PLASMA-lab: A Flexible, Distributable Statistical Model Checking Library

Benot Boyer, Kevin Corre, Axel Legay, Sean Sedwards

PLASMA-lab: a flexible, distributable statistical model checking library

Benoît Boyer, Kevin Corre, Axel Legay and Sean Sedwards

INRIA Rennes – Bretagne Atlantique

Abstract We present PLASMA-lab, a statistical model checking (SMC) library that provides the functionality to create custom statistical model checkers based on arbitrary discrete event modelling languages. PLASMA-lab is written in Java for maximum cross-platform compatibility and has already been incorporated in various performance-critical software and embedded hardware platforms. Users need only implement a few simple methods in a simulator class to take advantage of our efficient SMC algorithms.
PLASMA-lab may be instantiated from the command line or from within other software. We have constructed a graphical user interface (GUI) that exposes the functionality of PLASMA-lab and facilitates its use as a standalone application with multiple 'drop-in' modelling languages. The GUI adds the notion of projects and experiments, and implements a simple, practical means of distributing simulations using remote clients.

Background and motivation

Statistical model checking (SMC) is a form of probabilistic model checking that employs Monte Carlo methods to avoid the state explosion problem. SMC uses a number of independent simulation traces of a discrete event model to estimate the probability of a property. The traces may be generated on different machines, so SMC can efficiently exploit parallel computation (see Fig. 2). Reachable states are generated on-the-fly and the length of simulations is only weakly related to the size of the state space. Hence SMC tends to scale polynomially with respect to system description (see Fig. 1). Properties may be specified in bounded versions of the same temporal logics used in probabilistic model checking. Since SMC is thus applied to finite traces, it is also possible to use logics and functions that would otherwise be intractable or undecidable.

SMC abstracts the probabilistic model checking problem to one of estimating the parameter of a Bernoulli random variable with well defined confidence (e.g., using a Chernoff bound). The complexity of the estimation problem with respect to confidence is largely independent of the total number of possible traces. Hence SMC may also be applied to stochastic models with continuous states.

Dedicated SMC tools, such as YMER[1], VESPA, APMC[2] and COSMOS[3], have been joined by statistical extensions of established tools such as PRISM[4] and UPPAAL[5]. In the case of UPPAAL-SMC, this has required the definition of

[1] www.tempastic.org/ymer
[2] sylvain.berbiqui.org/apmc
[3] www.lsv.ens-cachan.fr/~barbot/cosmos/
[4] www.prismmodelchecker.org
[5] www.uppaal.org

stochastic timed semantics. The tool MRMC[6] has both numerical and statistical functionality, but takes as input a low level textual description of a Markov chain. Many other tools are available or under development, with most using a single high level modelling language related to a specific semantics. Our previous tool [2] suffered the same limitation, prompting us to develop a radically new tool with modular architecture.

PLASMA-lab

PLASMA-lab [3] is an efficient SMC library written in Java, featuring a customisable simulator class. This allows SMC functionality to be added to existing domain-specific modelling platforms, such as DESYRE[7], and allows rapid prototyping of formal verification solutions using, e.g., Scilab[8] and MATLAB[9]. High performance standalone model checkers can also be constructed with PLASMA-lab by including a suitable language parser in the simulator class. PLASMA-lab's integrated development environment facilitates distributed simulation and can work with multiple user-defined language plug-ins.

Properties PLASMA-lab 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.

Model checking modes PLASMA-lab 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-lab must use to estimate the probability of a property.
- Chernoff: the user specifies an absolute error ε and a probability δ. PLASMA-lab calculates the number of simulations required to ensure that the resulting estimate is within $\pm\varepsilon$ of the correct value with minimum probability δ.
- Sequential: PLASMA-lab adopts the sequential hypothesis ratio test of [4] 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 [4] for details.

Usage PLASMA-lab may be invoked from the command line or embedded in other software as a library. PLASMA-lab is provided as a pre-compiled jar file (plasmalab.jar) and a source template (Simulator.java) to create the simulator class. The minimum requirement is to implement the methods newTrace() and nextState(), that initiate a new simulation and advance the simulation by one step, respectively. Language parsers are typically invoked in the constructor.

[6] www.mrmc-tool.org [7] www.ales.eu.com [8] www.scilab.org
[9] www.mathworks.com

Graphical user interface The GUI provides an integrated development environment (IDE) to facilitate the use of PLASMA-lab 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 [3] 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.

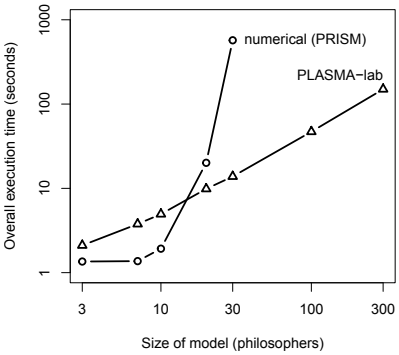

Figure 1. Exponential scaling of numerical model checking vs. linear scaling of PLASMA-lab SMC, considering a fairness property of the probabilistic dining philosophers protocol.

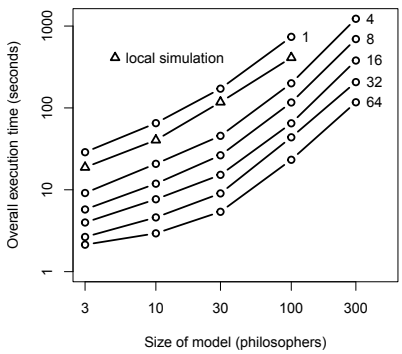

Figure 2. Scaling of PLASMA-lab distributed algorithm applied to dining philosophers. Numbers are quantity of simulation nodes. Local simulation scaling is shown for reference.

Distributed algorithm The administrative time needed to distribute SMC on parallel computing architectures is often a deterrent. To overcome this, the PLASMA-lab GUI implements a simple and robust client-server architecture, based on Java Remote Method Invocation (RMI) using IPv4/6 protocols. 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-lab implements the SMC distribution algorithm of [4], which avoids the statistical bias that might otherwise occur from load balancing. Distributed performance in illustrated in Fig. 2. The user selects the distributed mode via the GUI and publishes the IP address of the instance of PLASMA-lab GUI that is acting as server. Clients (instances of the PLASMA-lab 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.

Applications

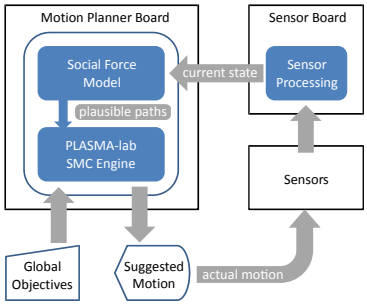

Figure 3. Control loop of DALi motion planner.

PLASMA-lab has been applied to problems from, e.g., systems biology, rare events, performance, reliability, motion planning and systems of systems [3]. PLASMA-lab is the focus of ongoing collaborations with companies Dassault, Thales, IBM, and EADS. PLASMA-lab is also used by several European projects. The following examples relate to the DALi[10] and DANSE[11] projects.

Motion planning PLASMA-lab 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' [1].

PLASMA-lab 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. PLASMA-lab improves the social force model's ability to avoid collisions by a factor of five.

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 and state space of the model are not necessarily known a priori). PLASMA-lab is now an integral part of the DANSE software platform, using a Simulator class that wraps the DESYRE[12] hybrid simulation engine to make dynamicity transparent to SMC.

References

1. D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, 1995.
2. 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.
3. PLASMA-lab project page. https://project.inria.fr/plasma-lab/.
4. H. L. S. Younes. Verification and planning for stochastic processes with asynchronous events. PhD thesis, Carnegie Mellon University, 2005.

[10] www.ict-dali.eu [11] www.danse-ip.eu [12] www.ales.eu.com