Artifact • AEC • Evaluated • CAV • Consistent • Complete • Well Documented • Easy to Reuse

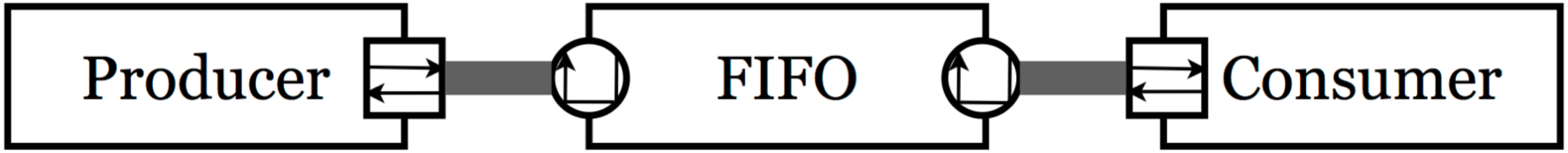
PSCV: A Runtime Verification Tool for Probabilistic SystemC Models

Van Chan Ngo
Axel Legay
Vania Joloboff

Carnegie Mellon University
INRIA Rennes

CAV 2016, Toronto, Canada

An example



- Message and FIFO buffer sizes are fixed, i.e. of 10
- Every 1 time unit, Producer writes (Consumer reads) 1 character to (from) the FIFO with probability p1 (p2)
- Quantitative analysis: Over 10000 time units of operation, what is the probability that messages are transferred completely by 15 time units?
- Qualitative analysis: Is this probability at least 0.6?

A solution - PMC

- Use *probabilistic model checking* such as PRISM* to verify the program model against the properties
- Main issues
  - PMC is infeasible for large systems due to the state space explosion
  - Translation from SystemC programs into formal models is not trivial
  - Time model is not fine-grained enough, i.e. very difficult to express time as the number of calls to a function

* http://www.prismmodelchecker.org

Another solution - SMC

- Use *statistical model checking* to verify properties expressed with bounded temporal operators
- Probability estimation, i.e. *MonteCarlo* method, *Chernoff* and *Hoeffding* bounds for quantitative analysis
- Hypothesis testing, i.e. *Sequential Probability Ratio Test* (SPRT) for qualitative analysis
  - Simulation is feasible for many large programs
  - Easier to parallelize
  - Answers may be wrong. However, error probability can be bounded (level of statistic confidence)
  - Simulation is incomplete (cannot cover all inputs)

PSCV - Main features

- SMC-based tool works directly with programs written in SystemC
- Required number and length of execution traces are finite
- A rich set of properties: A wide range of abstraction* from statement level to system level
- A more fine-grained model of time than the cycle-based simulation
- A random scheduler rather than the deterministic one in the current SystemC kernel

* Tabakov, D. and Vardi, M. : Monitoring Temporal SystemC Properties. In Formal Methods and Models for Codesign, 2010

State and execution trace

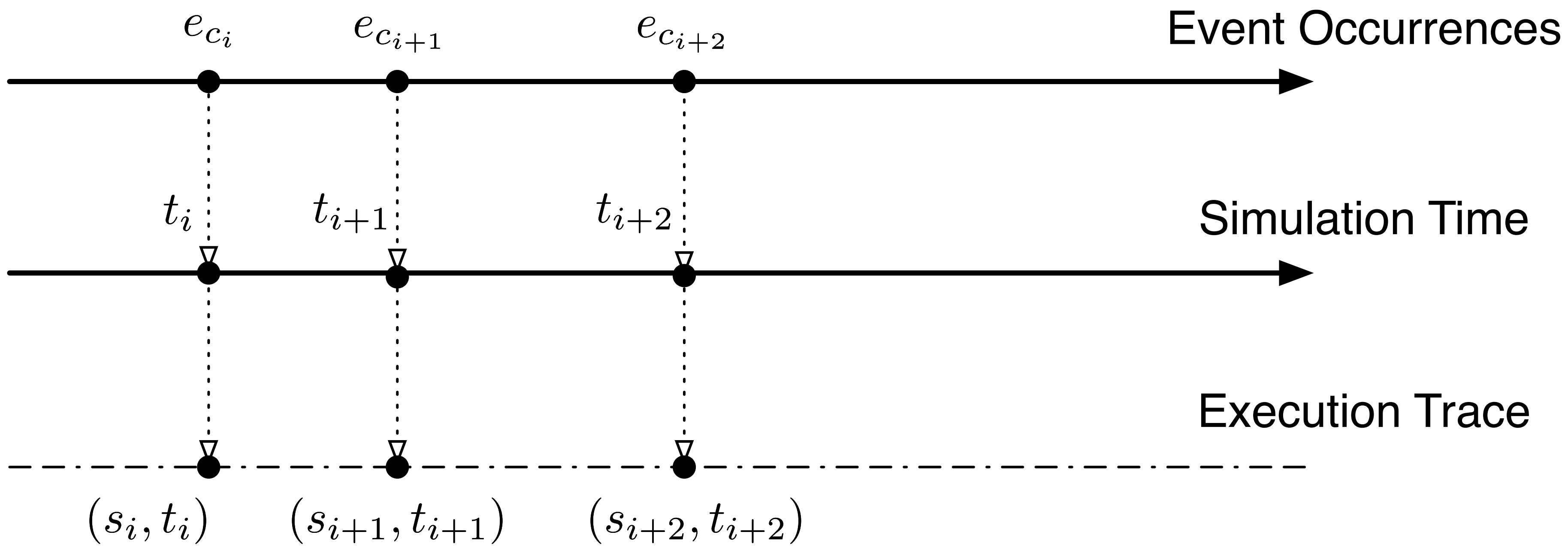
A state is an evaluation of observed variables which represent

- Simulation kernel state
  - Current phase of the simulation scheduler, i.e. delta-cycle, simulation-cycle notification phases
  - Events notified during the execution of the program
- Program state: full state of C++ code
  - All module’s attributes
  - Program counter, i.e. executing statement, function call
  - Call stack, i.e. function arguments and return values
  - Status of module processes

An execution trace is a sequence of state along with simulation time

Model of time - Temporal resolution

- A disjunction of Boolean expressions, called temporal events, defined over kernel state, location of program counter, and process' status
- Whenever a temporal event is true, a new state is sampled
- Time unit is the duration between two event occurrences
- States are snapshots of program at event occurrences



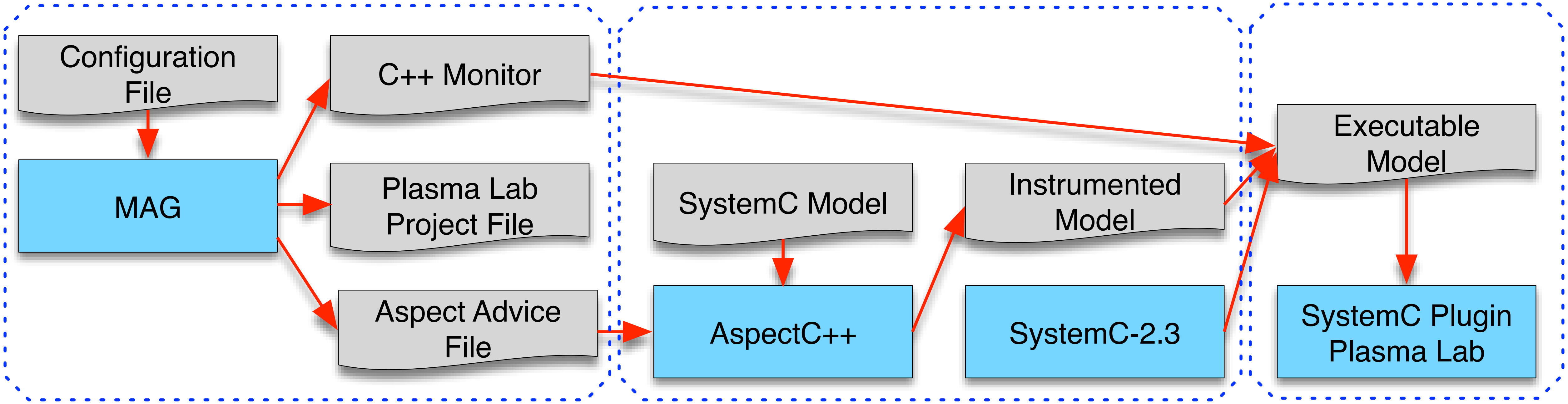
Expressing properties

- Properties are of the forms $\Pr(\varphi)$ and $\Pr_{\geq\theta}(\varphi)$ where φ is a BLTL formula and θ is a threshold
- BLTL is an extension of LTL with time bounds on temporal operators
- For example, the formula of the running example is

$$\varphi = \mathrm{G}_{\leq 10000}((\mathrm{c\_read} = \text{‘\&‘}) \Rightarrow \mathrm{F}_{\leq 15}(\mathrm{c\_read} = \text{‘@‘}))$$

  - c_read is observed variable representing the current character read by Consumer
  - Simulation-cycle notification phase is defined as temporal resolution
  - & and @ are starting and ending delimiters of a message

Verification flow



- Configuration file contains observed variables, time resolution and properties
- MAG generates the monitor, aspect-advices used for automatically instrumenting with AspectC++, and Plasma Lab project file
- SystemC Plugin built on top of the SMC checker Plasma Lab verifies the properties

Blue blocks are components of PSCV

Give it a try!

- Implementation and case studies are available at the project website

  https://project.inria.fr/pscv

- A short tutorial including writing configuration files is also available at the project website

- Plasma Lab and its documents are obtained at

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