

# Statistical Model Checking for SystemC Models



Chan Ngo

Axel Legay

Jean Quilbeuf

INRIA, Rennes, France

HASE 2016

## Executive Summary

---

- **SystemC** has been become increasingly prominent in describing the behavior of embedded systems, i.e., System-on-Chips (SoCs)
- **State Space Exploration** is infeasible for large systems
  - Partial Order Reduction
  - Symbolic Model Checking
  - SAT-based Bounded Model Checking
  - Predicate Abstraction
  - Counterexample Guided Abstraction Refinement
- Often easier to simulate a system: **Statistical Model Checking**
- Our goal: Provide **probabilistic guarantees of correctness** of stochastic SystemC models using a number of simulations
  - How to define an execution trace?
  - How to generate each execution trace?
  - How many simulation runs to make?

## An Example

---



- The message's length and FIFO's buffer are **fixed** (i.e., of **10**)
- Producer writes 1 character to the FIFO with **probability  $p_1$**  every 1 time unit
- Consumer reads 1 character from the FIFO with **probability  $p_2$**  every 1 time unit
- What is the **probability that messages are transferred completely** within **15** time units during **10000** time units of operation? (**Quantitative Analysis**)
- Is this probability **at least 0.6**? (**Qualitative Analysis**)

## A Solution - Probabilistic Model Checking

---

- Given a **stochastic model**  $\mathcal{M}$  such as a Markov chain
- A property  $\varphi$  expressed in **Bounded Linear Temporal Logic** (BLTL) and a probability threshold  $\theta \in (0, 1)$
- Does  $\mathcal{M}$  satisfy  $\varphi$  with probability at least  $\theta$ ?

$$\mathcal{M} \models Pr_{\geq \theta}(\varphi)$$

**Example:** Messages are transferred completely within  $T_1$  time units during  $T$  time units of operation

$$G_{\leq T}((c\_read = '\&') \rightarrow F_{\leq T_1}(c\_read = '@'))$$

## PMC - Scalability

---

- PMC is infeasible for large systems due to **State Space Exploration**
  - PMC employs Symbolic Model Checking which can scale to  $\sim 10^{100}$  states)
  - **Scalability** depends on the structure of the system
- PMC does not **work directly** with SystemC models
  - Formal model is sometime much **over-approximated**. It cannot capture the concrete implementation of the system
- **Example**: PRISM probabilistic model checker created at Oxford and Birmingham

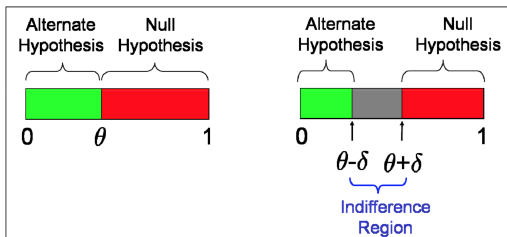
## Another Solution - Statistical Model Checking

---

- Associate the  $i$ th execution trace with a random variable  $B_i$  having a Bernoulli distribution, i.e.,  $Pr[B_i = 1] = p$  and  $Pr[B_i = 0] = 1 - p$
- An observation  $b_i = 1$  of  $B_i$  if the trace satisfies the property,  $b_i = 0$  otherwise
- Decide between two hypotheses:
  - Null hypothesis:  $H : p \geq \theta$
  - Alternate hypothesis:  $K : p < \theta$
- Or estimate the probability  $p$  instead of hypothesis testing
- Feature:
  - Pros: Simulation is feasible for many more systems
  - Pros: Easier to parallelize
  - Cons: Answers may be wrong. But error probability can be bounded (i.e., at most  $\alpha \sim 0$ )
  - Cons: Simulation is incomplete

## SMC - Existing Work

- [Younes et al. 06, [CMU](#)] use Wald's [Sequential Probability Ratio Test](#)
- The SPRT with an [indifferent region](#)  $2\delta$  decides between
  - The [simple null](#) hypothesis  $H_0 : p \geq p_0 = \theta + \delta$
  - The [simple alternate](#) hypothesis  $H_1 : p < p_1 = \theta - \delta$



- [Plasma Lab, [INRIA](#)] checker uses MonteCarlo method, Chernoff and Hoeffding bounds, etc. to estimate the probability

$$\tilde{p} = \frac{1}{n} \sum_{i=1}^n b_i \text{ such that } Pr[|\tilde{p} - p| < \delta] \geq 1 - \alpha$$

## Bounded Linear Temporal Logic

---

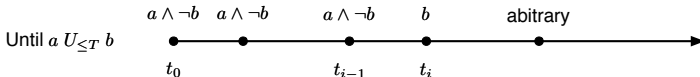
- **Bounded Linear Temporal Logic** (BLTL): Extension of LTL with time bounds on temporal operations (i.e.,  $\varphi_1 U_{\leq T} \varphi_2$ )
- Let  $\omega = (s_0, t_0), (s_1, t_1), \dots$  be an execution of the model
  - along states  $s_0, s_1, \dots$
  - the system stays in the state  $s_i$  for time  $t_i$
- $\omega^i$ : Suffix execution trace starting at state  $i$
- $V(\omega, i, v)$ : Value of the variable  $v$  at the state  $s_i$
- A natural model for SystemC traces
  - SystemC has discrete time semantics (i.e., time unit by setting **time resolution**)



# Semantics of BLTL

The semantics of BLTL for a trace suffix  $\omega^k$ :

- $\omega^k \models \text{true}$  and  $\omega^k \not\models \text{false}$
- $\omega^k \models p, p \in AP$  iff  $p \in L(s_k)$
- $\omega^k \models \varphi_1 \wedge \varphi_2$  iff  $\omega^k \models \varphi_1$  and  $\omega^k \models \varphi_2$
- $\omega^k \models \neg \varphi$  iff  $\omega^k \not\models \varphi$
- $\omega^k \models \varphi_1 U_{\leq T} \varphi_2$  iff there exists an integer  $i$  such that
  - $\omega^{k+i} \models \varphi_2$
  - $\sum_{0 < j \leq i} (t_{k+j} - t_{k+j-1}) \leq T$
  - for each  $0 \leq j < i, \omega^{k+j} \models \varphi_1$



## SystemC Model State

---

A state is an evaluation of a set of variables  $V$  which consists of:

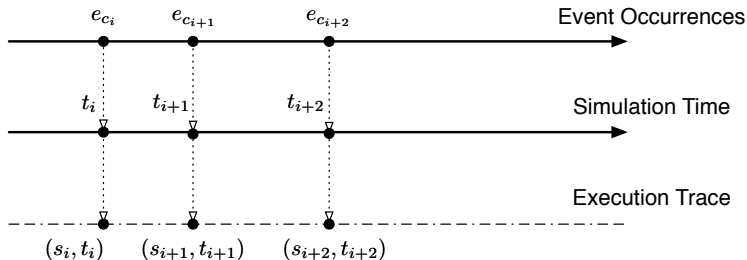
- Simulation kernel state
  - **Current phase** of the simulation scheduler (i.e., delta-cycle notification, simulation-cycle notification)
  - **Events notified** during the execution of the model
- SystemC model state: Full state of the C++ code
  - All module's attributes,
  - Location of the **program counter** (i.e., executed statement, function call)
  - Call stack (i.e., function parameters and return values)
  - Status of module processes

**Note:** External libraries are considered as **block boxes**

# Temporal Resolution

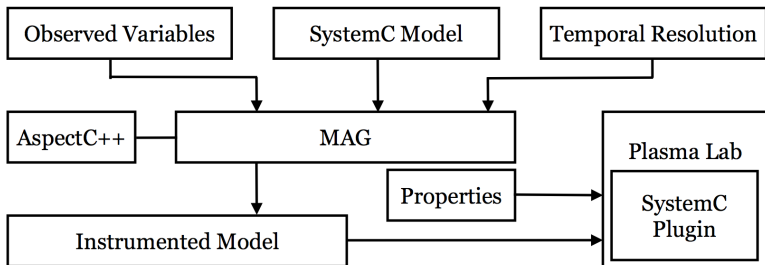
- Temporal resolution is a set of Boolean expressions, called **temporal events**, are defined over the simulation kernel state, location of the program counter, and processes' status
- Whenever a temporal event is **true**, a **new state** is sampled
- A state is a **snapshot** of system at **event occurrence**  $e_{c_i}$

$$s_i = (V(\omega, i, v_0), \dots, V(\omega, i, v_{n-1})) = (\xi_{val}^{v_0}(e_{c_i}), \dots, \xi_{val}^{v_{n-1}}(e_{c_i}))$$



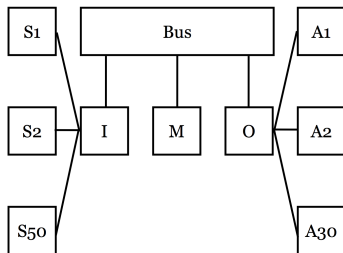
## SMC for SystemC Models

---



- **MAG**: Automatically instruments SystemC model with the help of AspectC++ in order to generate the traces and communicate with the checker
- **SystemC Plugin**: Communicates with the instrumented model and applies appropriate statistical algorithms provided by Plasma Lab

## Case Study - Dependability Analysis



- 50 groups of 3 sensors, 30 groups of 2 actuators
- Main, input and output processors communicate via a reliable bus

Component	Mean time
Sensor	1 month
Actuator	2 months
Transient Fault	1 day
Processor	1 year
Reboot to Repair	30 seconds

## Case Study - Dependability Analysis

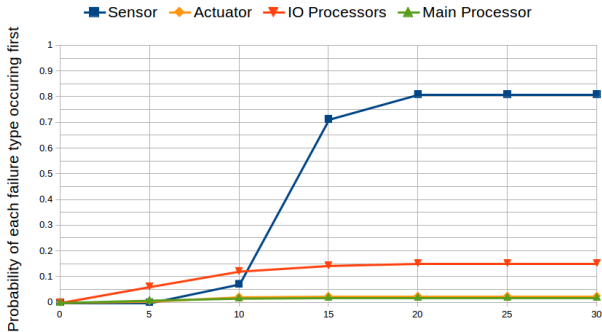
---

- Time to failure of sensors, actuators and processors and time to repair of I/O processors can be modeled by exponential distributions
- The reliability is modelled as a Continuous Time Markov Chain (CTMC)
  - Sensor group: 4 states
  - Actuator group: 3 states
  - Main processor: 2 states
  - I/O processors: 3 states (including 1 state of transient failure)
  - The model has  $4^{50} \times 3^{30} \times 2 \times 3^2 \sim 2^{150}$  states

## Dependability Analysis - Results

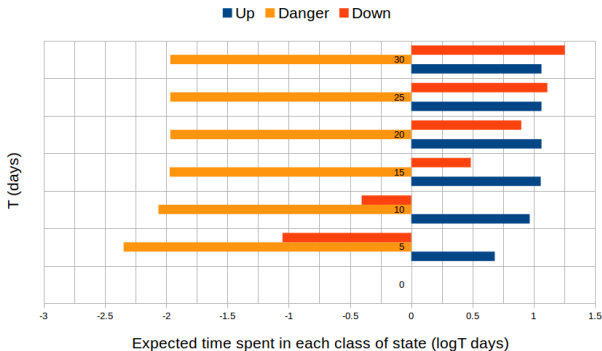
The probability that each of the 4 failure types is the cause of system shutdown in the first  $T$  time of operation

$$shutdown = \bigvee_{i=1}^4 failure_i \quad \neg shutdown \quad U_{\leq T} failure_i$$



## Dependability Analysis - Results

- The expected amount of time spent in each of the states: “up”, “danger” and “shutdown”
- $X_{\leq T} \text{reward\_c}$  returns the mean value of *reward\_c* after  $T$  time of execution





# Tool in Action

The screenshot displays the Inria PlasmaLab tool interface, which is used for configuring and running simulations. The interface is divided into several panels:

- Explorer:** A tree view on the left showing the project structure, including models, requirements, and various components like sensors, actuators, and processes.
- File selection:** A panel for selecting the project and model. The current selection is 'Project: ecs' and 'Model: ecs\_model'.
- Requirements:** A list of requirements for the simulation, including 'ev\_actuators\_43200', 'ev\_actuators\_57600', 'ev\_actuators\_72000', and 'ev\_actuators\_86400'.
- Algorithm:** A dropdown menu set to 'Montecarlo' and a 'Total samples' field set to '1,000'.
- Execution:** A section for configuring the execution environment, including options for 'distributed' or 'local' execution, 'Port', 'Threads', and 'Batch'.
- Experimentation results:** A table on the right showing the results of the simulation. The table has columns for 'Name', '# Simulations', '# Positive Simulation', and 'Result'.

The 'Experimentation results' table contains the following data:

#	Name	# Simulations	# Positive Simulation	Result
68	261nw_up_28800	1,000	26156021	26156.021
69	320nw_up_43200	1,000	32053084	32053.084
70	323nw_up_57600	1,000	32349305	32349.305
71	323nw_up_72000	1,000	32349305	32349.305
72	323nw_up_86400	1,000	32349305	32349.305
73	13nw_danger_14400	1,000	13005	13.005
74	24nw_danger_28800	1,000	24503	24.503
75	30nw_danger_43200	1,000	30021	30.021
76	30nw_danger_57600	1,000	30313	30.313
77	30nw_danger_72000	1,000	30313	30.313
78	30nw_danger_86400	1,000	30313	30.313
79	301nw_shutdown_14400	1,000	301998	301.998
80	133nw_shutdown_28800	1,000	1339941	1339.941
81	917nw_shutdown_43200	1,000	9171632	9171.632
82	232nw_shutdown_57600	1,000	23249379	23249.379
83	376nw_shutdown_72000	1,000	37649379	37649.379
84	520nw_shutdown_86400	1,000	52049379	52049.379
85	46ev_sensors_14400	1,000	46757	46.757
86	40ev_sensors_28800	1,000	40166	40.166
87	32ev_sensors_43200	1,000	32829	32.829
88	25ev_sensors_57600	1,000	25982	25.982
89	20ev_sensors_72000	1,000	20145	20.145
90	15ev_sensors_86400	1,000	15336	15.336
91	29ev_actuators_14400	1,000	29820	29.82
92	29ev_actuators_28800	1,000	29316	29.316
93	28ev_actuators_43200	1,000	28560	28.56
94	27ev_actuators_57600	1,000	27619	27.619
95	26ev_actuators_72000	1,000	26581	26.581
96	25ev_actuators_86400	1,000	25444	25.444

The bottom status bar shows the current project is 'ecs', the simulation is running, and the optimization is complete. The date and time are 1.3.0-2015-03-16 14:12:55.

<https://project.inria.fr/plasma-lab/>

# Conclusions

---

- An introduction about [Statistical Model Checking](#)
- Some evidence that SMC scales to [large](#) systems
  - SystemC models
  - Simulink models [Clarke et al. 09, CMU]
- Initial experiments on SystemC for dependability analysis are carried out
- Plan:
  - More SystemC examples
  - [Parallel implementation](#) of the statistical analyzer
  - Consider an implementation of [Random Scheduler](#) for SystemC kernel

Questions? :-)