Probabilistic model checking in practice: Case studies with PRISM[1]

Marta Kwiatkowska, Gethin Norman and David Parker
School of Computer Science, University of Birmingham,
Edgbaston, Birmingham, B15 2TT, UK

Abstract

In this paper, we describe some practical applications of *probabilistic model checking*, a technique for the formal analysis of systems which exhibit stochastic behaviour. We give an overview of a selection of case studies carried out using the probabilistic model checking tool PRISM, demonstrating the wide range of application domains to which these methods are applicable. We also illustrate several benefits of using formal verification techniques to analyse probabilistic systems, including: (i) that they allow a wide range of numerical properties to be computed accurately; and (ii) that they perform a complete and exhaustive analysis enabling, for example, a study of best- and worst-case scenarios.

1 Introduction

Probabilistic model checking is a formal verification technique for the analysis of systems which exhibit stochastic behaviour. As with traditional formal verification techniques, it involves the construction of a precise mathematical model of a real-life system to be analysed, formal specification of one or more properties of this system, and then analysis of these properties based on an exhaustive exploration of the constructed model.

The use of probabilistic model checking is motivated by the fact that there are many instances of real-life systems whose behaviour can only be accurately modelled by considering their stochastic characteristics. One example is algorithms which make random choices based on coin tosses to give simple, elegant solutions to many distributed coordination problems, e.g., leader election. These algorithms can nowadays be found in real-world protocols in domains such as network communication and security. Other real-life systems can be inherently stochastic in nature because they include components which are known to be unreliable, e.g., fault-tolerant architectures, or because the exact timing of inputs to system remain unpredictable, e.g., computer networks, manufacturing systems or biological processes.

The three types of models most commonly employed in probabilistic model checking are discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs). The model used will depend on the nature of the system being studied. DTMCs provide a comparatively simple model for systems where the exact probability of different behaviours at each discrete time-step is known. MDPs extend DTMCs with nondeterminism, which can be used to model concurrency between processes operating in parallel or for underspecification, where exact values for some system parameters are unknown. CTMCs, on the other hand, permit specification of events which happen in real-time, by modelling delays as exponential distributions.

Properties to be analysed by probabilistic model checking are typically specified using temporal logics such as PCTL or CSL, probabilistic extensions of the classical temporal logic CTL. These properties relate not just to the functional correctness of a system, as is the case with non-probabilistic formal verification techniques, but also to quantitative measures such as performance and reliability, for example: "the probability that a message will be delivered within 30ms is at least 0.75"; or "the probability of shutdown occurring is at most 0.01".

Furthermore, instead of following the approach used in traditional verification of expressing properties to which the answer is "yes" or "no", it is usually more beneficial to analyse, for example: "the probability that an error has occurred within T seconds" for a range of values of T. Additionally, by augmenting probabilistic models with information about the costs that are incurred during the execution of the system, we can analyse for example: "the expected time for N processes to successfully elect a leader"; or "the expected power consumption when the arrival rate of jobs is λ". As we will see later, analysing such properties for a range of parameter values (e.g., for T, N and λ above) is often key to identifying interesting or anomalous behaviour.

There are now a number of tools available for probabilistic model checking, e.g., PRISM [12, 1], $\mathsf{E}\vdash\mathsf{MC}^2$ [9], and Rapture [10]. In this paper we use PRISM, which offers support for all the types of models and properties discussed above and which features sophisticated data structures designed to reduce the time and space requirements of verification.

PRISM has a high-level description language which is used to describe models to be analysed. Unlike simulation tech-

[1]This work was supported in part by EPSRC grants GR/N22960 and GR/S11107 and FORWARD.

niques, which are another very common analysis method for stochastic systems, the first step a probabilistic model checker such as PRISM performs is to construct, from a description in this language, the entire probabilistic model, based on an exhaustive search of its state space. Based on this model, property analysis is then carried out via a combination of numerical computation methods, such as solving linear equation systems or linear optimisation problems, and graph-based algorithms, such as reachable state exploration. Although this will inevitably have an effect on the size of models which can be studied, such exhaustive analysis is one of the true strengths of formal verification. Firstly, the measures computed will be exact, rather than approximations based on a large number of simulations. Secondly, it is feasible to assert more complete, exhaustive conclusions, e.g., computing the best-/worst-case performance for all possible initial configurations of a system, all possible values of some model parameter, or all possible scheduling of parallel components.

In the remaining sections of this paper, we describe six case studies which illustrate the strengths of probabilistic model checking. Through these examples, we aim to demonstrate that this technique and, in particular, the tool PRISM:

- are applicable, and indeed useful, in a wide range of application domains,
- are expressive enough to analyse properties of genuine interest to system designers,
- have successfully identified interesting and anomalous behaviour in several real-life systems.

For a more detailed introduction to the field of probabilistic model checking see, for example, [27, 2]. For further information about all the examples described in this paper, see the case studies section of the PRISM website [1].

2 Self-stabilisation algorithms

A *randomised algorithm* is an algorithm whose execution can depend on the result of random choices, for example based on electronic coin tosses. Randomised algorithms can provide simple, elegant and fast solutions to a wide range of distributed coordination problems. In some cases, randomised algorithms not only outperform their deterministic counterparts, but also provide a solution where no corresponding deterministic algorithm exists.

Randomised distributed algorithms, i.e., those which operate on a number of asynchronous parallel processes, are often particularly difficult to analyse because of non-trivial interactions between the probabilistic behaviour of each process and the nondeterminism arising from concurrency between them. This makes automated formal analysis techniques an attractive option. Probabilistic model checking has been applied, for example, to randomised distributed algorithms for the problems of consensus [14], Byzantine agreement [11], mutual exclusion [1] and leader election [6, 1].

Here, we show an example from a class of algorithms called *self-stabilisation algorithms*, whose aim is to transform a system from an *unstable* state to a *stable* state in a finite number of steps. Our example is the algorithm of Herman [8] for a set of processes which are arranged in a ring and can each possess a token. Tokens can be passed unidirectionally around the ring, and when two tokens meet they are both eliminated. A stable state is one in which there is only a single token. At every step of the algorithm, each process with a token decides whether to keep it or pass it on based on the outcome of a random coin toss; processes without a token do nothing.

Since the processes operate synchronously, a PRISM model of the algorithm is constructed as a DTMC. This can first be used to verify the property “a leader is always eventually elected with probability 1”. Secondly, by assigning a cost of one unit to each step of the algorithm, PRISM can be used to compute “the expected time (number of steps) for self-stabilisation to complete”. Furthermore, PRISM can calculate the worst-case performance for different classes of initial token configurations (more specifically, all configurations with K tokens, for different values of K). Figure 1 shows these worst-case expected times for a range of numbers of processes (N) and a range of values of K.

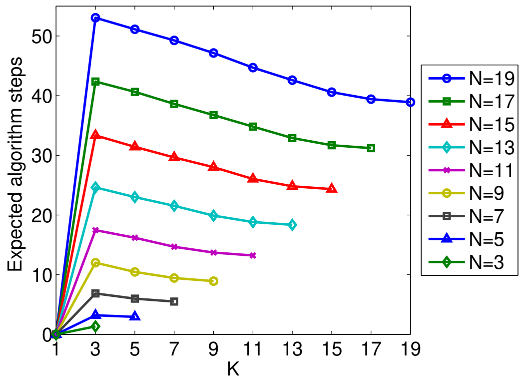

Figure 1: Worst-case expected completion times for Herman’s self-stabilisation algorithm with a ring of N processes and an initial configuration of K tokens.

To give an idea of scale, the total number of possible initial configurations for the case $N = 19$ is $528,288$. While PRISM constructs these cases as a single DTMC with multiple initial states and executes a single analysis of it, simulation, for example, would have to be performed separately for each initial state (i.e., half a million times) to obtain comparable results. Interestingly, for this case study, PRISM helps illustrate an unproven conjecture from [20] that the worst case execution time for this algorithm always results from the case where there are initially three tokens.

3 Root contention in IEEE 1394 FireWire

Another application domain where probabilistic model checking has proved extremely useful is that of probabilistic communication protocols. Thanks to the ever-growing number of both wired and wireless communication and multimedia devices in today's society, research into the performance of these protocols is attracting increasing interest. An analysis of their performance, however, is particularly challenging because it must incorporate aspects relating to probability (the protocols often incorporate randomisation), concurrency (the protocols are distributed between devices) and time (the protocols often have precise real-time requirements). Examples of the use of probabilistic model checking to analyse communication protocols include Bluetooth device discovery [4], IPv4 Zeroconf [13], IEEE 802.11 wireless LANs [15] and IEEE 802.3 CSMA/CD [17, 3].

Here, we discuss the example of the IEEE 1394 FireWire protocol, a standard for a high performance serial bus, aimed at connecting networks of multimedia devices. The protocol is designed to facilitate the addition and removal of devices from the network at any time. When this occurs, in order for the devices to arrange themselves into a tree, a leader election process is executed which selects a *root* device. This is done by propagating messages between neighbouring nodes, starting at leaf nodes of the network. It is possible for a situation to arise where two nodes contend to be the root node. The algorithm executed to resolve this is known as the *root contention protocol*. In short, this works by the introduction of delays between message transmissions, the lengths of which are determined randomly. More specifically, each time a node sends a message, it tosses a coin to choose between waiting for a 'short' or a 'long' delay before doing so.

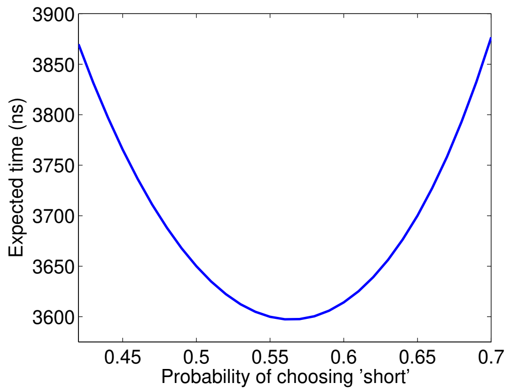

Figure 2: Performance (expected completion time) of the FireWire root contention protocol for a range of coin biases.

A PRISM model of FireWire root contention was constructed in [16] as an MDP. Probability in the model arises from the use of coin tosses to choose between time delays. Nondeterminism in the model results from multiple sources: firstly, from concurrency between contending nodes; and secondly, from underspecification in the official documentation regarding delay times (only lower and upper bounds are provided). PRISM has been used to analyse "the probability of a electing a leader within a given time bound" and "the expected time for leader election to complete". In fact, it computes the *minimum* probability or *maximum* expected time for all possible resolutions of nondeterminism (i.e., for all schedulings of parallel components and for all possible delay lengths which meet the IEEE specification). In this way, PRISM has been used to investigate an existing conjecture [29] that the performance of the protocol could be improved by using a biased coin. Figure 2 shows "the maximum expected time for leader election" for a range of values of the probability of choosing a 'short' delay. We see that the best performance is obtained indeed not from a fair coin, but one which selects a 'short' delay with probability approximately 0.56.

4 Probabilistic contract signing

Probabilistic protocols have also proved useful in the field of security. Non-probabilistic formal verification has been used with great success in the past to ascertain the correctness (and to identify flaws) of security protocols. More recently, probabilistic model checking has been used to examine probabilistic security protocols, including those for anonymity (Crowds) [28], non-repudiation [18], contract signing [24, 25] and fair exchange [25, 1].

Here, we discuss the probabilistic contract signing protocol of Even, Goldreich and Lempel [5]. This protocol is designed to allow two parties, A and B, to exchange commitments to a contract. In an asynchronous setting, it is difficult to perform this task in a way that is fair to both parties, i.e., such that if B has obtained A's commitment, then A will always be able to obtain B's. In the Even, Goldreich and Lempel (EGL) protocol, the parties A and B each generate a set of pairs of secrets which are then revealed to the other party in a probabilistic fashion. A is committed to the contract once B knows both parts of one of A's pairs (and vice versa).

PRISM was used to identify a weakness of the protocol [25, 1], showing that, by quitting the protocol early, one of the two parties (the one which did not initiate the protocol) can be at an advantage by being in possession of a complete pair of secrets while the other party knows no complete pairs. Various modifications to the basic EGL protocol were proposed [25, 1] and PRISM was used to quantify the fairness of each, i.e., "the probability of the protocol reaching a state where one party knows a complete pair of secrets but the other knows no complete pairs".

Figure 3 shows plots of these values for both the basic protocol (EGL) and three modifications (EGL2, EGL3 and EGL4). The solid lines and dashed lines represent the values for parties A and B, respectively (where process B initiated the protocol). The data is computed for a range of values of N, the number of pairs of secrets which each party generates. It can be seen that, for the original protocol (EGL), the probability of party A being unfairly advantaged is 1 for all values of N, whereas for the most successful modification (EGL4), the protocol performs equally fairly for each party.

PRISM can further be used to analyse more precisely the de-

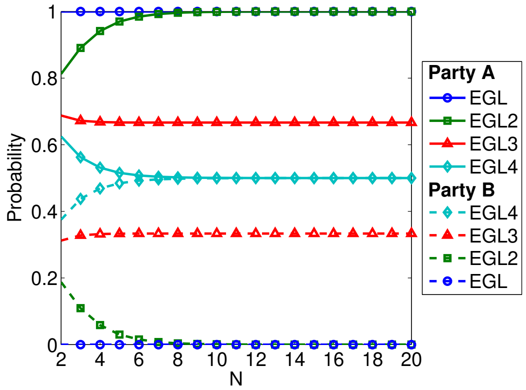

Figure 3: Probability of reaching a state where one party knows a complete pair of secrets but the other does not for the EGL protocol and three modified versions: solid lines for party A; dashed lines for party B.

gree to which a 'disadvantaged' party suffers, e.g., by computing "the expected number of messages that must be sent to get from a state where only one party knows a complete pair of secrets to one in which both parties do".

5 NAND multiplexing

This case study is an example of applying PRISM to the evaluation of reliability and redundancy properties of fault-tolerant systems in the field of computer-aided design. In particular, this concerns *multiplexing*, a technique due to von Neumann [30] for performing reliable computations with unreliable devices. Originally motivated by the use of unreliable valve-based computers, such techniques are again becoming relevant in the field of nanotechnology, where the small scales involved mean that components are inherently unreliable.

Taking the example of a NAND gate, a *multiplexed* NAND gate is constructed by duplicating N copies of each pair of inputs to the original NAND gate and permuting these randomly amongst N NAND gates operating in parallel. The result of the multiplexed gate is based on a consensus of the N individual gates. In this way, redundancy between the devices is used to minimise the effect of errors in the individual devices. Performance can be further improved by adding *restorative stages*, essentially replicating this process several times in attempt to reduce errors further.

Figure 4 shows a plot from [21] of the effectiveness of NAND multiplexing ($N = 60$), measured as "the probability that less than 10% of the outputs are erroneous", for varying numbers of restorative stages. Plots are given for four different values of λ, the failure rate of each individual gate. Through the construction of a formal model specification, as required in the probabilistic model checking approach, a flaw was detected in the analytical approach of [7] (dashed lines in Figure 4). The results show that this flaw can lead to both an under- and over-approximation of the reliability of the system.

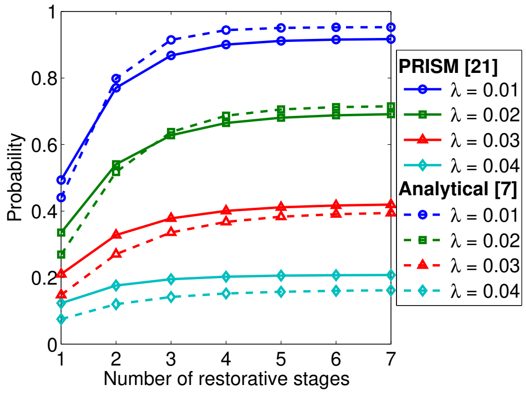

Figure 4: Performance (probability that less than 10% of outputs are erroneous) for multiplexed NAND gates with varying numbers of restorative stages and gate failure rates (λ): solid lines for [21] and dashed lines for [7].

6 Dynamic voltage scaling

Next, we consider the area of *power management*. This has become extremely relevant due to the increasing prevalence of battery-powered computing devices. In this domain, battery life and hence power efficiency are of utmost importance. Here, we consider a technique called *dynamic voltage scaling*, used in real-time embedded systems to achieve a compromise between battery life and performance. The technique is used to schedule a number of tasks which must be executed periodically. Each task has an associated period and a worst-case execution time. The voltage of the system can also be varied during scheduling, which has the effect of reducing the power consumption of the system. This will, however, slow down the execution of the current task. The aim is to schedule tasks and voltage changes in such a way that power consumption is minimised whilst ensuring that all tasks are executed within their deadlines.

PRISM has been used to model and analyse the performance of several scheduling schemes from [26]. The model used is an MDP. This incorporates probabilistic information because the actual execution time of each task is random (only a worst-case figure is known) and nondeterminism, which represents the fact that it is sometimes unspecified which task a scheduling scheme will pick. We can hence examine the worst-case behaviour of any implementation of each algorithm.

Figure 5 shows a comparison of "the maximum expected energy consumed by a given time bound" for four scheduling schemes (see [1] for more details). The actual cost measured is the square of the system's voltage, which is proportional to the energy consumed. The comparisons match those observed in [26], obtained through simulation.

Another probabilistic model checking case study in this area can be found in [22, 23], which studies stochastic dynamic power management strategies. Here, a wide range of properties can be analysed, e.g.: "the expected number of jobs awaiting service at time T", "the probability that 50 job re-

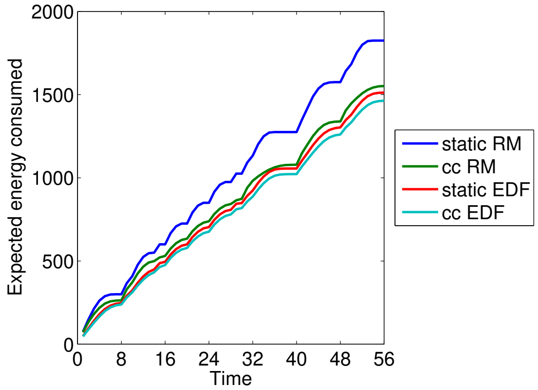

Figure 5: Expected energy consumption for four different dynamic voltage scaling scheduling schemes over time.

quests have been lost by time T" and "the expected long-run power consumption".

7 Biological process modelling

Lastly, we introduce an example from an application domain in which formal probabilistic analysis techniques are now beginning to be adopted: biological process modelling. Understanding the low-level interactions in complex biological processes is crucial to the development of many areas of biological and medical research. By constructing and analysing mathematical models of these systems, biologists can explore theories about how these complex reactions function.

In this example, we show results from a probabilistic model of cell cycle control in eukaryotes, a very commonly occurring class of single-celled or multi-cellular organisms. The results are taken from a PRISM model, based on the formal specification given in [19], which studies the relative concentration of a number of types of proteins, partaking concurrently in several complex chemical reactions. Since the timing characteristics of these reactions are accurately represented by exponential probability distributions, this system is modelled as a CTMC. The resulting plot, showing the amount of three of these proteins over time, is given in Figure 6. These plots complement similar results from [19] obtained by simulation.

It is also possible to analyse properties of the following type in PRISM: "the probability that the concentration of protein X drops below level K by time T" and "the probability that it takes longer than time T for the amount of protein Y to reach level L". Furthermore, PRISM can accurately compute properties of the system as it reaches equilibrium, e.g., "the long-run concentration of protein X".

8 Conclusion

In this paper, we have shown that probabilistic model checking and, in particular, the PRISM tool have been successfully

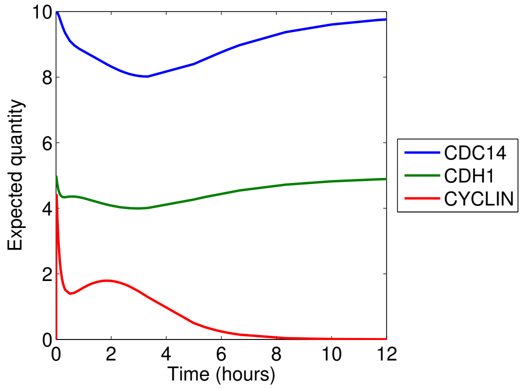

Figure 6: Reactant quantities over time for cell cycle control in eukaryotes.

applied to case studies from a very wide range of application domains. We have also illustrated the variety of properties which can be analysed using this approach and the fact that this allows identification of flaws and anomalies both in the systems being studied and in existing analyses of these systems. Furthermore, we have demonstrated one of the key strengths of the probabilistic formal verification approach: the ability to compute best- and worst-case performance and reliability measures, via exhaustive model analysis.

References

[1] PRISM web site. www.cs.bham.ac.uk/~dxp/prism.

[2] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[3] M. Duflot, L. Fribourg, T. Hérault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. In *Proc. 4th Workshop on Automated Verification of Critical Systems (AVoCS'04)*. Elsevier Science, Electronic Notes in Theoretical Computer Science Science, 2004. To appear.

[4] M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker. A formal analysis of Bluetooth device discovery. In *Proc. 1st International Symposium on Leveraging Applications of Formal Methods (ISOLA'04)*, 2004. To appear.

[5] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[6] W. Fokkink and J. Pang. Simplifying Itai-Rodeh leader election for anonymous rings. In *Proc. 4th Workshop on Automated Verification of Critical Systems (AVoCS'04)*. Elsevier Science, Electronic Notes in Theoretical Computer Science Science, 2004. To appear.

[7] J. Han and P. Jonker. A system architecture solution for unreliable nanoelectronic devices. *IEEE Transactions on Nanotechnology*, 1:201–208, 12002.

[8] T. Herman. Probabilistic self-stabilization. *Information Processing Letters*, 35(2):63–67, 1990.

[9] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov chain model checker. In S. Graf and M. Schwartzbach, editors, *Proc. 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'00)*, volume 1785 of *LNCS*, pages 347–362. Springer, 2000.

[10] B. Jeannet, P. D'Argenio, and K. Larsen. RAPTURE: A tool for verifying Markov decision processes. In I. Cerna, editor, *Proc. Tools Day, affiliated to 13th Int. Conf. Concurrency Theory (CONCUR'02)*, Technical Report FIMU-RS-2002-05, Faculty of Informatics, Masaryk University, pages 84–98, 2002.

[11] M. Kwiatkowska and G. Norman. Verifying randomized Byzantine agreement. In D. Peled and M. Vardi, editors, *Proc. Formal Techniques for Networked and Distributed Systems (FORTE'02)*, volume 2529 of *LNCS*, pages 194–209. Springer, 2002.

[12] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 2.0: A tool for probabilistic model checking. In *Proc. 1st International Conference on Quantitative Evaluation of Systems (QEST'04)*, pages 322–323. IEEE Computer Society Press, 2004.

[13] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. In K. Larsen and P. Niebert, editors, *Proc. Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, volume 2791 of *LNCS*, pages 105–120. Springer-Verlag, 2003.

[14] M. Kwiatkowska, G. Norman, and R. Segala. Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *LNCS*, pages 194–206. Springer, 2001.

[15] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In H. Hermanns and R. Segala, editors, *Proc. 2nd Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV'02)*, volume 2399 of *LNCS*, pages 169–187. Springer, 2002.

[16] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Special Issue of Formal Aspects of Computing*, 14:295–318, 2003.

[17] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. In Y. Lakhnech and S. Yovine, editors, *Joint Conference on Formal Modelling and Analysis of Timed Systems (FORMATS) and Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT)*, volume 3253 of *LNCS*, pages 293–308. Springer, 2004.

[18] R. Lanotte, A. Maggiolo-Schettini, and A. Troina. Automatic analysis of a non-repudiation protocol. In *Proc. 2nd International Workshop on Quantitative Aspects of Programming Languages (QAPL'04)*, 2004.

[19] P. Lecca and C. Priami. Cell cycle control in eukaryotes: A BioSpi model. In *Proc. Workshop on Concurrent Models in Molecular Biology (BioConcur'03)*, Electronic Notes in Theoretical Computer Science, 2003.

[20] A. McIver and C. Morgan. An elementary proof that Herman's ring is $\theta(n^2)$. Submitted for publication.

[21] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla. Evaluating the reliability of NAND multiplexing with PRISM. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005. To appear.

[22] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, and R. Gupta. Formal analysis and validation of continuous time Markov chain based system level power management strategies. In W. Rosenstiel, editor, *Proc. 7th Annual IEEE International Workshop on High Level Design Validation and Test (HLDVT'02)*, pages 45–50. IEEE Computer Society Press, 2002.

[23] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, and R. Gupta. Using probabilistic model checking for dynamic power management. In M. Leuschel, S. Gruner, and S. L. Presti, editors, *Proc. 3rd Workshop on Automated Verification of Critical Systems (AVoCS'03)*, Technical Report DSSE-TR-2003-2, University of Southampton, pages 202–215, April 2003.

[24] G. Norman and V. Shmatikov. Analysis of probabilistic contract signing. In A. Abdallah, P. Ryan, and S. Schneider, editors, *Proc. BCS-FACS Formal Aspects of Security (FASec'02)*, volume 2629 of *LNCS*, pages 81–96. Springer, 2003.

[25] G. Norman and V. Shmatikov. Analysis of probabilistic contract signing. Submitted, 2005.

[26] P. Pillai and K. Shin. Real-time dynamic voltage scaling for low-powered embedded operating systems. *Operating Systems Review*, 35(5):89–102, 2001.

[27] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.

[28] V. Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3/4):355–377, 2004.

[29] M. Stoelinga. *Alea jacta est: Verification of probabilistic, real-time and parametric systems*. PhD thesis, University of Nijmegen, 2002.

[30] J. von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956.