordination Office for Space-Based Positioning, Navigation, and Timing. `http://www.gps.gov/systems/gps/space`, 2013.

[109] G. Norman, C. Palamidessi, D. Parker, and P. Wu. Model Checking Probabilistic and Stochastic Extensions of the π-Calculus. *IEEE Transactions on Software Engineering*, 35(2):209–223, 2009.

[110] G. Norman, C. Palamidessi, D. Parker, and P. Wu. Model Checking Probabilistic and Stochastic Extensions of the π-Calculus. *IEEE Transactions on Software Engineering*, 35(2):209–223, 2009.

[111] G. Norman, D. Parker, and J. Sproston. Model checking for probabilistic timed automata. *Formal Methods in System Design*, 43(2):164–190, 2013.

[112] C. M. noz, V. C. no, and G. Dowek. Formal Analysis of the Operational Concept for the Small Aircraft Transportation System. In M. Butler, C. B. Jones, A. Romanovsky, and E. Troubitsyna, editors, *Rigorous Development of Complex Fault-Tolerant Systems*, volume 4157 of *Lecture Notes in Computer Science*, pages 306–325. Springer Berlin Heidelberg, 2006.

[113] S. Owre, J. M. Rushby, and N. Shankar. PVS: A Prototype Verification System. In D. Kapur, editor, *Automated Deduction—CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 748–752. Springer-Verlag, 1992.

[114] Z. Peng, Y. Lu, and A. A. Miller. Uncertainty Analysis of Phased Mission Systems with Probabilistic Timed Automata. In *Proceedings of the 7th IEEE International Conference on Prognostics and Health Management (PHM 2016)*, pages 1–8. IEEE, 2016.

[115] Z. Peng, Y. Lu, A. A. Miller, C. W. Johnson, and T. Zhao. A Probabilistic Model Checking Approach to Analysing Reliability, Availability, and Maintainability of a Single Satellite System. In *Proceedings of the 7th European Modelling Symposium (EMS 2013)*, pages 611–616. IEEE, 2013.

[116] Z. Peng, Y. Lu, A. A. Miller, C. W. Johnson, and T. Zhao. Risk Assessment of Railway Transportation Systems using Timed Fault Trees. *Quality and Reliability Engineering International*, 32(1):181–194, February 2014.

[117] Z. Peng, Y. Lu, A. A. Miller, C. W. Johnson, and T. Zhao. Formal Specification and Quantitative Analysis of a Constellation of Navigation Satellites. *Quality and Reliability Engineering International*, 32(2):345–361, March 2016.

[118] G. D. Penna, B. Intrigila, I. Melatti, E. Tronci, and M. V. Zilli. Exploiting transition locality in automatic verification of finite-state concurrent systems. *International Journal on Software Tools for Technology Transfer*, 6(4):320–341, 2004.

[119] W. S. Phlong and B. D. Elrod. Availability Characteristics of GPS and Augmentation Alternatives. *Navigation*, 40(4):409–428, 1993.

[120] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 46–57, 1977.

[121] M. Pontani and P. Teofilatto. Satellite Constellations for Continuous and Early Warning Observation: A Correlation-Based Approach. *Journal of Guidance, Control, and Dynamics*, 30(4):910–921, 2007.

[122] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi. Multi-Robot Planning: A Timed Automata Approach. In *Proceedings of ICRA 2014*, volume 5, pages 4417–4422. IEEE, 2004.

[123] N. Ramsey and A. Pfeffer. Stochastic Lambda Calculus and Monads of Probability Distributions. *ACM SIGPLAN Notices*, 37(1):154–165, 2002.

[124] M. Rausand and A. Høyland. *System Reliability Theory: Models, Statistical Methods, and Applications*. John Wiley & Sons, Hoboken, New Jersey, 2nd edition, 2009.

[125] D. Reijsbergen, S. Gilmore, and J. Hillston. Patch-based Modelling of City-centre Bus Movement with Phase-type Distributions. *Electronic Notes in Theoretical Computer Science*, 310:157–177, January 2015.

[126] A. Richards, T. Schouwenaars, J. P. How, and E. Feron. Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764, July-August 2002.

[127] J. Rushby. Formal Methods and their Role in the Certification of Critical Systems. In R. Shaw, editor, *Safety and Reliability of Software Based Systems*, pages 1–42. Springer London, 1997.

[128] M. Schlotterer and S. Novoschilov. On-Ground Path Planning Experiments for Multiple Satellites. In *Proceedings of the 23rd International Symposium on Space Flight Dynamics*, pages 1–12, 2012.

[129] M. Shaw. GPS Modernization: On the Road to the Future GPS IIR/IIR-M and GPS III. In *Proceedings of the International Symposium of Global Navigation Satellite Systems (IGNSS 2009)*, Queensland, Australia, 2009.

[130] Z. Shen. Model Checking for the MPL Entry and Descent Sequence. Technical report, Department of Aerospace Engineering, Iowa State University, 2001.

[131] G. M. P. Simões. Rams analysis of railway track infrastructure. Master's thesis, University of Lisbon, 2008 September.

[132] G. Taylor, J. Li, D. Kidner, C. Brunsdon, and M. Ware. Modelling and prediction of GPS availability with digital photogrammetry and LiDAR. *International Journal of Geographical Information Science*, 21(1):1–20, 2007.

[133] P. J. G. Teunissen and A. Kleusberg. GPS Observation Equations and Positioning Concepts. In A. Kleusberg and P. J. G. Teunissen, editors, *GPS for Geodesy*, volume 60 of *Lecture Notes in Earth Sciences*, pages 175–217. Springer Berlin Heidelberg, 1996.

[134] J. M. T.P.Khanh Nguyen, Julie Beugin. RAMS analysis of GNSS based localisation system for the train control application. In *Proceedings of the 2nd International Conference on Computing, Management and Telecommunications (ComManTel 2014)*, pages 101–106. IEEE, 2014.

[135] Y. Ulybyshev. Geometric analysis of low-earth-orbit satellite communication systems: covering functions. *Journal of Spacecraft and Rockets*, 37(3):385–391, 2000.

[136] Y. Ulybyshev. Satellite Constellation Design for Complex Coverage. *Journal of Spacecraft and Rockets*, 45(4):843–849, 2008.

[137] C. von Essen and D. Giannakopoulou. Analyzing the Next Generation Airborne Collision Avoidance System. In E. Ábrahám and K. Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *Lecture Notes in Computer Science*, pages 620–635. Springer Berlin Heidelberg, 2014.

[138] J. G. Walker. Some circular orbit patterns providing continuous whole earth coverage. *Journal of the British Interplanetary Society*, 24(11):369–384, 1971.

[139] H.-S. Wang and P.-C. Hsiao. GNSS Availability Analysis in Taiwan - a Markov Model Approach. In *Proceedings of the National Technical Meeting of The Institute of Navigation*, pages 759–769, 2006.

[140] W. Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18:293–297, 1951.

[141] Q. Xin, S. J. Thomas J. E. Schwarz, and E. L. Miller. Disk infant mortality in large storage systems. In *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2005)*, pages 125–134. IEEE, 2005.

[142] L. Xing and J. B. Dugan. Analysis of generalized phased-mission system reliability, performance, and sensitivity. *IEEE Transactions on Reliability*, 51(2):199–211, 2002.

[143] S. Yovine. KRONOS: a verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer*, 1(1):123–133, December 1997.

[144] Y. Zhao and K. Y. Rozier. Formal specification and verification of a coordination protocol for an automated air traffic control system. *Science of Computer Programming*, 93:337–353, December 2014.

[145] Y. Zhao and K. Y. Rozier. Probabilistic Model Checking for Comparative Analysis of Automated Air Traffic Control Systems. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2014)*, pages 690–695. IEEE, 2014.

[146] K.-J. Zhu, J.-F. Li, and H.-X. Baoyin. Satellite scheduling considering maximum observation coverage time and minimum orbital transfer fuel cost. *Acta Astronautica*, 66(1-2):220–229, 2010.