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 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?

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

Example: Messages are transfered 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

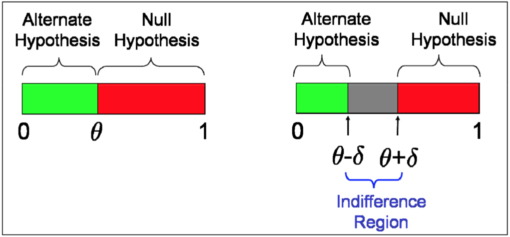
- PMC is infeasible for large systems due to State Space Exploration
  - PMC employs Symbolic Model Checking which can scale to $\sim 10^{100}$ states)
  - Scalability depends on the structure of the system
- PMC does not work directly with SystemC models
  - Formal model is sometime much over-approximated. 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 ith 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 2δ 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

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

Bounded Linear Temporal Logic

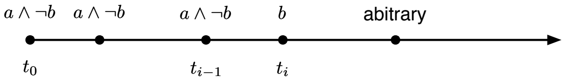
- Bounded Linear Temporal Logic (BLTL): Extension of LTL with time bounds on temporal operations (i.e., $\varphi_1 \; U_{\leq T} \; \varphi_2$)
- Let $\omega = (s_0, t_0), (s_1, t_1), \ldots$ be an execution of the model
  - along states $s_0, s_1, \ldots$
  - the system stays in the state s_i for time t_i
- ω^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 ω^k:

- $\omega^k \models \textit{true}$ and $\omega^k \not\models \textit{false}$
- $\omega^k \models p, p \in AP$ 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
  - $\omega^{k+i} \models \varphi_2$
  - $\Sigma_{0<j\leq i}(t_{k+j} - t_{k+j-1}) \leq T$
  - for each $0 \leq j < i, \omega^{k+j} \models \varphi_1$

Until $a \; U_{\leq T} \; b$



SystemC Model State

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

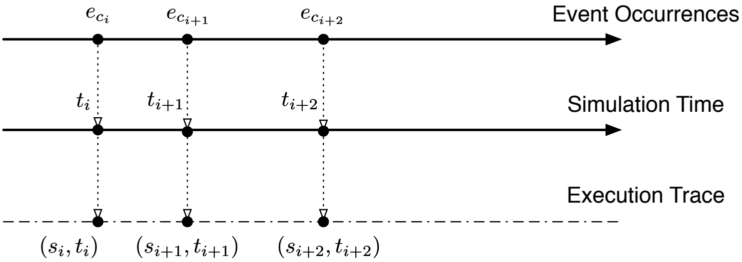
- Simulation kernel state
  - Current phase of the simulation scheduler (i.e., delta-cycle notification, simulation-cycle notification)
  - Events notified during the execution of the model
- SystemC model state: Full state of the C++ code
  - All module's attributes,
  - Location of the program counter (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

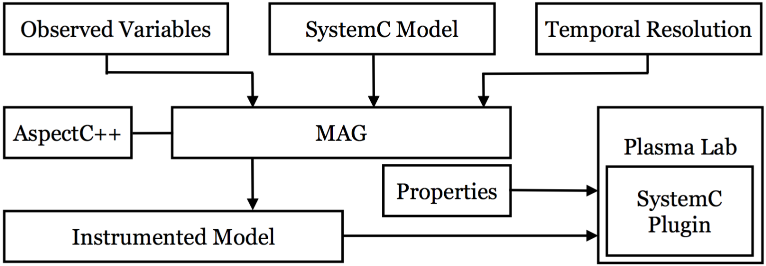
Temporal Resolution

- 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 sate is a snapshot of system at event occurrence e_{c_i}

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

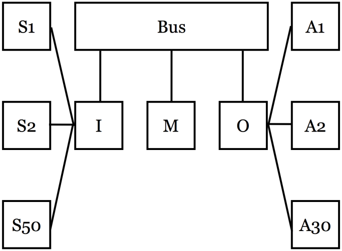


SMC for SystemC Models



- 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*

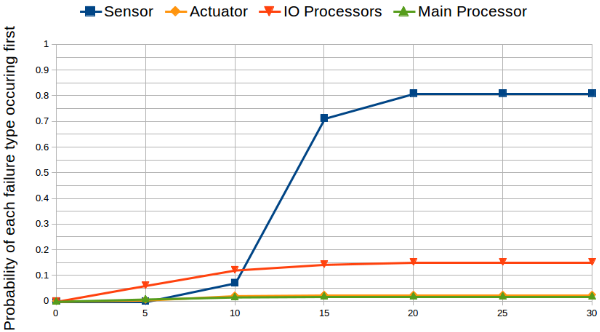
Case Study - Dependability Analysis

- 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

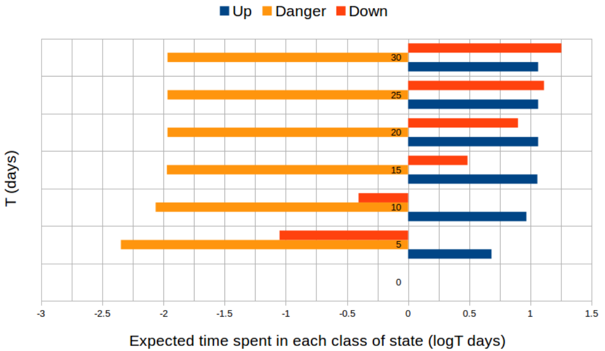
The probability that each of the 4 failure types is the cause of system shutdown in the first T time of operation

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

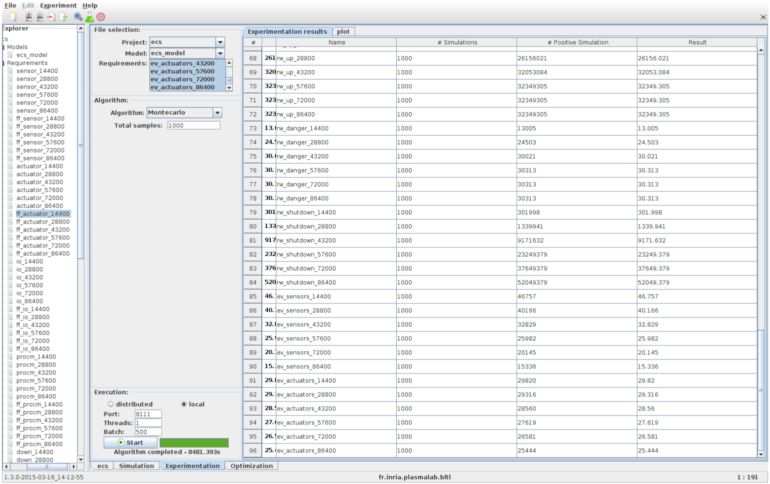


Dependability Analysis - Results

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



Tool in Action



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? :-)