Project no.:	**318490 (FP7-ICT-2011-8)**
Project full title:	**Self Energy-Supporting Autonomous Computation**
Project Acronym:	**SENSATION**
Deliverable no.:	**D2.1**
Title of Deliverable:	**Deliverable D2.1**

Contractual Date of Delivery to the CEC:	**31 OCT 2013**
Actual Date of Delivery to the CEC:	**to be defined**
Organisation name of lead contractor for this deliverable:	**Inria**
Author(s):	**Axel Legay**
Participants(s):	**Inria**
Work package contributing to the deliverable:	**WP 5**
Nature:	**O (other)**
Version:	**1.0**
Total number of pages:	**18**
Start date of project:	1. Oct. 2012 **Duration:** 36 month

Project co-funded by the European Commission within the Seventh Framework Programme (2012-2015)

Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Keyword list: Statistical Model Checking, new algorithms, tools, experiments.

1 Introduction

Prime focus of this deliverable is to report advance made on statistical model checking (SMC) techniques. New SMC techniques aimed at (quickly) estimating expected energy consumption (within a desired level of confidence) and identifying dependencies between behavioural aspects and energy consumption. Related to the needs of addressing stochastic effects at the hardware level we investigate rare-event simulation techniques to improve the estimation of low probabilities of erroneous computational behaviour. While rare-event simulation techniques have been applied to detect simple bugs characterized by sets of states, bounded temporal properties of complex systems represented by stochastic timed automata are significantly more complex and have not been considered. The scientific challenge —developing a "probability estimator"— is tackled by combining classical model checking for timed automata with existing rare-event simulation approaches. The long term objectives are the following:

- A property-driven technique to reduce the number of simulation runs. Typically, a designer has some prior knowledge regarding the probability for the system to violate a property. We will estimate these probabilities and use it in a Bayesian fashion to reduce the number of necessary simulations in sequential hypothesis testing. Similar techniques have been applied in systems biology, but have been restricted to uniform distributions for refutation probabilities; in our setting we will consider more general distributions.
- A distributed version of SMC and new techniques to parallelise long simulations. Similar to regenerative simulation techniques, the idea is to identify a break point state in the system from which individual finite simulations can be run. The results of all the individual simulations need to be combined, which is extremely complex for continuous timed systems such as energy automata.
- Finally, we will exploit SMC for modelling formalisms of WP1 such as MoDeST and POOSL that are both equipped with tailored discrete-event simulation techniques. As timed automata are a subset of the models that can be expressed in these formalisms, we will study and compare SMC with these simulative techniques and adopt variance reduction techniques for both transient and steady-state metrics so as to improve industrial applicability.

1.1 Summary of the contributions

Aachen During the first year of the project, Aachen has developed a new application for the project. This application is a microgrid that presents several challenges related to energy consumption. The new application has been used to challenge the UPPAAL-SMC toolset developed by the Aalborg partner.

Aalborg During the first year of the project, Aalborg (in collaboration with Inria for parts of the work) has improved their UPPAAL toolset by adding support for floating-point types, support

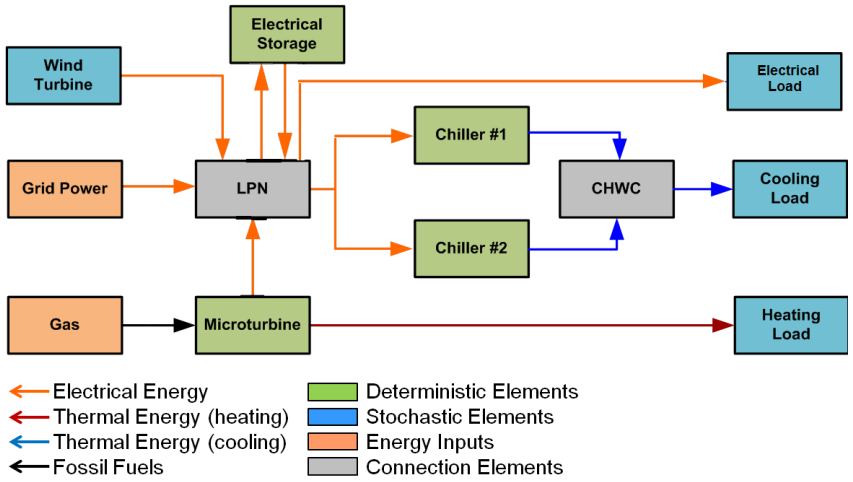

Figure 1: Configuration of the microgrid case study for statistical model checking

for MITL with dynamic creation of processes. A concrete simulator in the graphical interface has also been added. Finally, progresses on case-studies, in particular on battery scheduling to optimize energy utilization.

Inria During the first year of the project, Inria has developed new efficient algorithms for rare events. Another contribution is the developement of Plasma, an open-source toolset for SMC.

Saarbrucken in collaboration with University of Twente Standard statistical model checking (SMC) is only applicable to stochastic processes. We developed an on-the-fly confluence check to allow the application of SMC to models with spurious nondeterministic choices. This approach improves on previous techniques based on partial-order reduction and preserves all the guarantees of standard SMC.

2 Contributions

2.1 Aachen

During the first year of the project, Aachen contributed on defining challenging case studies for the project. Another contribution is on applying SMC toolsets of the Aalborg partners on those examples. In this deliverable, we reports on the modelling and analysis of a microgrid with wind, microturbines, and the main grid as generation resources. The microgrid is modelled as a parallel composition of various stochastic hybrid automata. Extensive simulation runs of the behaviour of the main individual microgrid components give insight into the complex dynamics of the system and provide useful information to determine adequate parameter settings. The analysis of the microgrid focuses on checking several temporal properties, expressed in the logic PCTL, using the statistical model checker UPPAAL-SMC.

Introduction to the example. We consider the microgrid configuration in Fig. 1. This microgrid is connected to the electricity and natural gas distribution grids and incorporates two local energy sources – wind turbine and a microturbine. The microturbine represents a deterministic energy source which produces electricity and heat. The heat can be utilized for satisfying the heating load such as domestic hot water. The wind turbine is of stochastic nature and its power output depends on actual wind properties. The local power network (LPN) stands for the junction point where the electricity supply meets the electricity demand e.g. chillers, electrical load, etc. Chillers remove the heat from the chilled water circuit (CHWC) which supplies the cooling load with the chill. Demand of cooling load as well as performances of chillers can be affected by the outside temperature. Electrical energy storage is connected to the LPN and enables the possibility to store/load electricity energy. This can be very useful for the optimization of operational costs of the microgrid.

We model this system as a composition of stochastic hybrid automata, and describe their continuous dynamics. For several components we have adopted the dynamics from the literature [19, 29, 32]. Extensive simulation runs of the behaviour of the microgrid components give insight into the complex dynamics of the system and provide useful information to determine adequate parameter settings. The parameter settings used in our verification models have been obtained in this way and have been validated by Honeywell. For the analysis of the microgrid, we resort to *model checking*. As the stochastic dynamics and complexity of the microgrid configuration go beyond the scope of the above techniques that typically rely on discretization and dynamic programming, we resort here to *statistical* model checking. This technique is basically employing Monte Carlo simulation in order to check the validity of temporal logic properties within a given a priori-defined confidence interval. In order to analyse probabilistic reachability and invariance properties of the microgrid case study, we use UPPAAL-SMC, a recent extension to the model checker Uppaal (`www.uppaal.org`). UPPAAL-SMC originally focused on (priced and probabilistic) timed automata. However, in Sensation our Aalborg partners improved the tool so that it also covers networks of stochastic hybrid automata [12]. We report on the statistical model checking of several (bounded) PCTL formulas and investigate the run-times for various confidence intervals and time horizons.

As described above, we have modelled all individual components as stochastic hybrid automata. Let *LPN*, CH_1, CH_2, *CHWC*, Z_i, MT, WT, ST, EL be the SHA models of the local power network, the two chillers, the chilled-water circuit, the i^{th} room (or zone), microturbine, wind turbine, storage, and electrical load, respectively. The composite model of the microgrid is now given as:

$$MG = (\, LPN \parallel CH_1 \parallel CH_2 \parallel CHWC \parallel \underbrace{Z \parallel \ldots \parallel Z}_{n \text{ zones}} \parallel MT \parallel WT \parallel ST \parallel EL)$$

where $\parallel$ denotes a parallel composition operator.

Simulation Results. There are various input parameters for the different components of the microgrid. Controllers are designed to modify these inputs in order to satisfy some optimality criteria, usually to minimize power consumption. Various controllers can be modelled by automata. The effect of these controllers can be studied by running the respective automaton in

parallel with the microgrid. For example, we can consider an automaton that depending on the required cooling power and ambient temperature, distributes the energy among the two chillers accordingly. It is possible that some chillers work better at lower energy requirements than others. The control parameter $\alpha_{ch}(t)$ can redistribute the cooling energy production between chillers. In the following experiment (see Fig. 2), we can see that the total power consumed by the chiller is higher when a fixed α is used in comparison to a more dynamic α which depends on the cooling energy requirements.

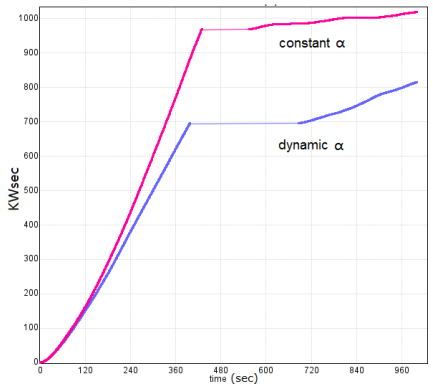

Figure 2: Cumulative electrical power consumed by the chillers

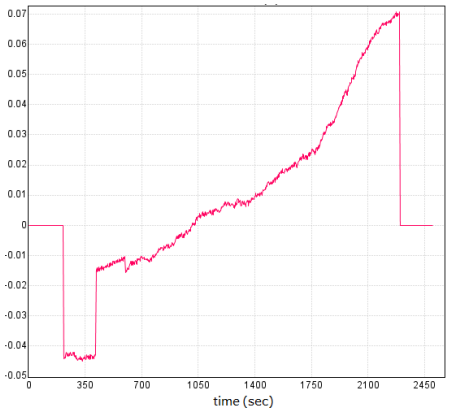

Figure 3: Frequency deviation

We are also interested in testing the behaviour of the microgrid under special conditions, e.g., observing the frequency deviation over time when the microgrid is disconnected from the main grid (islander mode). Figure 3 shows the frequency deviation when the microgrid is running in islander mode with six wind turbines providing for two chillers and one electric load along with a storage device. Initially, the power generated by wind turbines is insufficient –in this experiment, it took some time for the wind turbine to pick up speed, so to speak– and hence the batteries were catering for the power deficit. At some point, the batteries run out and we have negative frequency deviation. When the six turbines accelerate, they provide (more than) enough energy for the demand yielding the positive deviation. Finally, the batteries start to recharge themselves to use the excess power and the frequency stabilizes.

The UPPAAL-SMC **Tool.** Numerical methods to solve model-checking problems of stochastic hybrid systems against some temporal formula are algorithmically involved and suffer from the curse of dimensionality [1, 2, 31]. The main reason for this is the use of discretization. In contrast, statistical model checking avoids these problems by resorting to discrete-event simulation. In a nutshell, it generates and examines finitely many simulation runs of the model at hand, and uses hypothesis testing to infer whether the obtained simulation samples provide statistical evidence for the satisfaction or violation of some (temporal logic) specification [33]. We analyze the microgrid case study using the UPPAAL-SMC tool. Initially, UPPAAL-SMC focused on networks of priced timed automata. These are timed systems in which real variables may have different rates (even potentially negative) in different modes (these variable are not used as guards). A recent extension allows to handle stochastic hybrid automata; this was first reported

in [12]. Here, the change of continuous variables is governed by linear differential equations. UPPAAL-SMC relies on results from statistics such as sequential hypothesis testing and Monte Carlo simulation. Crucial statistical input parameters are the confidence ξ which quantifies the error, given by the parameter α and β (as before), and the probability interval defined by the indifference region δ. Apart from being able to simulate various variables, we can carry out statistical model checking of temporal logic formulas, in particular, PCTL (probabilistic CTL). This logic allows for specifying (a bound on) the likelihood of reaching a specific set T of target states within n steps as denoted by the formula $\mathbb{P}(\Diamond_{\leq n}\, T)$, or the probability to stay in a given set T of states for the next time period, denoted by $\mathbb{P}(\Box_{\leq n}\, T)$.

PCTL property	T_{CWSP}	T_{ZASP}	$\langle\alpha,\beta\rangle$	δ	Pr	t	runs (n)	time (s)
$\mathbb{P}\left(\Diamond_{\leq t}\, \lvert T_{ZA} - T_{ZASP}\rvert < 1\right)$	20	25	$\langle 0.1, 0.1\rangle$	0.1	[0.9,1]	200	150	1.713
	20	25	$\langle 0.05, 0.05\rangle$	0.1	[0.9,1]	200	185	1.825
	20	25	$\langle 0.1, 0.1\rangle$	0.05	[0.95,1]	200	600	4.223
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	200	738	5.628
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	1000	738	5.565
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	2000	738	5.566
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	3000	738	5.668
	20	22	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	3000	738	46.605
	20	20	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	3000	738	82.246
$\mathbb{P}\left(\Diamond_{\leq t}\, \lvert T_{CW} - T_{CWSP}\rvert < 1\right)$	20	25	$\langle 0.1, 0.0.1\rangle$	0.1	[0,0.1]	200	150	16.81
	20	25	$\langle 0.05, 0.05\rangle$	0.1	[0,0.1]	200	185	20.505
	20	25	$\langle 0.1, 0.0.1\rangle$	0.05	[0,0.05]	200	600	64.942
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0,0.05]	200	738	79.154
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	1000	738	80.17
	15	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	1000	738	87.53
	10	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	1000	738	90.121
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	1000	738	81.551
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	2000	738	81.441
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	3000	738	81.401
$\mathbb{P}\left(\Diamond_{\leq t}\, Q_{cool} \geq 4500\right)$	20	25	$\langle 0.1, 0.1\rangle$	0.1	[0,0.1]	2000	150	162.837
	20	25	$\langle 0.05, 0.05\rangle$	0.1	[0,0.1]	2000	185	204.517
	20	25	$\langle 0.1, 0.1\rangle$	0.05	[0,0.05]	2000	600	661.487
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0,0.05]	2000	738	808.748
	20	25	$\langle 0.05, 0.05\rangle$	0.05	[0,0.05]	1000	738	399.706
	15	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	1000	738	20.355
	10	25	$\langle 0.05, 0.05\rangle$	0.05	[0.95,1]	1000	738	10.25

Table 1: SMC results for temperature control

Statistical Model Checking Results. For the microgrid case study we are mainly interested in reachability properties.

The first property of interest is the probability of the room temperature to be close to the desired set points within some finite time interval. We vary the value of the time interval in order to investigate the impact of this value on the run time and vary some of the temperature values (set points and desired ones). The model-checking results are listed in Table 1. The first column indicates the checked property, the second and third column indicate the values of the control inputs (temperate set points). The fourth and fifth column indicate the parameters α, β and δ of the SMC algorithm, the sixth column presents the probability interval that results from the SMC, the seventh column indicates the length t of the interval, the eighth column indicates

the number of required simulation runs, and the last column presents the run time of UPPAAL-SMC. We present the results of checking three properties. The first property refers to the time until the zone temperature T_{ZA} is at most one degree Celsius different from the desired room temperature T_{ZASP}. The second property aims at determining the probability that the chilled water temperature T_{CW} is close to its set point T_{CWSP}. As one can see from the results, this probability is low for small t (say, $t \leq 200$), whereas from t=$1,000$, T_{CW} has reached its set point. The last property focuses on the thermal capacity of the water circuit Q_{cool}, as realized by a PI controller, reaches a threshold 4,500. We experiment with various combination of control parameters, namely, the room temperature set point of the thermostat (T_{ZASP}) and chilled water (coolant) temperature set point (T_{CWSP}).

Observe that relaxing the probability interval has a larger effect on the run time than changing the confidence. As we expected lowering of set point temperature makes it longer for the system to attain equilibrium (the simulation figures) and hence the model checker need to run the simulation for longer duration. This causes an increase in verification time for highly probable properties. On the other hand, the energy requirement variable Q_{cool}, which almost never reached $4500J$ now crosses this threshold very quickly because of the high energy demand forced by lowering of set point temperature. Hence, simulations are short and verifications are quite quick.

As a next property, we are interested in the behaviour of the chillers. The PCTL-property of interest is the likelihood that the chiller power demand (of the first chiller, say) exceeds a certain threshold (2, say) within the next 3,000 seconds. The *cooling load (CL)* is determined by the components outside temperature, zone temperature, thermostat, chilled water temperature, chilled water controller, and the chillers. The results for the first chiller are listed in the upper part of Table 2. The lower part of this table presents the results of checking the maximal generated power by the wind turbine to exceed 0.25 kW. Observe that even if the indifference interval is specified to be 0.05, the probability interval given by UPPAAL-SMC actually is larger. We cannot explain this anomaly.

PCTL property	$\langle \alpha, \beta \rangle$	δ	Pr	# runs	time (in s)
$\mathbb{P}\left(\Diamond_{\leq 3000}\, P_{Ch,1} \geq 2\right)$	$\langle 0.05, 0.05 \rangle$	0.05	[0,0.092]	738	2484.379
$\mathbb{P}\left(\Diamond_{\leq 3000}\, P_{W,max} \leq 0.25\right)$	$\langle 0.05, 0.05 \rangle$	0.05	[0.3009,0.4009]	738	307.055

Table 2: SMC results for cooling load (upper) and maximal wind production (lower)

As a next property, we check whether the microgrid in islander mode keeps itself stable or not. We consider the following components: one cooling load (CL), one electrical load (EL), wind (W), five wind turbines (WT) and the local power network (LPN). The LPN crosses dangerous levels, when there is no storage device. Some verification results are listed in Table 3 (upper part). Finally, we consider a more complex scenario where it is assumed that the microturbine (MT) was in the *warm-up* mode, there are two electrical loads, one cooling load with two chillers and two wind turbines. The microgrid goes to the islander mode so as to compensate for the discrepancy the microtubine was speed up to 70000 rpm[1]. We check whether the LPN remains

[1]Such decisions are taken by network administrators.

stable or not, cf. Table 3 (lower part).

PCTL property	$\langle \alpha, \beta \rangle$	δ	Pr	# runs	time (in s)
$\mathbb{P}\left(\Diamond_{\leq 3000} \left\|fr\right\| \geq 0.5\right)$	$\langle 0.05, 0.05 \rangle$	0.05	[0.832,0.932]	738	523.193
$\mathbb{P}\left(\Diamond_{\leq 1000} \left\|fr\right\| \geq 0.5\right)$	$\langle 0.05, 0.05 \rangle$	0.05	[0,0.05]	738	1152

Table 3: SMC results for microgrid operating in islander mode

2.2 Aalborg

Statistical model-checking (SMC) is a relatively recent technique that complements traditional model-checking. The technique consists in generating finite runs and observing a given formula for every run. Then different statistical methods are used to do the actual model-checking. The checks are in two main categories: evaluation or hypothesis testing. For evaluation, a probability is computed with a confidence interval (given by the user). The number or runs can be bounded statically with the so-called Chernoff-Hoeffding bound or on-the-fly with Clopper-Pearson's method. For hypothesis testing the idea is to test if some probability is greater than a given threshold: $Pr(\phi) \geq \theta$. The algorithm tests two hypothesis: $H_0 : Pr(\phi) > \theta + \delta$ and $H_1 : Pr(\phi) < \theta - \delta$. The parameter δ defines the *region of indifference*. The method used is called sequential testing and the number of runs is determined on-the-fly.

UPPAAL-SMC [17, 11, 13] implements hypothesis testing, probability evaluation, probability comparisons (reduced to hypothesis testing), and expected value of the min or max of some expression on a run. In addition, the tool provides simulation capabilities useful for visualizing expressions or error traces. The tool supports filtering of simulations to only see runs that have certain properties. The tool also provides a plot composer to display different plots from the data gathered during the statistical check, such as, probability distributions or cumulative probabilities over time. The results are given with a confidence interval as well. The models accepted by UPPAAL-SMC are stochastic hybrid systems where ordinary differential equations (ODEs) and floating-point types (double) are supported. Moreover the SMC algorithm can be distributed with relative ease [9, 10].

UPPAAL-SMC has been used for different case-studies ranging from communication protocols such as Bluetooth, Firewire, schedulability [15], and energy aware buildings [14] to biological models [16]. The main use is performance analysis. While standard model-checkers are qualitative and can check if a given property holds or not, here we can analyze quantitative values, e.g., energy consumption of a sensor or probability of collision in a wireless network. Also, since we have a good match between our stochastic semantics and some kinetic laws in biochemistry, we can simulate biological models and build hybrid models that mix biological behaviour and discrete controllers. Another use of SMC is to complement the weaknesses of traditional model-checking. Sometimes, over-approximation techniques must be used for model-checking and the results may be inconclusive if some error state is declared to be reachable – in fact the

model-checker is not sure of it. SMC can be used in these cases to find concrete traces and confirm these errors.

3 Monitoring in UPPAAL-SMC

UPPAAL-SMC supports a very small fragment of LTL, namely $\Box\phi$ and $\Diamond\phi$. We extended the statistical model checking engine to support the MITL logic with dynamically generated processes. Contrary to the model-checker SPIN that uses monitors to check LTL properties, we do not create monitors in advance. The monitors we could construct for MITL are non-deterministic and cannot, in general, be determinized. Monitors used for statistical model checking are however required to be deterministic [8]. We circumvent this problem by not creating the monitor explicitly. Instead, we employ a rewriting technique of formulas that ensures all possible non-deterministic choices of the aforementioned monitor are explored [7, 18]. During the generation of a run, UPPAAL-SMC rewrites the formula every time the monitored system changes its internal state resulting in a new observation. An observation of the system in this setting is a truth assignment to all of the propositions in the MITL formula. The new formula is satisfied at the next observation if and only if the original formula is satisfied at the current observation. Eventually the formula is rewritten into $\top$ or $\bot$, at which point a verdict can be made: The original formula is satisfied if it becomes $\top$ and not satisfied if it becomes $\bot$. The new monitoring technique is guaranteed to terminate provided that the generated run is time diverging.

We have applied our technique to the verification of the IEEE 802.15.4 CSMA/CA protocol. In this work we verified that whenever a collision occurred for Node_i in the modelled network then that node node would re-transmit. The MITL property we used for verifying this was

$$\bigwedge_{i=1}^{n}(\Box_{\leq 10000}(\text{collision}_i \Rightarrow \Diamond_{\leq 4000}\text{send}_i)$$

where n is the number of nodes in the network.

4 Experiments

Wireless systems such as satellites are typically battery-powered. These systems, and therefore their batteries, are often exposed to loads with uncertain timings. We model the Kinetic Battery Model by Manwell and McGowan (1993) in combination with stochastic workloads. Using UPPAAL-SMC we can study system performance under various soft real-time scheduling principles.

Batteries contain electrochemical cells where reactions transform chemical energy to electrical energy. The speed of the diffusion of reactants in an electrochemical cell gives rise to a low-pass filtering of the changes in load that can be described by a system of differential equations in the Kinetic Battery Model. Battery modelling can thus be used to predict the state of charge during the run of a battery-powered system.

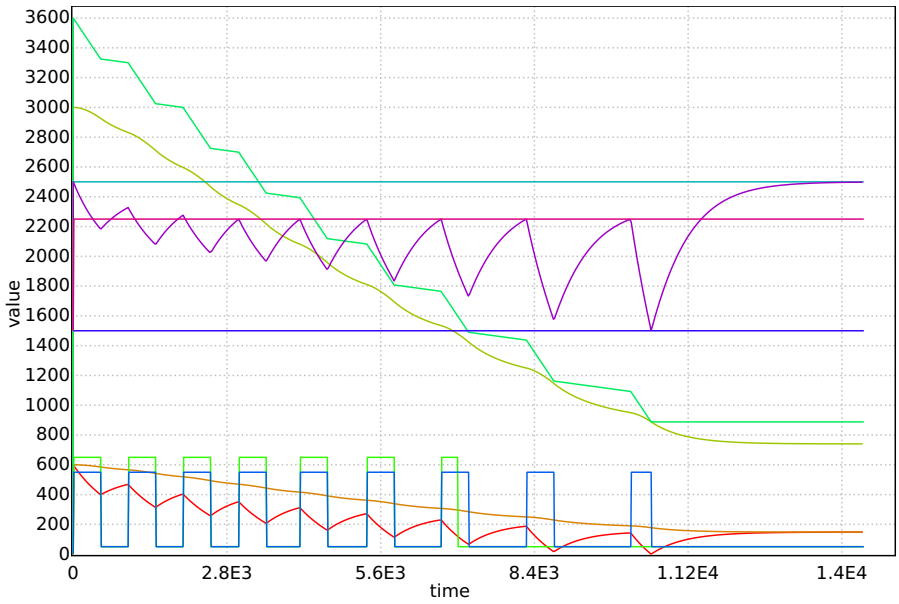

Figure 4: Balance scheduler simulation

We use a simple test load consisting of a constant idle load of 50 mA and a periodic task of 500 mA, a 1000 s period and deadline, a 400 s best-case execution time (BCET), and a 500 s worst-case execution time (WCET). Such a high-level task (on a scale of minutes instead of, say, milliseconds) may for example represent a satellite transmitting data to a ground station while passing over it. If the battery state of charge is low, there may be different strategies for still having a reasonable data throughput while trying to postpone the event of the battery becoming empty. For example, by considering the balance, i.e., the ratio between the available and bound charge in the battery according to the battery model, a scheduler may choose to delay the execution of the task until more charge is available. Such a system run is show in Figure 4. The horizontal "band" of lines between values 1500 and 2500 show the balance of the battery fluctuating from 100% to 0% where the battery has no more available charge. Before each task instance is allowed to execute, the balance must reach the threshold of 75%. The task instances are spaced out as the battery state of charge falls, thus giving it more time to release bound charge between each instance.

4.1 Inria

During the first year of the project, Inria contributed in two directions. First direction has been to develop new SMC algorithms for the efficient detection of rare events. The second contribution is the creation of a toolset.

New Algorithms for detecting 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 statistical model checking (SMC) is to reduce the number and length of simulations necessary to produce a result with a given level of confidence. In the litterature, one finds two techniques to cope with rare events: Importance Sampling and Importance Splitting. However, until now, those techniques have not yet been used to solve model checking problems. During the first year of the project, Inria has adapted both techniques to SMC.

In order to minimize the number of simulations, Importance sampling works by estimating a result using biased simulations and compensating for the bias. For *Importance sampling* to be efficient, it is thus important to find good importance sampling distributions without considering the entire state space. In [25], we present a simple algorithm that uses the notion of cross-entropy 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. We demonstrate the efficacy of our methodology by applying it to models from reliability engineering and to systems biology. One of the major drawbacks with Importance sampling is that the variance of the estimator cannot be characterized. However, our experiments show that our estimator provides variance reductions of more than 10^5.

As we said key objective for statistical model checking rare events is thus to reduce the variance of the estimator. *Importance splitting* achieves this by estimating a sequence of conditional probabilities, whose product is the required result. In [26], we applied *Importance splitting* to model checking problems. To apply this idea to model checking it is necessary to define a *score function* based on logical properties, and a set of *levels* that delimit the conditional probabilities. In [] we motivate the use of importance splitting for statistical model checking and describe the necessary and desirable properties of score functions and levels. We illustrate how a score function may be derived from a property and give two importance splitting algorithms: one that uses fixed levels and one that discovers optimal levels adaptively.

All the algorithms have been implemented in prototypes. Some work needs to be done to make implementations accessible to engineers.

The Plasma Toolset. PLASMA-lab [30] 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[2], and allows rapid prototyping of formal verification solutions using, e.g., Scilab[3] and MATLAB[4]. 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.

[2] www.ales.eu.com

[3] www.scilab.org

[4] www.mathworks.com

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 [34] 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 [34] 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.

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 [30] 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.

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

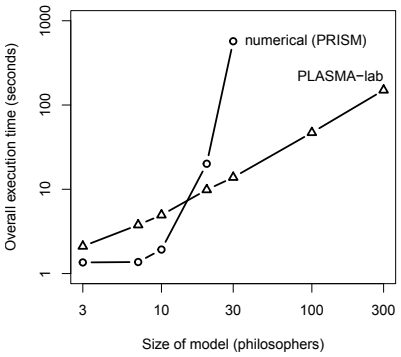

Figure 5: Exponential scaling of numerical model checking vs. linear scaling of PLASMA-lab SMC, considering a fairness property of the probabilistic dining philosophers protocol.

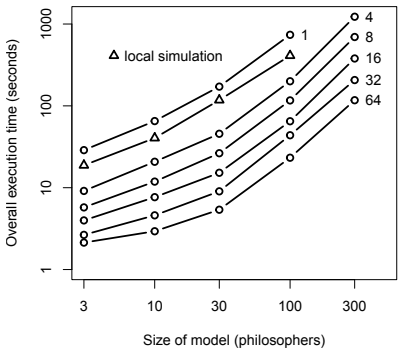

Figure 6: Scaling of PLASMA-lab distributed algorithm applied to dining philosophers. Numbers are quantity of simulation nodes. Local simulation scaling is shown for reference.

that the IP address of the GUI is available to the clients. PLASMA-lab implements the SMC distribution algorithm of [34], which avoids the statistical bias that might otherwise occur from load balancing. Distributed performance in illustrated in Fig. 6. 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

PLASMA-lab has been applied to problems from, e.g., systems biology, rare events, performance, reliability, motion planning and systems of systems [30].

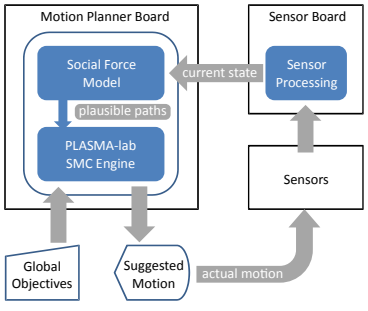

Figure 7: Control loop of DALi motion planner.

4.2 Saarbrucken

Traditional and probabilistic model checking have grown to be useful techniques for finding inconsistencies in designs and computing quantitative aspects of systems and protocols. However, model checking is subject to the state space explosion problem, with probabilistic model checking being particularly affected due to its additional numerical complexity. Several techniques have been introduced to stretch the limits of model checking while preserving its basic nature of performing state space exploration to obtain results that unconditionally, certainly hold for the entire state space. Two of them, partial order reduc-

tion (POR) [3] and confluence reduction [21, 28], work by selecting a subset of the transitions of a model—and thus a subset of the reachable states—in a way that ensures that the reduced system is equivalent to the complete system.

A much different approach for probabilistic models is statistical model checking (SMC)[24, 27, 35]: instead of exploring—and storing in memory—the entire state space, or even a reduced version of it, traces through the state space are generated and evaluated statistically. This comes at constant memory usage and thus circumvents state space explosion entirely, but cannot deliver results that hold with absolute certainty. Statistical methods such as sequential hypothesis testing are then used to make sure that the *probability* of returning the wrong result is below a certain threshold. As a simulation-based approach, however, SMC is limited to fully stochastic models such as Markov chains [22].

Previously, an approach based on POR was presented [4] to extend SMC and simulation to the nondeterministic model of Markov decision processes (MDPs). In that approach, simulation proceeds as usual until a nondeterministic choice is encountered; at that point, an on-the-fly check is performed to find a singleton subset of the available transitions that satisfies the *ample set* conditions of probabilistic POR. If such an ample set is found, simulation can continue that way with the guarantee that ignoring the other transitions does not affect the verification results, i.e., the nondeterminism was *spurious*. Yet, the ample set conditions are based on the notion of *independence* of actions, which can in practice only feasibly be checked on a symbolic/syntactic level. This limits the approach to resolve spurious nondeterminism only when it results from the *interleaving* of behaviours of concurrently executing (deterministic) components.

We proposed to use confluence reduction as an alternative, which has recently been shown theoretically to be more powerful than branching time POR [21]. It is absolutely vital for the search for a valid singleton subset to succeed in the approach discussed above: one choice that cannot be resolved means that the entire analysis fails and SMC cannot safely be applied to the given model at all. Therefore, any additional reduction power is highly welcome. Furthermore, in practice, confluence reduction is easily implemented on the level of the concrete state space alone, without any need to go back to the symbolic/syntactic level for an independence check. As opposed to the POR-based approach, it thus allows even spurious nondeterminism that is internal to components to be ignored during simulation. Of course, models containing non-spurious nondeterminism can still not be dealt with.

Contributions. The three main contributions of our work, published at NFM 2013 [23], are as follows: (1) Since simulation works with a fully composed, closed system, we relax the standard definition of confluence with respect to action labels compared to [21]. We thus achieve more reduction/detection power at no computational cost and have proven that this adapted notion of confluence still preserves PCTL* formulae without the *next* operator. (2) We introduced an algorithm for detecting our new notion of probabilistic confluence on a concrete state space and state its correctness. The algorithm is inspired by, but different from, the one given in [20]; in particular, it does not require initial knowledge of the entire state space and can therefore be used on-the-fly during simulation. (3) We implemented the new approach in the MODES statistical model checker [5] for the MODEST modelling language [6] and evaluated it on a set

of examples. Since the previous POR-based approach is also implemented in MODES, we could compare the two in terms of reduction power and, on the one case that can actually be handled by the POR-based implementation as well, performance.

5 Future Work

Aachen Future work may include modeling the smart grid caes in the compositional modeling language HMoDeST, by comparing simulations with the tool for that language to Uppaal-SMC, Plasma, or by Matlab/Simulink models.

Aalborg Future work is on extending our logic for defining properties over systems with dynamically changing number of components. This logic (QDMTL) is based on MTL extended with means for quantifying over components. Another extension is to support dynamic creation of channels and to pass the channels as values in the model.

Inria We have three main objectives for year two. First objective is to implement rare events algorithms developed during the first year. We will particularly focus on the human/machine interaction. Second objective is to apply rare events algorithms to the challenging experiment proposed by Aachen. Last objective is to interconnect Plasma with the CADP tool of Inria. This will permit to apply SMC on several case studies developed in WP1.

Saarbrucken An obvious direction for future work is to compare our techniques to the fundamentally different learning-based approach of Henriques et al. on models where all three methods are applicable.

References

[1] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. Approximate model checking of stochastic hybrid systems. *European J. of Control*, 16(6):1–18, 2010.

[2] A. Abate, J.-P. Katoen, and A. Mereacre. Quantitative automata model checking of autonomous stochastic hybrid systems. In *HSCC*, pages 83–92. ACM, 2011.

[3] C. Baier, P. R. D'Argenio, and M. Größer. Partial order reduction for probabilistic branching time. *ENTCS*, 153(2), 2006.

[4] J. Bogdoll, L. M. F. Fioriti, A. Hartmanns, and H. Hermanns. Partial order methods for statistical model checking and simulation. In *FMOODS/FORTE*, volume 6722 of *LNCS*, pages 59–74. Springer, 2011.

[5] J. Bogdoll, A. Hartmanns, and H. Hermanns. Simulation and statistical model checking for Modestly nondeterministic models. In *MMB/DFT*, volume 7201 of *LNCS*, pages 249–252. Springer, 2012.

[6] H. C. Bohnenkamp, P. R. D'Argenio, H. Hermanns, and J.-P. Katoen. MoDeST: A compositional modeling formalism for hard and softly timed systems. *IEEE Transactions on Software Engineering*, 32(10):812–830, 2006.

[7] P. Bulychev, A. David, K. G. Larsen, A. Legay, G. Li, and D. B. Poulsen. Rewrite-based statistical model checking of wmtl. In *Runtime Verification*, volume 7687 of *LNCS*, pages 260–275, 2012.

[8] P. Bulychev, A. David, K. G. Larsen, A. Legay, G. Li, D. B. Poulsen, and A. Stainer. Monitor-based statistical model checking for weighted metric temporal logic. In N. Bjørner and A. Voronkov, editors, *18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 7180 of *LNCS*, pages 168–182. Springer, 2012.

[9] P. Bulychev, A. David, K. G. Larsen, A. Legay, and M. Mikucionis. Distributed parametric and statistical model checking. In K. H. Jiří Barnat, editor, *Proceedings 10th International Workshop on Parallel and Distributed Methods in Verification*, volume 72 of *EPTCS*, pages 30–42.

[10] P. Bulychev, A. David, K. G. Larsen, A. Legay, M. Mikucionis, and D. B. Poulsen. Checking & distributing statistical model checking. In A. E. Goodloe and S. Person, editors, *4th NASA FORMAL METHODS SYMPOSIUM*, volume 7226 of *LNCS*, pages 449–463. Springer, 2012.

[11] P. Bulychev, A. David, K. G. Larsen, M. Mikučionis, D. Bøgsted Poulsen, A. Legay, and Z. Wang. Uppaal-smc: Statistical model checking for priced timed automata. In H. Wiklicky and M. Massink, editors, Proceedings 10th Workshop on *Quantitative Aspects of Programming Languages and Systems,* Tallinn, Estonia, 31 March and 1 April 2012, volume 85 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–16. Open Publishing Association, 2012.

[12] A. David, D. Du, K. G. Larsen, A. Legay, M. Mikucionis, D. Poulsen, and S. Sedwards. Statistical model checking for stochastic hybrid systems. In *Workshop on Hybrid Systems and Biology (HSB)*, volume 92 of *EPTCS*, pages 122–136, 2012.

[13] 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, *1st International Workshop on Hybrid Systems and Biology*, volume 92 of *Electronic Proceedings in Theoretical Computer Science*, pages 122–136. Open Publishing Association, 2012.

[14] A. David, D. Du, K. G. Larsen, M. Mikucionis, and A. Skou. An evaluation framework for energy aware buildings using statistical model checking. *SCIENCE CHINA Information Sciences*, 55(12):2694–2707, 2012.

[15] A. David, K. G. Larsen, A. Legay, and M. Mikucionis. Schedulability of herschel-planck revisited using statistical model checking. In *ISoLA (2)*, volume 7610 of *LNCS*, pages 293–307. Springer, 2012.

[16] A. David, K. G. Larsen, A. Legay, M. Mikucionis, D. B. Poulsen, and S. Sedwards. Runtime verification of biological systems. In *ISoLA*, volume 7609 of *LNCS*, pages 388–404. Springer, 2012.

[17] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and Z. Wang. Time for statistical model checking of real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 349–355. Springer, 2011.

[18] A. David, K. G. Larsen, A. Legay, and D. B. Poulsen. Statistical model checking of dynamic networks of stochastic hybrid automata. In *Proceedings of the Automated Verification of Critical Systems (AVoCS2013)*, volume to appear, page to appear, 2013.

[19] J. Gordon and K. Ng. Predictive and diagnostic aspects of a universal thermodynamic model for chillers. *Int. J. of Heat and Mass Transfer*, 38(5):807–818, 1995.

[20] J. F. Groote and J. C. van de Pol. State space reduction using partial tau-confluence. In *MFCS*, volume 1893 of *LNCS*, pages 383–393. Springer, 2000.

[21] H. Hansen and M. Timmer. A comparison of confluence and ample sets in probabilistic and non-probabilistic branching time. *Accepted for TCS.*, 2013.

[22] A. Hartmanns. Model-checking and simulation for stochastic timed systems. In *FMCO*, volume 6957 of *LNCS*, pages 372–391. Springer, 2010.

[23] A. Hartmanns and M. Timmer. On-the-fly confluence detection for statistical model checking. In G. Brat, N. Rungta, and A. Venet, editors, *NASA Formal Methods*, volume 7871 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2013.

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

[25] C. Jégourel, A. Legay, and S. Sedwards. Cross-entropy optimisation of importance sampling parameters for statistical model checking. In *CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2012.

[26] C. Jégourel, A. Legay, and S. Sedwards. Importance splitting for statistical model checking rare properties. In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 576–591. Springer, 2013.

[27] A. Legay, B. Delahaye, and S. Bensalem. Statistical model checking: An overview. In *RV*, volume 6418 of *LNCS*, pages 122–135. Springer, 2010.

[28] M.Timmer, M. I. A. Stoelinga, and J. C. van de Pol. Confluence reduction for probabilistic systems. In *TACAS*, volume 6605 of *LNCS*, pages 311–325. Springer, 2011.

[29] A. Parisio and L. Glielmo. Energy efficient microgrid management using model predictive control. In *50th IEEE Conf. on Decision and Control and European Control Conference (CDC-ECC 2011)*, pages 5449–5454, 2011.

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

[31] F. Ramponi, D. Chatterjee, S. Summers, and J. Lygeros. On the connections between PCTL and dynamic programming. In *HSCC*, pages 253–262. ACM, 2010.

[32] M. Tigges. *Modellbasierte Analyse zur Verbesserung der elektrischen Energiebereitstellung zuknftiger Offshore-Windparks mittels Biogastechnologie*. PhD thesis, Universitaet Paderborn, 2010.

[33] H. Younes and R. G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, 204(9):1368–1409, 2006.

[34] H. L. S. Younes. Verification and planning for stochastic processes with asynchronous events. PhD thesis, Carnegie Mellon University, 2005.

[35] H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, volume 2404 of *LNCS*, pages 223–235. Springer, 2002.