Statistical Model Checking and Uppaal SMC

Formal Methods for Concurrent and Real-Time Systems, A.Y. 20/21

Livia Lestingi

25 MAY 2021

Why *Statistical* Model Checking?

(Some) Issues with numerical techniques:

- If the system has complex dynamics, basic problems may not have efficient solution methods
- They require an explicit description of system implementation
- They do not scale!

Why *Statistical* Model Checking?

Statistical techniques to the rescue!

- Applicable to any system, as long as its behavior is stochastic...
- ...and we have samples
- The state-space explosion problem can often be elided

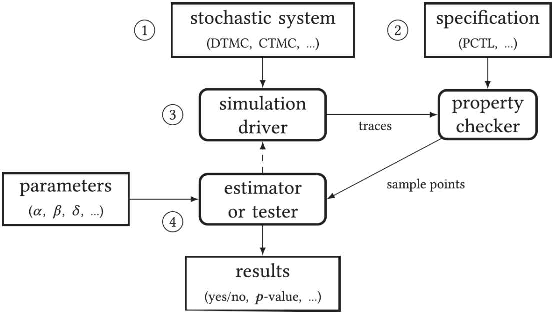
Some facts about SMC:

- Research active for more than 15 years since 2002-2005 (but some key ideas can be traced back to a decade before)
- Many dedicated tools developed (14 surveyed by Agha and Palmskog[1])
- Areas of application ranging from *computer science* to *biology*

[1] Agha, Gul, and Karl Palmskog. "A survey of statistical model checking." ACM Transactions on Modeling and Computer Simulation (TOMACS) 28.1 (2018): 1-39.

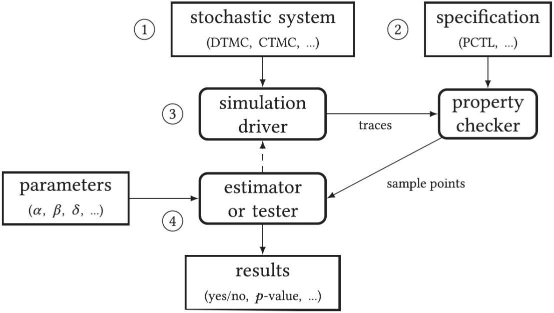
SMC Workflow

SMC Workflow



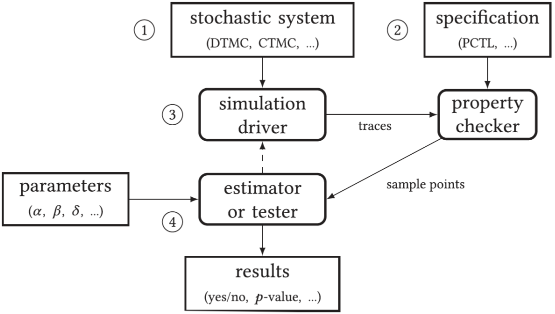
1. Input model of system

SMC Workflow



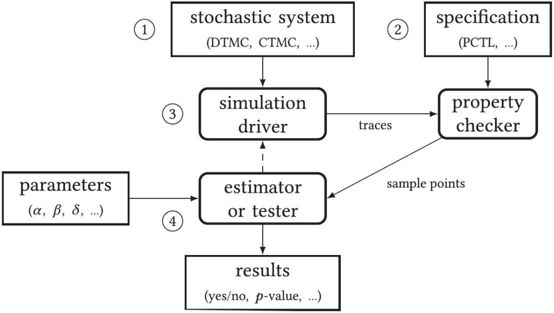
1. Input model of system
2. Formal property specification

SMC Workflow

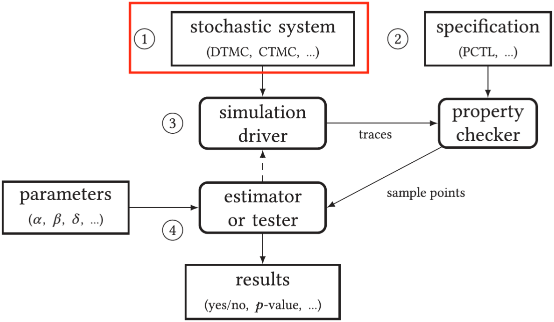


1. Input model of system
2. Formal property specification
3. *Trace*-generator (out-of-scope for this lecture)

SMC Workflow



1. Input model of system
2. Formal property specification
3. *Trace*-generator (out-of-scope for this lecture)
4. Statistical technique



1
stochastic system
(DTMC, CTMC, ...)
2
specification
(PCTL, ...)
3
simulation driver
traces
property checker
sample points
parameters
(α, β, δ, ...)
4
estimator or tester
results
(yes/no, p-value, ...)

SMC is applicable to any process that can be considered a *stochastic discrete event system*: DTMCs (Discrete-Time Markov Chains) (Fig.1) are often used as a reference, but results carry over to other types of systems.

SMC is applicable to any process that can be considered a *stochastic discrete event system*: DTMCs (Discrete-Time Markov Chains) (Fig.1) are often used as a reference, but results carry over to other types of systems.

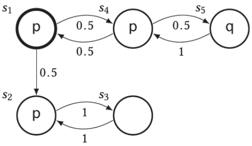


Fig.1. An example of DTMC.

Definition

Let AP be a finite set of atomic propositions. A DTMC is a tuple $\mathcal{M} = (S, s_i, M, L)$ where S is a finite set of states, $s_i \in S$ is the initial state, $M : S \times S \to [0, 1]$ is a transition probability function s.t. for all $s \in S$, $\sum_{s' \in S} M(s, s') = 1$, and $L : S \to 2^{AP}$ is a *labeling function* associating states with their respective true atomic propositions.

SMC is applicable to any process that can be considered a *stochastic discrete event system*: DTMCs (Discrete-Time Markov Chains) (Fig.1) are often used as a reference, but results carry over to other types of systems.

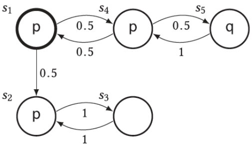


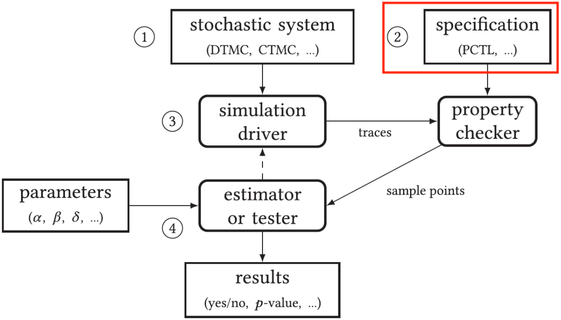
Fig.1. An example of DTMC.

Definition

Let AP be a finite set of atomic propositions. A DTMC is a tuple $\mathcal{M} = (S, s_i, M, L)$ where S is a finite set of states, $s_i \in S$ is the initial state, $M : S \times S \to [0,1]$ is a transition probability function s.t. for all $s \in S$, $\sum_{s' \in S} M(s, s') = 1$, and $L : S \to 2^{AP}$ is a *labeling function* associating states with their respective true atomic propositions.

A *path* is an infinite sequence $s_1, s_2, \dots$ of elements of S, and a *trace* is a finite non-empty prefix of a path.

Stochastic Logics



- **Qualitative** properties of a system are expressed in *temporal logics* (typically Computational Tree Logic)

- If the process is stochastic, we require a logic with which we can express **quantitative** properties about time and probability

- We will see Probabilistic Computational Tree Logics (PCTL)

The semantics of a PCTL formula are defined w.r.t a DTMC M and a state $s \in S$. Syntax can be found in Fig.2.

$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi' \mid \mathrm{P}_{\geq\theta}(\psi)$	$a \in AP$	atomic proposition
$\psi ::= \phi \mid \mathrm{X}\,\phi \mid \phi\,\mathrm{U}^{\leq t}\,\phi' \mid \phi\,\mathrm{U}\,\phi'$	$\theta \in [0,1]$	probability bound
	$t \in \mathbb{Z}^{\geq 0}$	time bound

Fig.2. PCTL Syntax.

The semantics of a PCTL formula are defined w.r.t a DTMC M and a state $s \in S$. Syntax can be found in Fig.2.

$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi' \mid \mathsf{P}_{\geq\theta}(\psi)$	$a \in AP$	atomic proposition
$\psi ::= \phi \mid \mathsf{X}\,\phi \mid \phi\,\mathsf{U}^{\leq t}\,\phi' \mid \phi\,\mathsf{U}\,\phi'$	$\theta \in [0,1]$	probability bound
	$t \in \mathbb{Z}^{\geq 0}$	time bound

Fig.2. PCTL Syntax.

The "unusual" operators are:

- $P_{\geq\theta}(\psi)$: true when the probability that ψ holds on paths starting from s is $\geq \theta$

The semantics of a PCTL formula are defined w.r.t a DTMC M and a state $s \in S$. Syntax can be found in Fig.2.

$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi' \mid \mathrm{P}_{\geq\theta}(\psi)$	$a \in AP$	atomic proposition
$\psi ::= \phi \mid \mathrm{X}\,\phi \mid \phi\,\mathrm{U}^{\leq t}\,\phi' \mid \phi\,\mathrm{U}\,\phi'$	$\theta \in [0,1]$	probability bound
	$t \in \mathbb{Z}^{\geq 0}$	time bound

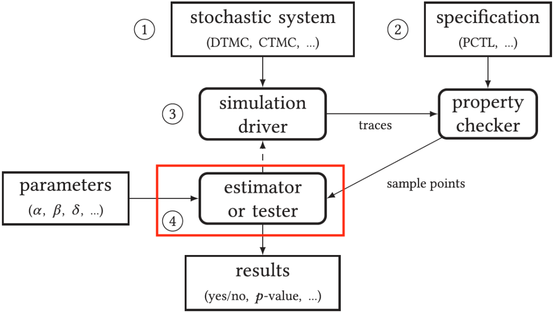
Fig.2. PCTL Syntax.

The "unusual" operators are:

- $P_{\geq\theta}(\psi)$: true when the probability that ψ holds on paths starting from s is $\geq \theta$
- $\phi U^{\leq t}\phi'$: true for a path $\pi = s_1, s_2 ...$ if $\exists s_{k+1}, k+1 \leq t$ in π where ϕ' holds, and ϕ holds in all the states up to s_k

s	s_1	$...$	s_k	s_{k+1}	$\cdots$	s_t
ϕ	ϕ	$\cdots$	ϕ	ϕ'	$\cdots$	$\top$

Fig.3. A path where $\phi U^{\leq t}\phi'$ holds.



1
stochastic system
(DTMC, CTMC, ...)
2
specification
(PCTL, ...)
3
simulation driver
traces
property checker
sample points
parameters
(α, β, δ, ...)
estimator or tester
4
results
(yes/no, p-value, ...)

Given a *stochastic discrete event system* (e.g., a DTMC M) and a property in stochastic logic (e.g., $P_{\geq\theta}(\psi)$), how can we establish if the property holds for some specific case?

Given a *stochastic discrete event system* (e.g., a DTMC M) and a property in stochastic logic (e.g., $P_{\geq\theta}(\psi)$), how can we establish if the property holds for some specific case?

- With SMC, the system is simulated multiple times and the property ψ is checked on every path prefix (trace): a checked trace constitutes a sample point of a Bernoulli variable X, with:

$$X = \begin{cases} 1, & \textit{if } \psi \textit{ holds} \\ 0, & \textit{if } !\psi \textit{ holds} \end{cases} \tag{1}$$

Given a *stochastic discrete event system* (e.g., a DTMC M) and a property in stochastic logic (e.g., $P_{\geq\theta}(\psi)$), how can we establish if the property holds for some specific case?

- With SMC, the system is simulated multiple times and the property ψ is checked on every path prefix (trace): a checked trace constitutes a sample point of a Bernoulli variable X, with:

$$X = \begin{cases} 1, & \textit{if } \psi \textit{ holds} \\ 0, & \textit{if } !\psi \textit{ holds} \end{cases} \tag{1}$$

- Therefore, $P(\psi) = E[X] = p$: the value p can be measured and compared with threshold $\theta \rightarrow$ Hypothesis Testing or Estimation

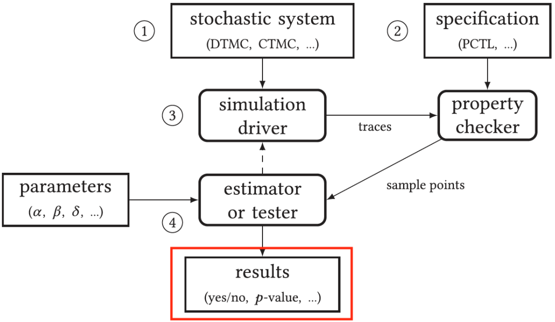
- *Hypothesis Testing*: Given two competing hypotheses $H_0 : p \geq \theta$ and $H_1 : p < \theta$, we evaluate the probability of obtaining such set of samples (the so-called p-value), assuming H_0 was true. If the p-value is smaller than a significance level α, we reject H_0, otherwise we accept it. If we reject H_0 when it is true we make a Type I error (probability α), in the mirrored case we make a Type II error (probability β), as in Fig.4

	Decision	
Truth	*accept* H_0, *reject* H_1	*reject* H_0, *accept* H_1
$p \geq \theta$: H_0 *true*, H_1 *false*	correct ($>1-\alpha$)	type I error ($\leq\alpha$)
$p < \theta$: H_0 *false*, H_1 *true*	type II error ($\leq\beta$)	correct ($>1-\beta$)

The conditions inside parentheses are on the probability for the given outcome.

Fig.4. Schema of errors in Hypothesis Testing.

- *Estimation*: Given a set of observations that represents a random sample, this can be used to estimate the parameters of the distribution, such as the value of p in a Bernoulli process. In this case, the proposed approximation, with m sample points, is $p' = \frac{\sum_{i=1}^{m} x_i}{m}$. Given a precision ϵ, p is guaranteed to belong to $[p' - \epsilon, p' + \epsilon]$.



- $P(\psi) = p \geq \theta$?
- Hypothesis Testing: accept $H_0 \rightarrow p \geq \theta$, reject $H_0 \rightarrow p < \theta$
- Estimation: approximate value of p and compare it with threshold θ

Uppaal SMC

Uppaal SMC

- Uppaal developers have recently (2012) released an Uppaal extension that allows us to model systems with stochastic features and run SMC experiments;
- Reading the Uppaal SMC Tutorial[2] is highly suggested if you choose to pursue the stochastic path for the project.
- If you do, make sure you download the latest build (**v.4.1**) that includes the SMC extension.

[2] https://link.springer.com/content/pdf/10.1007/s10009-014-0361-y.pdf

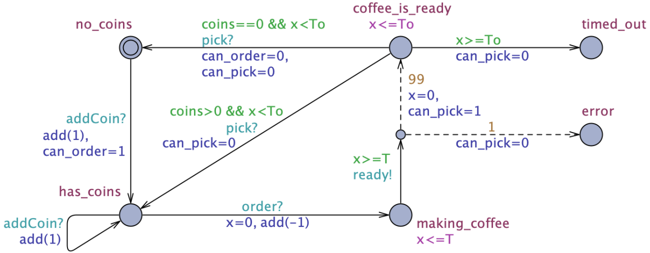
- Uppaal SMC extends the formalism underlying the tool to Stochastic Hybrid Automata (SHA):
  - *stochastic* features replace plain non-determinism with probabilistic choices;
  - *hybrid* systems are an extension of Timed Automata where clock rates can be given as general expressions (i.e., a differential equation)[3]

[3] No time to look into hybrid systems today (and they are not required by the project) but, if you are interested, we got theses!

Case Study: Stochastic Coffee Machine

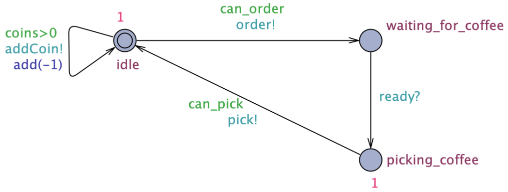
- We want to apply the following changes to previous coffee machine model:
  - When time T required to brew coffee elapses, with a 1% probability the machine shuts down and enters a deadlock *error* state;
  - When the user is idle, the probability of them performing an action is distributed according to an *exponential* distribution with fixed rate $\lambda = 1$.
  - When the coffee is ready, the probability of the user picking up the coffee is distributed according to an *exponential* distribution with fixed rate $\lambda = 1$.

- Stochastic Coffee Machine:



Case Study: Stochastic Coffee Machine

- Stochastic User:



- The model will be uploaded on Beep.

Uppaal SMC: Verifier

- Uppaal queries are also extended to include PCTL operators:
  - $\texttt{Pr}[<= \texttt{TAU}; \texttt{N}](\psi)$ is equivalent to PCTL formula $P^{\leq\tau}(\psi)$ and yields a probability range for formula ψ holding within time-bound τ. Optional parameter $\texttt{N}$ bounds the number of runs generated for the SMC experiment (decreasing its value might save you some time, but it may also reduce the degree of confidence!)

Uppaal SMC: Verifier

- Uppaal queries are also extended to include PCTL operators:
  - $\texttt{Pr}[<= \texttt{TAU}; \texttt{N}](\psi)$ is equivalent to PCTL formula $P^{\leq\tau}(\psi)$ and yields a probability range for formula ψ holding within time-bound τ. Optional parameter `N` bounds the number of runs generated for the SMC experiment (decreasing its value might save you some time, but it may also reduce the degree of confidence!)
  - $\texttt{simulate}[<= \texttt{TAU}; \texttt{N}]\{\texttt{E}_1, \ldots, \texttt{E}_\texttt{m}\}$ generates N runs of the system (that you can plot or export), each `TAU` time instants long, where $\texttt{E}_1, \ldots, \texttt{E}_\texttt{m}$ are the monitored expressions (this is a precious tool to test your system!).

Case Study: Stochastic Coffee Machine

- Let us run some queries on the updated coffee machine model:
  - `Pr[<= TAU](<> u.waiting_for_coffee)` yields a probability range for the user *eventually* ordering coffee;
  - `Pr[<= TAU](<> m.coffee_is_ready)` yields a probability range for the machine *eventually* producing a cup of coffee (play around to see how this changes in relation to the probability that the user will actually order coffee);
  - `Pr[<= TAU](<> m.error)` yields a probability range for the coffee machine *eventually* shutting down;

- Let us run some queries on the updated coffee machine model:
  - $\texttt{Pr}[<= \texttt{TAU}](<> \texttt{u.waiting\_for\_coffee})$ yields a probability range for the user *eventually* ordering coffee;
  - $\texttt{Pr}[<= \texttt{TAU}](<> \texttt{m.coffee\_is\_ready})$ yields a probability range for the machine *eventually* producing a cup of coffee (play around to see how this changes in relation to the probability that the user will actually order coffee);
  - $\texttt{Pr}[<= \texttt{TAU}](<> \texttt{m.error})$ yields a probability range for the coffee machine *eventually* shutting down;

  - $\texttt{simulate}[<= \texttt{TAU}; 1]\{\texttt{u.idle} * 2, \texttt{u.waiting\_for\_coffee} * 2, \texttt{u.picking\_coffee} * 2, \texttt{m.x}/\texttt{m.T}, \texttt{m.making\_coffee}, \texttt{m.coffee\_is\_ready}\}$ generates a trace monitoring user/machine states (the $*2$ is purely for visualization purposes).