

Statistical Model Checking for SystemC Models

Van Chan Ngo Axel Legay Jean Quilbeuf

Inria Rennes, France

HASE 2016, Orlando, FL

Executive summary

- PMC: **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
- SMC: often easier to simulate a system
- 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



- 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
- Quantitative analysis: What is the **probability that messages are transferred completely** within 15 time units during 10000 time units of operation?
- Qualitative analysis: Is this probability **at least 0.6**?

A solution - PMC

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

$$\mathcal{M} \models \text{Pr}_{\geq \theta}(\varphi)$$

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

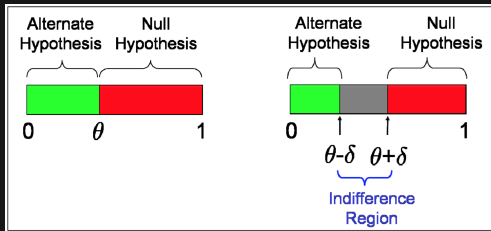
- PMC is infeasible for large systems due to the **state space exploration**
 - PMC employs symbolic model checking which can scale to $\sim 10^{100}$ states
 - Scalability** depends on the structure of the system
- PMC does not **work directly** with SystemC models
 - Formal model is sometime much **over-approximated**.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

Another solution - SMC

- Associate the i^{th} execution trace with a random variable B_i having a Bernoulli distribution,
 $\Pr[B_i = 1] = p, \Pr[B_i = 0] = 1 - p$
- An observation $b_i = 1$ 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
Simulation is feasible for many more systems
Easier to parallelize
Answers may be wrong. But error probability can be bounded (i.e. at most $\alpha \sim 0$)
Simulation is incomplete

SMC - Existing work

- [Younes et al. 06, CMU] use Wald's Sequential Probability Ratio Test
- 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

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

BLTL

- An extension of LTL with time bounds on temporal operations (i.e. $\varphi_1 U_{\leq T} \varphi_2$)

- The **semantics** of BLTL for a trace suffix

$\omega^k = (s_k, t_k), (s_{k+1}, t_{k+1}), \dots$ is defined as follows

$$\omega^k \models \text{true} \text{ and } \omega^k \not\models \text{false}$$

$$\omega^k \models p, p \in \text{AP} \text{ iff } p \in L(s_k)$$

$$\omega^k \models \varphi_1 \wedge \varphi_2 \text{ iff } \omega^k \models \varphi_1 \text{ and } \omega^k \models \varphi_2$$

$$\omega^k \models \neg \varphi \text{ iff } \omega^k \not\models \varphi$$

$$\omega^k \models \varphi_1 U_{\leq T} \varphi_2 \text{ iff there exists an integer } i \text{ such that}$$

$$\omega^{k+i} \models \varphi_2$$

$$\sum_{0 < j \leq i} (t_{k+j} - t_{k+j-1}) \leq T$$

$$\text{for each } 0 \leq j < i, \omega^{k+j} \models \varphi_1$$

SystemC model state

A state is an evaluation of variables which represent

- 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)

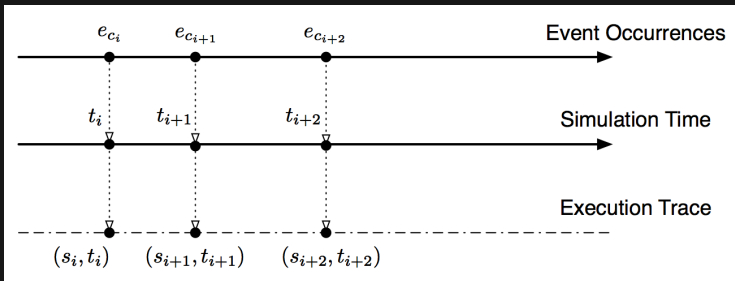
- Call stack (i.e. function parameters and return values)

- Status of module processes

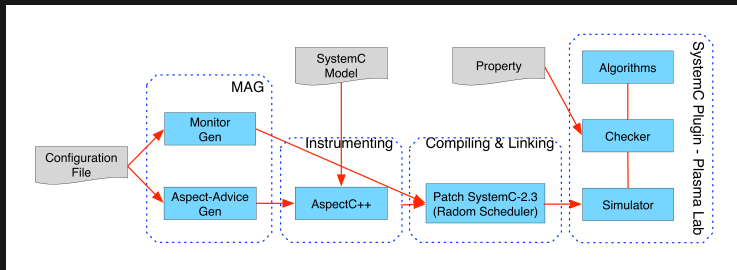
Note: external libraries are considered as block boxes

Temporal resolution

- Is a set of Boolean expressions, called **temporal events**, defined over the simulation kernel states, location of the program counter, and processes' status
- Whenever a temporal event is **true**, a **new state** is sampled
- A time unit is duration between two **event occurrences**
- States are **snapshots** of system at event occurrences



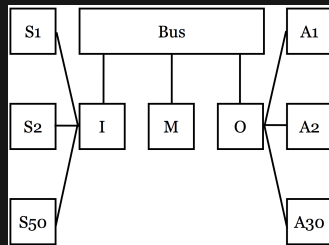
SMC for SystemC models



- **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	<i>1 month</i>
Actuator	<i>2 months</i>
Transient Fault	<i>1 day</i>
Processor	<i>1 year</i>
Reboot to Repair	<i>30 seconds</i>

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

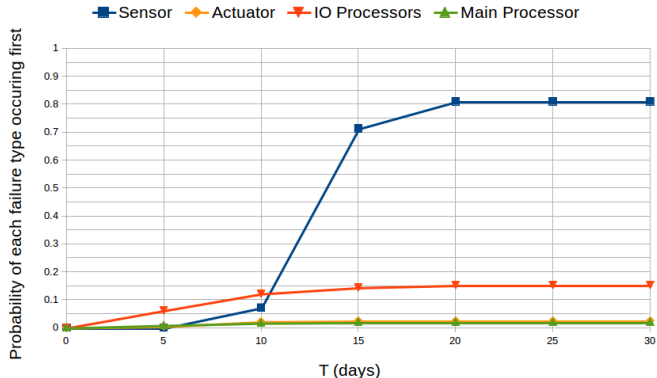
I/O processors: 3 states (including 1 state of transient failure)

The model has $4^{50} \times 3^{30} \times 2 \times 3^2 \sim 2^{150}$ states

Results

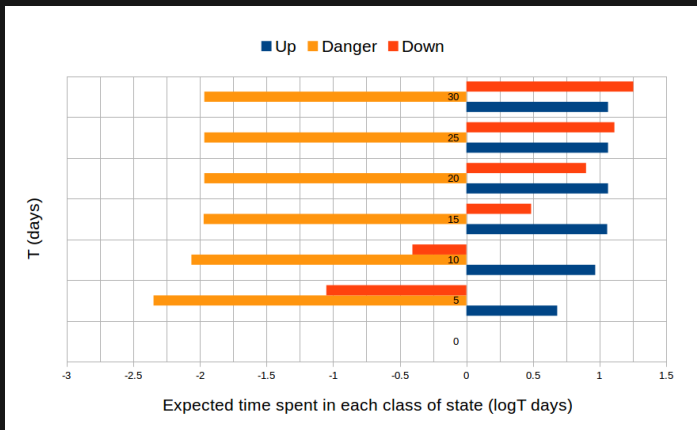
The probability that each of the 4 failure types is the cause of system shutdown in the first T time of operation

$$\text{shutdown} = \bigvee_{i=1}^4 \text{failure}_i \quad \neg \text{shutdown} \bigwedge_{i=1}^4 \neg \text{failure}_i$$



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 of **reward_c** after **T** time of execution



Tool in action

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

- File selection:** Shows the project name 'ecs' and the model 'ecs_model'.
- Requirements:** Lists various components such as 'ev_actuators_43200', 'ev_actuators_57600', 'sensor_28800', 'sensor_43200', 'sensor_57600', 'sensor_72000', 'sensor_86400', 'fl_sensor_14400', 'fl_sensor_28800', 'fl_sensor_43200', 'fl_sensor_57600', 'fl_sensor_72000', 'fl_sensor_86400', 'actuator_14400', 'actuator_28800', 'actuator_43200', 'actuator_57600', 'actuator_72000', 'actuator_86400', 'fl_actuator_14400', 'fl_actuator_28800', 'fl_actuator_43200', 'fl_actuator_57600', 'fl_actuator_72000', 'fl_actuator_86400', 'io_14400', 'io_28800', 'io_43200', 'io_57600', 'io_72000', 'io_86400', 'fl_io_14400', 'fl_io_28800', 'fl_io_43200', 'fl_io_57600', 'fl_io_72000', 'fl_io_86400', 'proc_m_14400', 'proc_m_28800', 'proc_m_43200', 'proc_m_57600', 'proc_m_72000', 'proc_m_86400', 'fl_proc_m_14400', 'fl_proc_m_28800', 'fl_proc_m_43200', 'fl_proc_m_57600', 'fl_proc_m_72000', 'fl_proc_m_86400', 'down_14400', 'down_28800'.
- Algorithm:** Set to 'MonteCarlo' with 'Total samples: 1000'.
- Execution:** Options for 'distributed' and 'local' execution. The 'local' option is selected. Parameters include 'Part: 8111', 'Threads: 1', and 'Batch: 500'. A 'Start' button is visible.
- Experimentation results:** A table showing the results of the simulation. The table has columns for '# Simulations', '# Positive Simulation', and 'Result'. The results are sorted by the number of positive simulations, with the top result being 26156.021.

The status bar at the bottom indicates the current state: 'ecs Simulation Experimentation Optimization' and 'fr.inria.plasmalab.brl'. The version number '1.3.0-2015-03-16 14:12:55' is also displayed.

<https://project.inria.fr/pscv/>

Conclusion

- An introduction about **Statistical Model Checking**
- Some evidences 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

Future work

- More SystemC examples
- **Parallel implementation** of the statistical analyzer
- Consider the implementation of a **random scheduler** for SystemC kernel