Uncertainty Analysis of Phased Mission Systems with Probabilistic Timed Automata

Zhaoguang Peng
Reliability Research and Analysis Center
China Ceprei Laboratory
Guangzhou, China
Email: ghuapeng@gmail.com

Yu Lu*
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
* Email: y.lu.3@research.gla.ac.uk

Alice Miller
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
Email: alice.miller@glasgow.ac.uk

***Abstract*—A phased mission is one in which the requirements may alter over time. We present a novel approach to analyse phased mission systems using probabilistic timed automata (PTA). We show how to construct PTA models which allow one to reflect system uncertainty, and how to analyse these models using the PRISM probabilistic model checker. We illustrate our approach via a simple case study, namely path planning for a Mars exploration rover, since the mission of the rover can be expected to be an instance of phased mission systems.**

***Keywords*—phased mission systems; uncertainty analysis; formal methods; probabilistic timed automata; probabilistic model checking**

I. Introduction

A phased mission [1] is one in which the requirements may alter over time. Mission success depends on the success of all of the individual phases. For example, consider an aircraft whose operation consists of a number of phases, including: taxiing, taking off, climbing to the correct altitude, cruising, descending, landing, and taxiing back to the terminal. Uncertainty analysis allows us to predict phase failure probabilities and thus compute the overall probability of mission failure.

In a phased mission system (PMS), different phases may have different logical structures. The system unit may be involved in different tasks, and vary in each phase. For example, space monitoring and control communication system refers to the systems which track, monitor and control the various stages of the spacecraft and its payload. Its reliability is directly related to successfully launching spacecraft, changing orbit and other tasks.

Many aerospace applications are PMSs. Communication monitoring varies between the stages but in order to understand the overall communication program, communication within each stage must be analysed.

In recent years the analysis of PMSs has developed into an important research area. Some researchers have focused on the problem of phase independence, mainly using Fault Tree Analysis and state-space-based analytical methods, including Markov methods and Petri Nets [2], [3]. But, with the growing number of the system construction, it is more difficult to be applied to these systems.

Other approaches for analysing PMSs involve simulation and testing. Simulation is insufficient for analyising this type of complex system since it only allows for partial coverage. We consider a numerical analysis technique based on Probabilistic Timed Automata (PTAs). PTAs allow us to model systems which have both probabilistic and timed behaviours, and to analyze them with respect to a range of associated properties.

978-1-5090-0382-2/16/$31.00 ©2016 IEEE

In this paper, we propose a method based on PTAs to evaluate quantitative properties of PMSs. This method allows us to solve specific problems in the practical operation of PMS systems. In general, traditional analysis methods do not allow for the modelling of *uncertainty* within a system. For example, they do not allow for uncertainty in event ordering, or in the timing of events. It might not be possible to determine a priori the exact time at which an event will occur, it is more likely that a time interval during which the event would occur (with a given probability) could be predicted. Without the inclusion of this type of uncertainty, PMS planning is unreliable, and consequently PMSs may be unpredictable and unsafe.

This paper describes how PTAs can be used to model PMSs with uncertainty. The aim of the paper is to promote the adoption of such an approach for modelling PMSs. PTAs are analysed using probabilistic model checking. Probabilistic model checking involves creating a probabilistic model (based on a Markov chain) of a system. In this model events have probabilities and transistions may have associated rewards. Properties of the overall system can be verified using probabilistic stochastic logic. Examples of such properties involve the maximum or minimum probabilities of the likelihood of a situation occurring within a given time interval, or the expected cumulative reward.

The method is described in several steps as follows. First, we introduce the concept of a PMS and its technology. Second, we give technical background on traditional techniques and time dependent techniques. Third, we introduce probabilistic model checking technology, and present the definition and formal specification of quantitative properties of PTAs. Then, a case study is provided to illustrate the approach, and we use formal methods to analyse the PTA models. Finally we present our conclusions and outline directions for future research.

II. Analysis Techniques

A. Traditional Techniques

In order to analyse PMSs, a number of analysis techniques have been developed, such as Event Tree Analysis (ETA),

Fault Tree Analysis (FTA) [4], and Binary Decision Diagrams (BDDs) [5], [6]. We outline some of these techniques below, for more details see [7].

ETA involves determining a trace of events that lead to a failure/accident (but not the specific event that led to it). On the other hand, FTA is a method that examines relationships between the subsystems and components of a system in order to find the root cause of a failure [8]. A graphical representation of the different combinations of events that lead to failure is used to identify failure causes, be they due to software error, user error, or otherwise [9]–[11]. FTA analysis is carried out in two ways. One of these involves the identification of sets of component failures that can, when combined, lead to system failure. These sets are known as *cut sets*, and FTA involves logically determining minimal cut sets `MCSs` that lead to a specific error event (the *Top Event*). It is also possible to determine the probability of each node in the fault tree [12], [13].

Binary Decision Diagrams (BDDs) have been used in the context of FTA in order to allow for the efficient verification of more expressive properties [14]. BDDs are an effective method for describing Boolean expressions, applied in system design and certification. In most cases, the use of BDDs to describe Boolean expressions uses less storage space than other methods. Whereas any BDD can be translated to an FTA model, the converse is not true. FTA involves more factors than can be modelled using a BDD, such as human factors and environmental factors. BDDs have often been used to analyse fault trees. The combination of FTA, with its rigorous logical structure, tree pattern, and ability to reveal fault causes, and BDDs which allow for the quantitative calculation of failure probability, is a powerful technique when applied in the context of reliability engineering. When BDDs are used in the context of FTA, the fault tree is not analysed directly, but rather an abstracted BDD representation. For this reason translation errors may be introduced [15].

B. Real Time Techniques

There have been efforts in the past to introduce the notion of real time to FTA. For example, in [16], a time-dependent methodology for fault tree analysis is proposed in which time is expressed as a function of the available information.

In other work, FTA has been augmented using temporal formulas, time dependencies and temporal fault trees [17]–[19]. The last two of these are extensions of traditional fault trees and designed for the safety analysis of safety critical systems.

Another popular methods that allow one to model time is Stochastic Petri Nets (SPNs). SPNs allow one to calculate aggregated performance values (in a similar way to rewards in probabilistic model checking). Typical values that are measured are the average number of tokens and average token delay [20]. Transition delays are assumed to be random variables from a negative exponential distribution.

Markov models (such as Discrete Time Markov Chains (DTMCs), Markov Decision Processes (MDPs), and Continuous Time Markov chains (CTMCs)) consist of a countable set of states together with a probabilistic transition matrix which determines the probability of moving between states (or the associated *rate*, drawn from a probability distribution, for CTMCs). MDPs allow us to also model non-deterministic transitions. Markov models allow us to verify a range of probabilistic properties, what is the maximum (minimum) probability that the system reaches a given state within a certain time period? Or: what is the expected fuel consumption in the first 10 seconds? Markov models cannot address all properties that can be verified using TFTs. An example of such a property is: how long does it take for an error to occur, after the initial failure has occured?

III. Probabilistic Model Checking

In this paper, we present the analysis of probabilistic timed automata using the PRISM probabilistic model checker [21].

A. Probabilistic Timed Automata

Full details about probabilistic timed automata (PTAs) can be found in [22], [23]. We outline the important aspects in this section.

PTAs allow us to use the real-valued clocks of timed automata, together with the discrete probabilistic choice of MDPs. PTAs have real-valued clocks and, like MDPs [24] and therefore allow us to model systems with a range of different characteristics (non-determinism, probability, and real time).

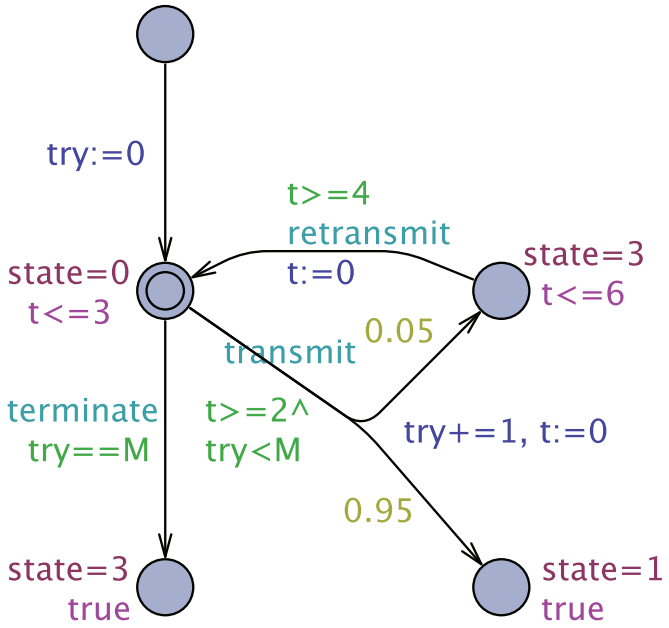


Fig. 1. An example of a probabilistic timed automaton.

In this section, we illustrate a number of basic PTA concepts using the example in Fig. 1. 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 "$s = 1$", when 4 to 6 time units have elapsed from whenever the message is lost, the sender tries to re-transmit the message.

Definition 1. *A probabilistic timed automaton PTA 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* $inv : L \rightarrow CC(\chi)$ *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 $\mathrm{t} < 3$ and $\mathrm{t} < 6$ 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 [25], the FireWire root contention protocol [26], and the IPv4 Zeroconf protocol [22].

B. The Probabilistic Model Checker PRISM

PRISM is a probabilistic model checker, which can be used to model, and verify systems that exhibit random or probabilistic behaviour based on different types of probabilistic models, such as DTMCs, CTMCs, MDPs, and PAs. PRISM has recently incorporated support for the analysis of 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. It has been used to analyse many different industrial applications, such as communication and multimedia protocols, security protocols, dynamic power management, biological systems, and autonomous systems.

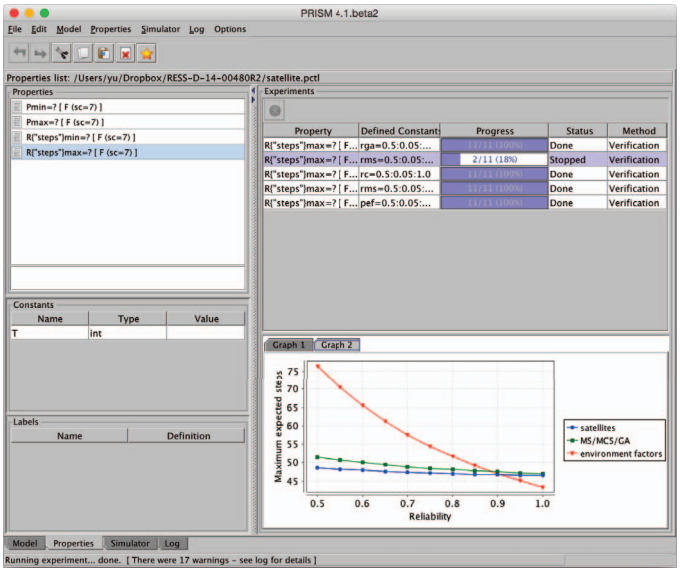


Fig. 2. A screenshot of the PRISM probabilisitic model checker.

In Fig. 2, 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*.

In order to verify a system using PRISM, models are specified using a defined language (PRISM) in which different components are defined as *modules*. A model of the entire system is constructed by composing a number of modules in a synchronous way. A module has the form:

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

and consists of a list of variable declarations, followed by a list of *commands*. The *state* of a model at any time is a snapshot of the system consisting of an instantaneous valuation of all the variables in the specification.

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) [27].

In Fig. 3 we illustrate the PRISM language by specifying the example from Fig. 1. Note the use of rewards: a reward of 3.5 is accumulated at state $s = 0$ whenever a message is transmitted.

C. Continuous Stochastic Logic

In this paper our properties are expressed using Continuous Stochastic Logic (CSL) [28], [29]. This logic is an extension

```
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' = 3) & (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;
endrewards
```

Fig. 3. The PRISM code of the probabilisitic timed automaton in Fig. 1.

of: Computation Tree Logic (CTL) [30]; probabilistic CTL (PCTL), which is an extension of CTL for discrete time stochastic systems (PCTL) [31]; and Timed CTL, which is an extension of CTL for continuous time non-stochastic systems (TCTL) [32]. Formulas in CSL are either state formulas (which can be either true or false in any given state), and path formulas which apply to entire paths in a model.

Definition 2. *Let $a \in AP$ 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 AP is defined inductively as follows:*

- *$true$ is a state-formula.*
- *Each $a \in AP$ 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 $\mathcal{S}_{\bowtie p}(\Phi)$.*
- *If φ is a path formula, then $\mathcal{P}_{\bowtie p}(\varphi)$.*
- *If Φ and Ψ are state formulas, then $\mathcal{X}_I\Phi$ and $\Phi\mathcal{U}_I\Psi$ are path formulas.*

Full details of CSL can be found in [28], [29]. Two of the main type of CSL properties are:

- $\mathcal{S}_{\bowtie p}(\Phi)$: the steady-state probability for a state satisfying Φ satisfies . An example property is $\mathcal{S}_{\leq 0.5}(x == 4)$.
- $\mathcal{P}_{\bowtie p}(\varphi)$: the probability measure of the paths satisfying path property φ satisfies $\bowtie$ p. An example property is $\mathcal{P}_{\geq 0.8}(p\mathcal{U}q)$ where p and q are defined atomic propositions and $\mathcal{U}$ the until operator.

Another important class of property involves the $\mathbf{P}$ operator, which indicates the probability of an event. For example, in this paper we use this operator within PRISM to calculate the probability of a robot completing a task within 30 minutes and within 15 minutes respectively.

It is also possible to express properties to measure cumulative rewards using the $\mathbf{R}$. We do not use properties of this form in this paper, but examples of such properties can be found on the Property Specification section of the PRISM website.

IV. Formal Analysis

A. Formal Modelling of A Mars Exploration Rover

In this section, we present a case study to illustrate the role of PTAs and probabilistic model checking for the analysis of PMSs: a Mars exploration rover [33] exploring an unknown region. The mission of Mars exploration rover can be expected to be an instance of phased mission systems [34].

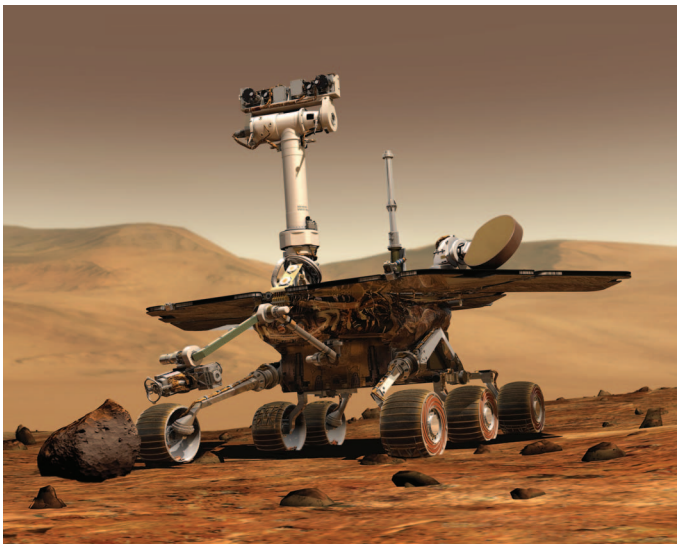


Fig. 4. NASA Mars exploration rover.

In Fig. 4, we illustrate the Mars exploration rover. The key purpose of the mission of the Mars exploration rover is to make it to pre-defined positions to undertake practical and systematic explorations. Specifically, the Mars exploration rover is planned to achieve different day-to-day conventional missions. Based on [34], a number of major missions can

TABLE I. 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	$2 \sim 5$	$3 \sim 6$	–	$4 \sim 5$	$5 \sim 7$	$6 \sim 8$	$4 \sim 5$	–	–	–

summarised as follows: (1) waking up at a specific time, (2) receiving commands from the base on Earth, (3) navigating to a specific destination, (4) carrying out surface operations, (5) processing data and sending it to the base (downlink), and (6) entering a sleep mode. The routine operations can consist other phases to be performed according to individual mission, we only analyse the underlying 6 phases that are commonly used.

This example is common in the field of artificial intelligence. The rover attempts to reach a destination by following instructions from a controller. In this case, a rover moves to the destination by command. A region is divided into 12 parts. These consist of 11 numbered parts (numbered from 0 to 10) and one blocked region which cannot be passed by the rover (indicated by shading in Fig. 5).

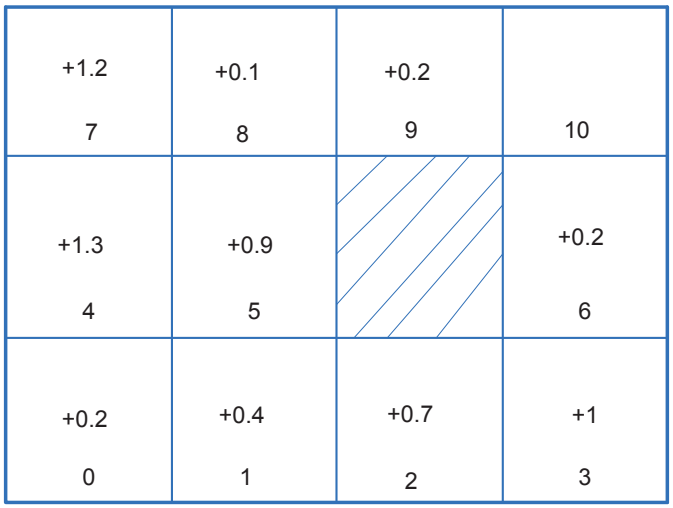


Fig. 5. Energy Consumption of the Mars exploration rover in Each Area.

Suppose that the robot attempts to move from area 0 to area 10. The condition of the area must be evaluated, and the route to area 10 planned. The robot starts from area 0, at which point its explore direction is uncertain. The movement of the rover is controlled externally by a human. However, due to uncertainty of control, the robot will not always respond correctly. At any point there is a 10% probability of deviation to the left and 10% probability of deviation to the right.

In this case, there are two further parameters: time and energy. The time the robot spends in each area moving in each direction varies. The time spent in each area is given as an interval between specified minimal and maximal times. The energy the robot uses in each area also varies. Fig. 5 indicates the energy consumption in each area. *Table I* shows the time spent by the robot in each area moving in each direction.

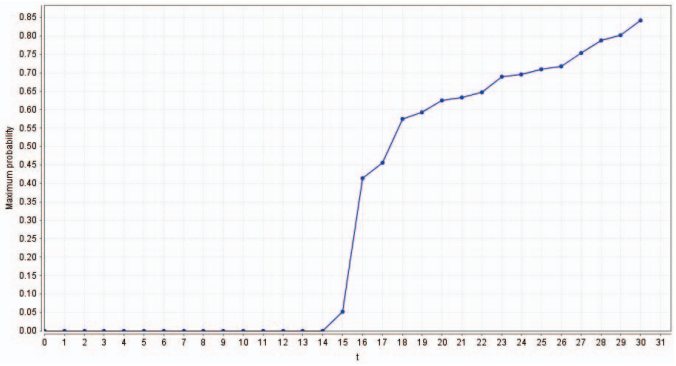


Fig. 6. Probability distribution of the Mars exploration rover completing the task within a given time.

On collision with the boundary of the obstacle and the perimeter of the exploration area, the robot will be bounced back, and have to re-select the path of the route. For example, when the robot gets the command that it should move forward to area 4 from area 0, there is an 80% probability that it will complete the task, a 10% probability that it will hit the exploration perimeter and return to area 0, and a 10% probability that it will move to area 1.

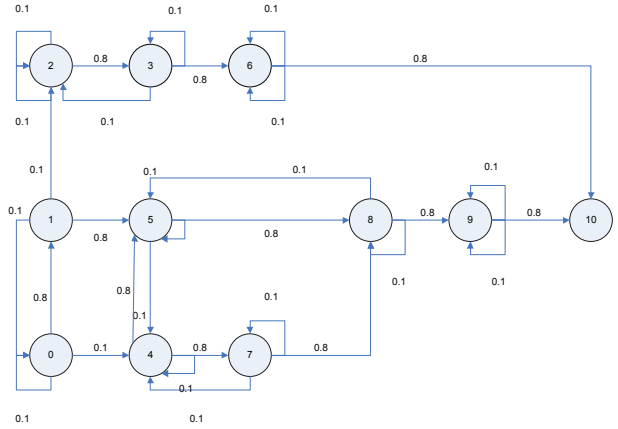
Fig. 8 shows the PRISM code for the PTA models shown in Fig. 7. It includes a reward declaration, denoted as "energy", to create a priced probabilistic timed automaton that associates states with different cost rates.

B. Uncertainty Analysis

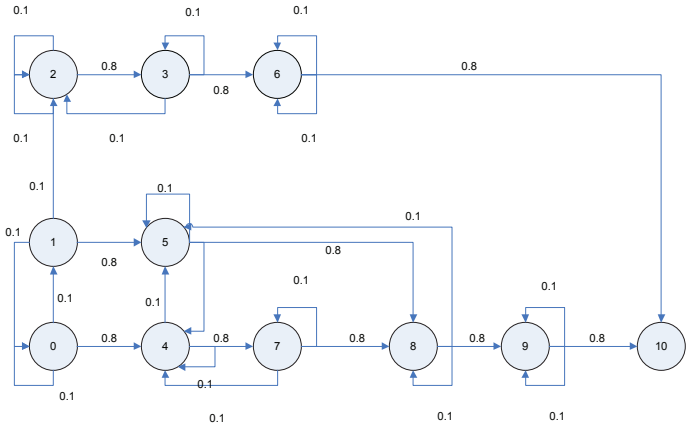
Using the model checker PRISM, we show that the probability of the robot completing the task within 30 minutes is .84134. The probability of the robot completing the task within 15 minutes is .05184. Fig. 6 shows the relationship between the probability and time to complete the task.

The PTA model in the case of minimal energy consumption is shown in Fig. 7(a). The route of the robot in this case is as shown in Fig. 9(a). The optimal route is: $0 \rightarrow 1 \rightarrow 5 \rightarrow 8 \rightarrow 9 \rightarrow 10$. The minimum energy consumption of the robot moving from area 0 to area 10 is 13.3W. If the direction of the robot deviates, the robot should reroute from its new position in such a way as to minimise energy consumption.

The PTA model in the case of minimal time consumption is as shown in Fig. 7(b). The shortest path is as shown in Fig. 9(b). The best route is: $0 \rightarrow 4 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10$.



(a) Probabilistic timed automaton w.r.t. minimum energy consumption.



(b) Probabilistic timed automaton w.r.t. minimal time.

Fig. 7. Probabilistic timed automata (PTAs) models of the Mars exploration rover.

```
pta
module rover
        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 = 0 => x <=a ) & (s = 1 => x <= b) &
        (s = 2 => x <=c ) & (s = 3 => x <= d) & (s = 4 => x <=e ) &
        (s = 5 => x <= f) & (s = 6 => x <=g ) & (s = 7 => x <= h) &
        (s = 8 => x <=i ) & (s = 9 => x <= j)
endinvariant
        [] s =0 & x >= 3 -> 0.8 : (s' = 4) + 0.1 : (s' = 0) + 0.1 : (s' = 1);
        [] s = 4 & x >= 2 -> (s' = 4) & (x' = 0);
        [] s = 4 & x >= 2 -> 0.05 : (s' = 0) + 0.05 : (s' = 4) + 0.1 : (s' = 5) +
                             0.8 : (s'=7);
        [] s = 0 & x >= 2 -> (s' = 0) & (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
```

Fig. 8. The PRISM code for the the Mars exploration rover.

Similarly, the time the robot spends moving from area 0 to area 10 is at least 22.8 minutes. There is one best direction to take in each area. Once the robot enters an area, it should reroute from its new position in such a way as to minimise time consumption.

V. Conclusion and Future Work

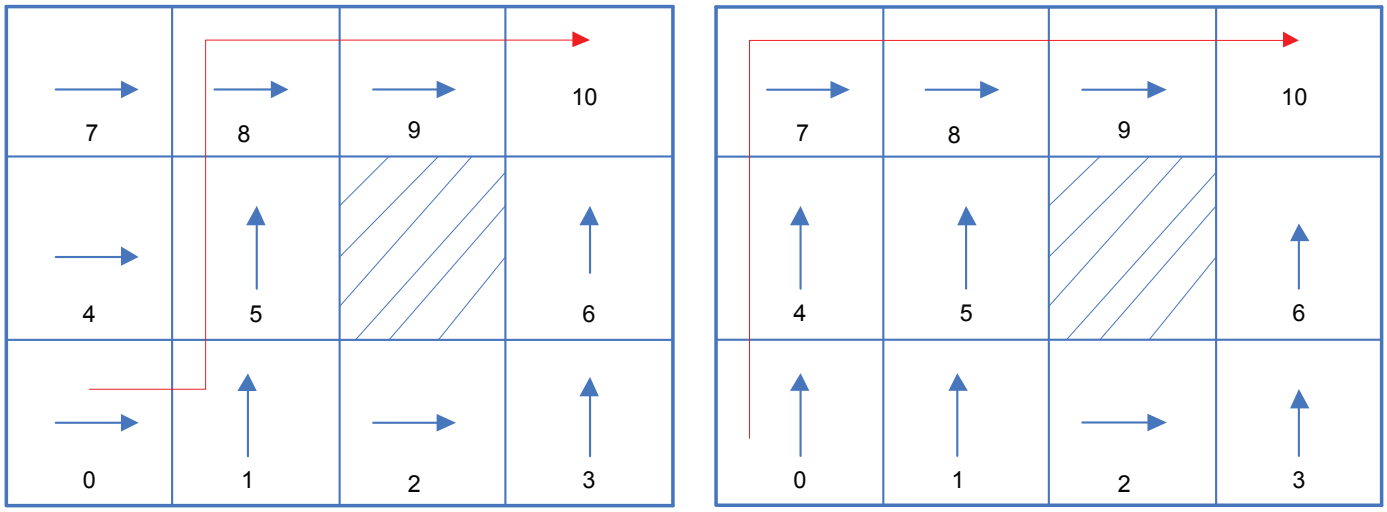
In this paper we consider an automated verification approach: probabilistic model checking, to analyse phased mission systems. A Probabilistic timed automaton is used to solve an uncertainty problem in PMS analysis. Using an example, we show how to construct PTAs for PMSs and use probabilistic model checking techniques to analyse the resulting models. The aim of the paper is to promote the adoption of PTAs and probabilistic model checking within this context. We have illustrated our approach using a simple case study involving a Mars explorer robot, for which we verify quantitative properties using the PRISM model checker.

References

[1] G. R. Burdick, J. B. Fussell, D. M. Rasmuson, and J. R. Wilson, "Phased Mission Analysis: A Review of New Developments and An Application," *IEEE Transactions on Reliability*, vol. R-26, no. 1, pp. 43–49, April 1977.

[2] S. Chew, S. Dunnett, and J. Andrews, "Phased mission modelling of systems with maintenance free operating periods using simulated petri nets," *Reliability Engineering & System Safety*, vol. 93, no. 7, pp. 980–994, July 2008.

[3] I. Mura and A. Bondavalli, "Markov Regenerative Stochastic Petri Nets to Model and Evaluate Phased Mission Systems Dependability," *IEEE Transactions on Computers*, vol. 50, no. 12, pp. 1337–1351, December 2001.



(a) Minimum energy consumption.

(b) Minimum time

Fig. 9. Results of the Mars exploration rover to reach a given target.

[4] F. I. Khana and T. Husaina, "Risk Assessment and Safety Evaluation Using Probabilistic Fault Tree Analysis," *Human and Ecological Risk Assessment: An International Journal*, vol. 7, no. 7, pp. 1909–1927, 2001.

[5] Y. Mo, F. Zhong, H. Liu, Q. Yang, and G. Cui, "Efficient Ordering Heuristics in Binary Decision Diagram based Fault Tree Analysis," *Quality and Reliability Engineering International*, vol. 29, no. 3, pp. 307–315, April 2013.

[6] H.-Z. Huang, H. Zhang, and Y. Li, "A New Ordering Method of Basic Events in Fault Tree Analysis," *Quality and Reliability Engineering International*, vol. 28, no. 3, pp. 297–305, April 2012.

[7] Z. Peng, Y. Lu, A. Miller, C. Johnson, and T. Zhao, "Risk Assessment of Railway Transportation Systems using Timed Fault Trees," *Quality and Reliability Engineering International*, 2014.

[8] V. R. Renjith, G. Madhu, V. L. G. Nayagam, and A. B. Bhasic, "Two-dimensional fuzzy fault tree analysis for chlorine release from a chlor-alkali industry using expert elicitation," *Journal of Hazardous Materials*, vol. 183, no. 1-3, pp. 103–110, November 2010.

[9] C. A. Ericson, *Hazard Analysis Techniques for System Safety*. Wiley-Blackwell, 2005.

[10] W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback, *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, 2002.

[11] Y.-T. Tsai, "Applying a case-based reasoning method for fault diagnosis during maintenance," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 223, no. 10, pp. 2431–2441, 2009.

[12] A. Volkanovski, M. Čepin, and B. Mavko, "Application of the fault tree analysis for assessment of power system reliability," *Reliability Engineering & System Safety*, vol. 94, no. 6, pp. 1116–1127, June 2009.

[13] G. Merle, J.-M. Roussel, J.-J. Lesage, and A. Bobbio, "Probabilistic Algebraic Analysis of Fault Trees With Priority Dynamic Gates and Repeated Events," *IEEE Transactions on Reliability*, vol. 59, no. 1, pp. 250–261, March 2010.

[14] R. Remenyte-Prescott and J. D. Andrews, "An Efficient Real-time Method of Analysis for Non-coherent Fault Trees," *Quality and Reliability Engineering International*, vol. 25, no. 2, pp. 129–150, March 2009.

[15] L. M. Bartlett and S. Du, "New Progressive Variable Ordering for Binary Decision Diagram Analysis of Fault Trees," *Quality and Reliability Engineering International*, vol. 21, no. 4, pp. 413–425, June 2005.

[16] W. Vesely, "A time-dependent methodology for fault tree evaluation," *Nuclear Engineering and Design*, vol. 13, no. 2, pp. 337–360, August 1970.

[17] J. Magott and P. Skrobanek, "A Method of Analysis of Fault Trees with Time Dependencies," in *Computer Safety, Reliability and Security*, ser. Lecture Notes in Computer Science, F. Koornneef and M. van der Meulen, Eds. Springer Berlin Heidelberg, 2010, vol. 1943, pp. 176–186.

[18] G. K. Palshikar, "Temporal fault trees," *Information and Software Technology*, vol. 44, no. 3, pp. 137–150, March 2002.

[19] J. Magott and P. Skrobanek, "Method of time Petri net analysis for analysis of fault trees with time dependencies," *IEE Proceedings - Computers and Digital Techniques*, vol. 149, no. 6, pp. 257–271, November 2002.

[20] P. J. Haas, *Stochastic Petri Nets: Modelling, Stability, Simulation*. Springer-Verlag New York, 2002.

[21] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: Probabilistic Symbolic Model Checker," in *Computer Performance Evaluation: Modelling Techniques and Tools*, ser. Lecture Notes in Computer Science, T. Field, P. G. Harrison, J. Bradley, and U. Harder, Eds. Springer Berlin Heidelberg, 2002, vol. 2324, pp. 200–204.

[22] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston, "Performance analysis of probabilistic timed automata using digital clocks," *Formal Methods in System Design*, vol. 29, no. 1, pp. 33–78, July 2006.

[23] G. Norman, D. Parker, and J. Sproston, "Model checking for probabilistic timed automata," *Formal Methods in System Design*, vol. 43, no. 2, pp. 164–190, October 2013.

[24] 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 *Formal Methods for Industrial Critical Systems: A Survey of Applications*, S. Gnesi and T. Margaria, Eds. Wiley, 2012, ch. 7, pp. 133–150.

[25] 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*, vol. 128, no. 6, pp. 195–214, May 2005.

[26] 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*, vol. 14, no. 3, pp. 295–318, April 2003.

[27] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of Probabilistic Real-Time Systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, G. Gopalakrishnan and S. Qadeer, Eds. Springer Berlin Heidelberg, 2011, vol. 6806, pp. 585–591.

[28] C. Baier, J.-P. Katoen, and H. Hermanns, "Approximative Symbolic Model Checking of Continuous-Time Markov Chains," in *CONCUR'99 Concurrency Theory*, ser. Lecture Notes in Computer Science, J. C. M.

Baeten and S. Mauw, Eds. Springer Berlin Heidelberg, 1999, vol. 1664, pp. 146–161.

[29] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Model-Checking Continuous-Time Markov Chains," *ACM Transactions on Computational Logic*, vol. 1, no. 1, pp. 162–170, July 2000.

[30] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Elsevier, 1990, pp. 995–1072.

[31] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994.

[32] 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)*. IEEE, 1990, pp. 414–425.

[33] J. K. Erickson, "Living the Dream - An Overview of the Mars Exploration Project," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 12–18, June 2006.

[34] S. Ballerini, L. Carnevali, M. Paolieri, K. Tadano, and F. Machida, "Software Rejuvenation Impacts on a Phased-Mission System for Mars Exploration," in *Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW 2013)*. IEEE, 2013, pp. 275–280.