On Statistical Model Checking with PLASMA

Axel Legay, Sean Sedwards

On Statistical Model Checking with PLASMA

Axel Legay and Sean Sedwards
IRISA/INRIA
Rennes, France
{*axel.legay, sean.sedwards*}*@inria.fr*

***Abstract*—This paper surveys the main functionalities of the PLASMA statistical model checking platform developed at Inria.**

***Keywords*-Probability, Simulation, Statistical Model Checking, Experiments.**

I. CONTEXT

Quantitative properties of stochastic systems are usually specified in logics that allow one to compare the measure of executions satisfying certain temporal properties with thresholds. The model checking problem for stochastic systems with respect to such logics is typically solved by a numerical approach [1], [6] that iteratively computes (or approximates) the exact measure of paths satisfying relevant subformulas; the algorithms themselves depend on the class of systems being analysed as well as the logic used for specifying the properties.

Another approach to solve the model checking problem is to *simulate* the system for finitely many runs, and use *hypothesis testing* to infer whether the samples provide *statistical* evidence for the satisfaction or violation of the specification [35]. This approach is known as statistical model checking (SMC) and is based on the notion that since sample runs of a stochastic system are drawn according to the distribution defined by the system, they can be used to obtain estimates of the probability measure on executions. Starting from time-bounded PCTL properties [35], the technique has been extended to handle properties with unbounded until operators [29], as well as to black-box systems [28], [35]. Tools, based on this idea have been built [16], [30], [35], and have been used to analyse many systems that are intractable numerical approaches.

The SMC approach enjoys many advantages. First, the algorithms require only that the system be simulatable (or rather, sample executions be drawn according to the measure space defined by the system). Thus, it can be applied to larger class of systems than numerical model checking algorithms, including black-box systems and infinite state systems. In particular, SMC avoids the 'state explosion problem' [7]. Second the approach can be generalized to a larger class of properties, including Fourier transform based logics. Third, SMC requires many independent simulation runs, making it easy to parallelise and scale to industrial-sized systems.

While it offers solutions to some intractable numerical model checking problems, SMC also introduces some additional problems. First, SMC only provides probabilistic guarantees about the correctness of the results. Second, the required sample size grows quadratically with respect to the required confidence of the result. This makes rare properties difficult to verify. Third, only the simulation of purely probabilistic systems is well defined. Nondeterministic systems, which are common in the field of formal verification, are especially challenging for SMC.

In this invited talk, we will overview and compare some of the existing statistical model checking algorithms. We will then present PLASMA, a modular statistical model checker being developed at Inria, which addresses the problems of rare events and nondeterminism described above. Finally, some experiments will conclude the paper.

II. ON STATISTICAL MODEL CHECKING

Consider a stochastic system $\mathcal{S}$ and a logical property φ that can be checked on finite executions of the system. Statistical Model Checking (SMC) refers to a series of simulation-based techniques that can be used to answer two questions: (1) *Qualitative*: Is the probability for $\mathcal{S}$ to satisfy φ greater or equal to a certain threshold? and (2) *Quantitative*: What is the probability for $\mathcal{S}$ to satisfy φ? In contrast to numerical approaches, the answer is given up to some correctness precision.

In the sequel, we overview two SMC techniques. Let B_i be a discrete random variable with a Bernoulli distribution of parameter p. Such a variable can only take 2 values 0 and 1 with $Pr[B_i = 1] = p$ and $Pr[B_i = 0] = 1 - p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies φ and 0 otherwise.

Qualitative Answer: The main approaches [35], [28] proposed to answer the qualitative question are based on *sequential hypothesis testing* [31]. Let $p = Pr(\varphi)$. To determine whether $p \geq \theta$, we can test $H : p \geq \theta$ against $K : p < \theta$. A test-based solution does not guarantee a correct result but it is possible to bound the probability of error. The *strength* of a test is determined by two parameters, α and β, such that the probability of accepting

K (respectively, H) when H (respectively, K) holds, called a Type-I error (respectively, a Type-II error) is less or equal to α (respectively, β). A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [31], [35] for details). A solution is to use an *indifference region* $[p_1, p_0]$ (given some δ, $p_1 = \theta - \delta$ and $p_0 = \theta + \delta$) and to test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$. We now sketch the Sequential Probability Ratio Test (SPRT). In this algorithm, one has to choose two values A and B ($A > B$) that ensure that the strength of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr(B_i = b_i \mid p = p_1)}{Pr(B_i = b_i \mid p = p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}},$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. The algorithm computes $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied. This has the advantage of minimizing the number of simulations required to make the decision.

Quantitative Answer: In [15] Peyronnet et al. propose an estimation procedure to compute the probability p for $\mathcal{S}$ to satisfy φ. Given a *precision* δ, the *Chernoff bound* of [23] is used to compute a value for p' such that $|p' - p| \leq \delta$ with *confidence* $1 - \alpha$. Let $B_1 \dots B_m$ be m Bernoulli random variables with parameter p, associated to m simulations of the system considering φ. Let $p' = \sum_{i=1}^{m} b_i/m$, then the Chernoff bound [23] gives $Pr(|p' - p| \geq \delta) \leq 2e^{-2m\delta^2}$. As a consequence, if we take $m = \lceil \ln(2/\alpha)/(2\delta^2) \rceil$, then $Pr(|p' - p| \leq \delta) \geq 1 - \alpha$.

A. Rare Events

Statistical model checking avoids the exponential growth of states associated with probabilistic model checking by estimating probabilities from multiple executions of a system and by giving results within confidence bounds. Rare properties are often important but pose a particular challenge for simulation-based approaches, hence a key objective for SMC is to reduce the number and length of simulations necessary to produce a result with a given level of confidence. In the literature, one finds two techniques to cope with rare events: *importance sampling* and *importance splitting*.

In order to minimize the number of simulations, importance sampling works by estimating a probability using weighted simulations that favour the rare property, then compensating for the weights. For importance sampling to be efficient, it is thus crucial to find good importance sampling distributions without considering the entire state space. In [17], we presented a simple algorithm that uses the notion of cross-entropy minimisation to find an optimal importance sampling distribution. In contrast to previous work, our algorithm uses a naturally defined low dimensional vector of parameters to specify this distribution and thus avoids the intractable explicit representation of a transition matrix. We show that our parametrisation leads to a unique optimum and can produce many orders of magnitude improvement in simulation efficiency (See Section IV-D).

One of the open challenges with importance sampling is that the variance of the estimator cannot be usefully bounded with only the knowledge gained from simulation. Importance *splitting* achieves this objective by estimating a sequence of conditional probabilities, whose product is the required result. In [18] we motivated the use of importance splitting for statistical model checking and were the first to link this standard variance reduction technique [19] with temporal logical. In particular, we showed how to create *score functions* based on logical properties, and thus define a set of *levels* that delimit the conditional probabilities. In [18] we also described the necessary and desirable properties of score functions and levels, and gave two importance splitting algorithms: one that uses fixed levels and one that discovers optimal levels adaptively.

B. Nondeterminism

Markov decision processes (MDP) and other nondeterministic models interleave nondeterministic *actions* and probabilistic transitions, possibly with rewards or costs assigned to the actions [2], [26]. These models have proved useful in many real optimisation problems (see [32], [33], [34] for a survey of applications of MDPs) and are also used in a more abstract sense to represent concurrent probabilistic systems (e.g., [3]). Such systems comprise probabilistic subsystems whose transitions depend on the states of the other subsystems, while the order in which concurrently enabled transitions execute is nondeterministic. This order may radically affect the expected reward or the probability that a system will satisfy a given property. Numerical model checking may be used to calculate the upper and lower bounds of these quantities, but a simulation semantics is not immediate for nondeterministic systems and SMC is therefore challenging.

SMC cannot be applied to nondeterministic systems without first resolving the nondeterminism using a *scheduler* (alternatively a *strategy* or a *policy*). Since nondeterministic and probabilistic choices are interleaved, schedulers are typically of the same order of complexity as the system as a whole and may be infinite.

In [22] we presented the basis of the first lightweight SMC algorithms for MDPs and other nondeterministic models, using an $\mathcal{O}(1)$ representation of history-dependent schedulers. Our solution is based on pseudo-random number generators and an efficient hash function, allowing schedulers to be sampled using Monte Carlo techniques. Previous attempts to apply SMC to nondeterministic models [4], [21], [14], [12] have been memory-intensive (heavyweight) and incomplete

in various ways. The algorithms of [4], [12] consider only systems with 'spurious' nondeterminism that does not actually affect the probability of a property. In [21] the authors consider only memoryless schedulers and do not consider the standard notion of optimality used in model checking (i.e., with respect to probability). The algorithm of [14] addresses a standard qualitative probabilistic model checking problem, but is limited to memoryless schedulers that must fit into memory and does not in general converge to the optimal scheduler.

III. The Plasma Statistical Model Checker

Plasma [24] is an efficient self-contained SMC tool and software library [5] written in Java. Plasma features a customisable simulator class that allows SMC functionality to be added to existing domain-specific modelling platforms, such as DESYRE[1], and allows rapid prototyping of formal verification solutions using, e.g., Scilab[2], MATLAB and SIMULINK[3]. High performance standalone model checkers can also be constructed with Plasma by including a suitable language parser in the simulator class. Plasma's integrated development environment (IDE) provides the reactive modules modelling language as standard and facilitates distributed simulation. Importantly, Plasma can work with multiple user-defined language plug-ins, which can take advantage of all the IDE's features. In contrast, other dedicated SMC tools, such as YMER[4], VESPA, APMC[5] and COSMOS[6] each use a single modelling language. The tool MRMC[7] has both numerical and statistical functionality, but takes as input an explicit textual description of a Markov chain and maintains it in memory. This approach makes MRMC impractical as an SMC tool. Other numerical model checking tools that have added SMC functionality, such as PRISM[8] and UPPAAL[9], are nevertheless limited to their own modelling language, albeit that these languages may encompass different semantics (e.g., [11], [10]).

A. Properties

Plasma accepts properties described in a form of bounded linear temporal logic (BLTL) extended with custom temporal operators based on concepts such as *minimum*, *maximum* and *mean* of a variable over time.

B. Model Checking Modes

Plasma offers three basic modes of model checking: simple Monte Carlo, Monte Carlo using a Chernoff confidence bound and sequential hypothesis testing. There is also a simulation mode for debugging. Rare event model checking modes, such as importance sampling and importance splitting, can be implemented as part of the simulator class when the modelling semantics support them.

- Monte Carlo: the user explicitly specifies the number of simulations that Plasma must use to estimate the probability of a property.
- Chernoff: the user specifies an absolute error δ and a probability α. Plasma calculates the number of simulations required to ensure that the resulting estimate is within $\pm\delta$ of the correct value with minimum probability $1-\alpha$.
- Sequential: Plasma adopts the sequential hypothesis ratio test of [31] to verify that the probability of a property is above a user-specified threshold. The user also specifies a level of indifference and parameters to control errors of Types I and II. The number of simulations is not specified a priori: simulations are performed as necessary. See [31], [35] for details.

C. Graphical User Interface

The GUI provides an integrated development environment (IDE) to facilitate the use of Plasma as a standalone statistical model checker with multiple 'drop-in' modelling languages. To demonstrate this, we have included a biochemical language and a language based on reactive modules. The website [24] includes other examples. The GUI implements the notion of a project file, that links the description of a model to a specific modelling language simulator and a set of associated properties and experiments. The GUI also provides 2D and 3D graphical output of results and implements a distributed algorithm that will work with any of its associated modelling languages.

D. Usage

Plasma is usually invoked via its GUI. It may also be invoked from the command line or embedded in other software as a library. In addition to the GUI, Plasma provides an SMC engine in the form of a pre-compiled jar file. A source template is also provided to create custom simulator classes. The minimum requirement to create a custom simulator is to implement methods that (*i*) initiate a new simulation and (*ii*) advance the simulation by one step. Dedicated language parsers are typically invoked in the constructor of the custom simulator class.

E. Distributed SMC

The administrative time needed to distribute SMC on parallel computing architectures is often a deterrent. To overcome this, the Plasma GUI implements a simple and robust client-server architecture. The algorithm will work on dedicated clusters and grids, but can also take advantage of ad hoc networks of heterogeneous computers. The minimum requirement is that the IP address of the GUI is available to the clients. Plasma implements the SMC distribution algorithm of [35], which avoids the statistical bias that might otherwise occur from load balancing. The typical "divide by

[1] www.ales.eu.com [2] www.scilab.org [3] www.mathworks.com
[4] www.tempastic.org/ymer [5] sylvain.berbiqui.org/apmc
[6] www.lsv.ens-cachan.fr/~barbot/cosmos/ [7] www.mrmc-tool.org
[8] www.prismmodelchecker.org [9] www.uppaal.org

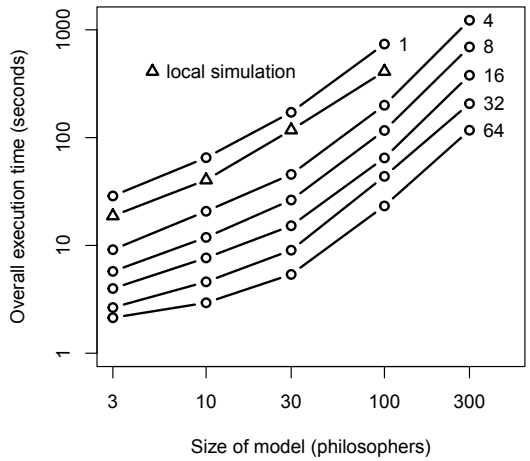

Figure 1. Scaling of PLASMA distributed algorithm applied to dining philosophers. Numbers are quantity of simulation nodes. Local simulation scaling is shown for reference.

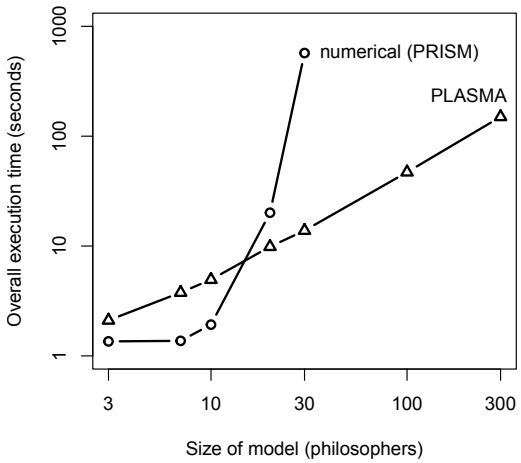

Figure 2. Exponential scaling of numerical model checking vs. linear scaling of PLASMA SMC, considering a fairness property of the probabilistic dining philosophers protocol.

number of clients" distributed performance is illustrated in Fig. 1.

The user selects the distributed mode via the GUI and publishes the IP address of the instance of PLASMA GUI that is acting as server. Clients (instances of the PLASMA service application) willing to participate respond by sending a message to the published IP address. The server sends an encapsulated version of the model and property to each of the participating clients, which then wait to be told how many simulations to perform. When sufficient clients are available, the user initiates the analysis by causing the server to broadcast the simulation requirements to each client.

IV. Overview of Results

PLASMA has state of the art performance, based on efficient simulation and optimised logic parsing. Figure 2 illustrates its performance in comparison that of a numerical approach. Checking a standard fairness property of increasing instance sizes of the probabilistic dining philosophers protocol [25], the figure illustrates how numerical model checking scales exponentially (with number of states) while PLASMA scales linearly (with length of property). While "toy" examples are often efficient with numerical model checking, "industrial scale" problems require SMC.

PLASMA has been applied to problems from, e.g., systems biology, rare events, performance, reliability, motion planning and systems of systems [24]. PLASMA is the focus of ongoing collaborations with companies Dassault, Thales, IBM, and EADS. PLASMA is also used by several European projects. In particular, sections IV-A and IV-B relate to the DALi[10] and DANSE[11] projects, respectively.

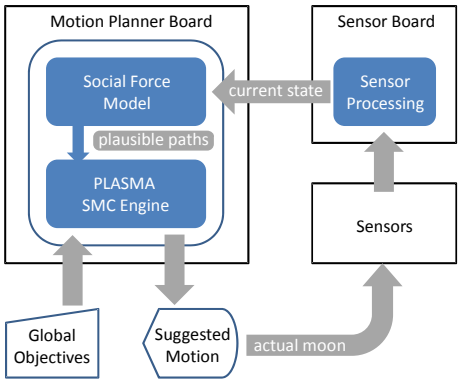

Figure 3. Control loop of DALi motion planner.

A. Motion Planning

PLASMA is used by the DALi project in a novel motion planning application of SMC. DALi aims to develop an autonomous device to help those with impaired ability to negotiate complex crowded environments (e.g. shopping malls). High level constraints and the objectives of the user are expressed in temporal logic, while low level behaviour is predicted by the 'social force model' [13]. The planner first hypothesises many plausible futures for a range of possible user actions, then chooses the action which maximises the probability of success.

PLASMA was integrated with MATLAB to develop the prototype algorithm. The final version is implemented directly in C on embedded hardware and finds the optimum trajectory in a fraction of a second [9]. PLASMA improves the social force model's ability to avoid collisions by a factor of five [9]. Using behavioural templates [8], the predictive power of our SMC-based motion planner can be even greater.

[10] www.ict-dali.eu [11] www.danse-ip.eu

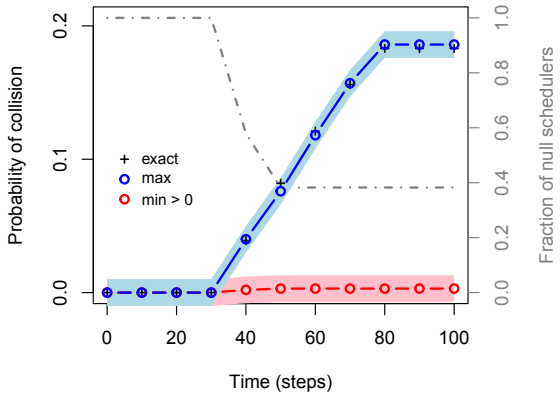

Figure 4. Max. and min. probabilities of second collision in WLAN protocol of [20].

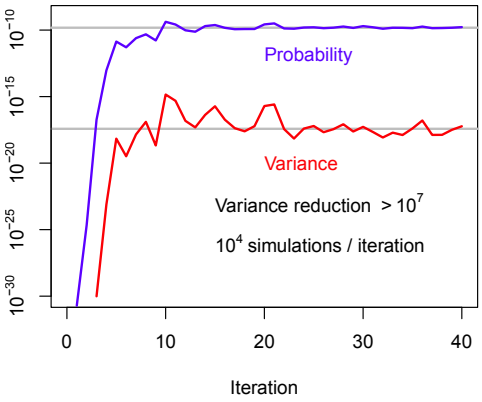

Figure 5. Convergence of parametrised C-E minimisation algorithm applied to chemical system.

B. Systems of Systems

The DANSE project is concerned with the design and analysis of 'systems of systems' (SoS). SoS feature a dynamicity of configurations that introduces significant additional complexity (the state space of the model is not necessarily definable a priori). PLASMA is now an integral part of the DANSE software platform, using a custom simulator class that wraps the DESYRE[12] hybrid simulation engine to make dynamicity transparent to SMC.

C. Markov Decision Processes

We have recently developed an efficient way to represent and sample general schedulers, which thus facilitates lightweight SMC algorithms for nondeterministic systems [22]. Figure 4 illustrates the results of sampling just 4000 schedulers per data point, to find the maximum probability of a second data "collision" on the shared communication channel of the WLAN protocol model of [20]. Maximum and minimum estimated probabilities are denoted by blue and red circles, with the correct values denoted by black crosses. The shaded areas indicate the $\pm\delta$ error of the estimates (using the Chernoff bound for multiple schedulers [22]) and reveal that our estimates are very close to the true values.

Our ongoing focus in this cutting-edge field is the development of lightweight learning algorithms and "smart sampling" to improve scalability.

D. Parametrised Cross-Entropy Minimisation

Rare events pose a challenge for SMC because it is necessary to consider error bounds ($\pm\delta$) that are relative to the estimate, such that the number of simulations required for a given confidence scales quadratically with rarity. Importance sampling is a standard technique to overcome this problem [19], but previous algorithms to find good importance sampling distributions do not scale, because they are parametrised at the level of individual transition probabilities [27]. Our cross-entropy (C-E) minimisation algorithm achieves impressive results, but uses a natural and convenient parametrisation at the level of the syntactical description of the system.

We illustrate our algorithm with a chemically reactive system. The stochastic model is based on the following set of chemical reactions, parametrised by rates λ_1, λ_2 and λ_3:

$$A + B \xrightarrow{\lambda_1} C \qquad C \xrightarrow{\lambda_2} D \qquad D \xrightarrow{\lambda_3} E$$

With initial numbers of molecules $A = B = 1000$ and $C = D = E = 0$, the system has approximately 1.6×10^8 states. Figure 5 illustrates the results of 40 iterations of our C-E algorithm, considering the property $Pr(\Diamond D \geq 470)$. For each iteration, the blue line gives the estimated probability, while the red line gives the variance of the estimator. For rare properties, variance $\approx$ probability, hence the horizontal grey lines (the mean of the last ten respective data points) indicate that the overall variance reduction is $\approx 10^7$, at a total cost of only 4×10^5 simulations.

V. CONCLUSION

This paper briefly summarises the main functionalities of PLASMA. Our ongoing work concerns the development of (*i*) algorithms to estimate unbounded properties with finite traces, (*ii*) lightweight learning algorithms for nondeterministic systems and (*iii*) techniques to provide confidence bounds for rare properties.

REFERENCES

[1] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE*, 29(6):524–541, 2003.

[2] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[12] www.ales.eu.com

[3] A. Bianco and L. De Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*, pages 499–513. Springer, 1995.

[4] J. Bogdoll, L. M. F. Fioriti, A. Hartmanns, and H. Hermanns. Partial order methods for statistical model checking and simulation. In *Formal Techniques for Distributed Systems*, pages 59–74. Springer, 2011.

[5] B. Boyer, K. Corre, A. Legay, and S. Sedwards. PLASMA-lab: A flexible, distributable statistical model checking library. In K. Joshi, M. Siegle, M. Stoelinga, and P. D'Argenio, editors, *Quantitative Evaluation of Systems*, volume 8054 of *Lecture Notes in Computer Science*, pages 160–164. Springer Berlin Heidelberg, 2013.

[6] F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, LNCS, 2925, pages 147–188. Springer, 2004.

[7] E. M. Clarke, E. A. Emerson, and J. Sifakis. Model checking: algorithmic verification and debugging. *Commun. ACM*, 52(11):74–84, Nov. 2009.

[8] A. Colombo, D. Fontanelli, D. Gandhi, A. De Angeli, L. Palopoli, S. Sedwards, and A. Legay. Behavioural templates improve robot motion planning with social force model in human environments. In *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, pages 1–6. IEEE, 2013.

[9] A. Colombo, D. Fontanelli, A. Legay, L. Palopoli, and S. Sedwards. Motion planning in crowds using statistical model checking to enhance the social force model. In *IEEE Conference on Decision and Control (CDC)*. IEEE, 2013.

[10] A. David, D. Du, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, and S. Sedwards. Statistical model checking for stochastic hybrid systems. In E. Bartocci and L. Bortolussi, editors, Proceedings First International Workshop on *Hybrid Systems and Biology,* Newcastle Upon Tyne, 3rd September 2012, volume 92 of *Electronic Proceedings in Theoretical Computer Science*, pages 122–136. Open Publishing Association, 2012.

[11] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, and S. Sedwards. Runtime verification of biological systems. In *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*, pages 388–404. Springer, 2012.

[12] A. Hartmanns and M. Timmer. On-the-fly confluence detection for statistical model checking. In *NASA Formal Methods*, pages 337–351. Springer, 2013.

[13] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, 1995.

[14] D. Henriques, J. G. Martins, P. Zuliani, A. Platzer, and E. M. Clarke. Statistical model checking for Markov decision processes. In *Quantitative Evaluation of Systems, 2012 Ninth International Conference on*, pages 84–93. IEEE, 2012.

[15] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *VMCAI*, volume 2937 of *LNCS*, pages 73–84. Springer, 2004.

[16] C. Jegourel, A. Legay, and S. Sedwards. A Platform for High Performance Statistical Model Checking – PLASMA. In C. Flanagan and B. König, editors, *TACAS*, volume 7214 of *LNCS*, pages 498–503. Springer, 2012.

[17] C. Jegourel, A. Legay, and S. Sedwards. Cross-Entropy Optimisation of Importance Sampling Parameters for Statistical Model Checking. In P. Madhusudan and S. A. Seshia, editors, *CAV*, volume 7358 of *LNCS*, pages 327–342. Springer, 2012.

[18] C. Jegourel, A. Legay, and S. Sedwards. Importance splitting for statistical model checking rare properties. In *Computer Aided Verification*, pages 576–591. Springer, 2013.

[19] H. Kahn and A. W. Marshall. Methods of Reducing Sample Size in Monte Carlo Computations. *Operations Research*, 1(5):263–278, November 1953.

[20] M. Z. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol. In *Proceedings of the Second Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, pages 169–187. Springer-Verlag, 2002.

[21] R. Lassaigne and S. Peyronnet. Approximate planning and verification for large Markov decision processes. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1314–1319. ACM, 2012.

[22] A. Legay and S. Sedwards. Lightweight Monte Carlo verification of Markov decision processes. *Submitted*, 2014.

[23] M. Okamoto. Some inequalities relating to the partial sum of binomial probabilities. *Annals of the Institute of Statistical Mathematics*, 10:29–35, 1959.

[24] PLASMA project page. https://project.inria.fr/plasma-lab/.

[25] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1:53–72, 1986.

[26] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[27] A. Ridder. Importance sampling simulations of markovian reliability systems using cross-entropy. *Annals of Operations Research*, 134:119–136, 2005.

[28] K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114, pages 202–215. Springer, 2004.

[29] K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *CAV*, LNCS 3576, pages 266–280, 2005.

[30] K. Sen, M. Viswanathan, and G. A. Agha. Vesta: A statistical model-checker and analyzer for probabilistic systems. In *QEST*, pages 251–252. IEEE Computer Society, 2005.

[31] A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[32] D. J. White. Real applications of Markov decision processes. *Interfaces*, 15(6):73–83, 1985.

[33] D. J. White. Further real applications of Markov decision processes. *Interfaces*, 18(5):55–61, 1988.

[34] D. J. White. A survey of applications of Markov decision processes. *Journal of the Operational Research Society*, 44(11):1073–1096, Nov. 1993.

[35] H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.