Steady State Property Verification: a Comparison Study

Diana EL RABIH [1], Gael Gorgo [2], Nihal PEKERGIN [1], Jean-Marc Vincent [2]

[1] LACL, University of Paris Est (Paris 12)
[2] LIG, University of Grenoble (Joseph Fourrier)
This work is supported by Checkbound, ANR-06-SETI-002

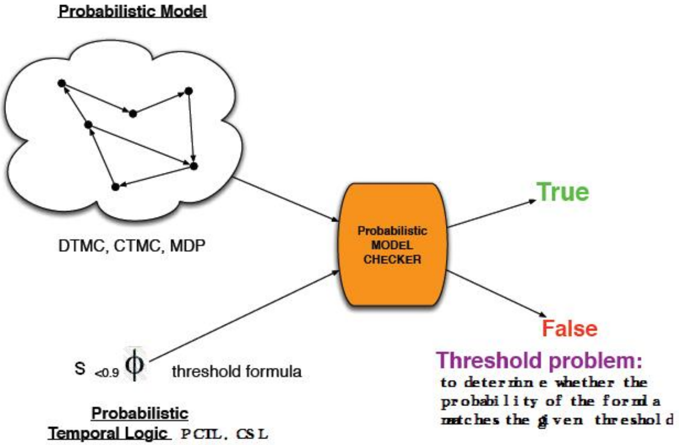
VECOS, Paris, 2010

Outline

Outline

1. Introduction
   - Probabilistic Model Checking
   - Perfect Sampling
2. SMC using Perfect Sampling
   - SMC Decision Method
   - SMC of CSL Steady State Formula
3. Experimental Comparison Study
   - Case studies
   - Compared Tools
   - Results and discussions
4. Conclusion and Future works

Probabilistic Model Checking



Probabilistic Model Checking

1. Probabilistic Models
   - CTMC, DTMC, MDP, ...
   - Queueing Networks, Network protocols, Distributed Systems
2. Dependability, availability and reachability properties with probabilistic temporal logics
   - CSL for CTMC, PCTL for DTMC
   - Steady State Operator: $\mathcal{S}_{\geq\theta}(\phi)$

     Ex: With probability at least θ, a system will be available at long run (in steady-state)

Probabilistic Model Checking

1. Probabilistic Models
   - CTMC, DTMC, MDP, ...
   - Queueing Networks, Network protocols, Distributed Systems
2. Dependability, availability and reachability properties with probabilistic temporal logics
   - CSL for CTMC, PCTL for DTMC
   - Steady State Operator: $S_{\geq\theta}(\phi)$

     Ex: With probability at least θ, a system will be available at long run (in steady-state)

Probabilistic Model Checking

1. Probabilistic Models
   - CTMC, DTMC, MDP, ...
   - Queueing Networks, Network protocols, Distributed Systems
2. Dependability, availability and reachability properties with probabilistic temporal logics
   - CSL for CTMC, PCTL for DTMC
   - Steady State Operator: $\mathcal{S}_{\geq\theta}(\phi)$

     Ex: With probability at least θ, a system will be available at long run (in steady-state)

Solution Methods

1. Numerical Model Checking (NMC)
   - Based on: Computation of distributions
   - \+ Highly accurate results
   - \- Intractable for systems with very large state space
2. Statistical Model Checking (SMC)
   - Based on: Sampling (by simulation or by measurement) and Statistical Methods for verification
   - \+ Low memory requirements
   - \- Expensive if high accuracy is required

Solution Methods

1. Numerical Model Checking (NMC)
   - Based on: Computation of distributions
   - \+ Highly accurate results
   - \- Intractable for systems with very large state space
2. Statistical Model Checking (SMC)
   - Based on: Sampling (by simulation or by measurement) and Statistical Methods for verification
   - \+ Low memory requirements
   - \- Expensive if high accuracy is required

Solution Methods

1. Numerical Model Checking (NMC)
   - Based on: Computation of distributions
   - \+ Highly accurate results
   - \- Intractable for systems with very large state space
2. Statistical Model Checking (SMC)
   - Based on: Sampling (by simulation or by measurement) and Statistical Methods for verification
   - \+ Low memory requirements
   - \- Expensive if high accuracy is required

Solution Methods

1. Numerical Model Checking (NMC)
   - Based on: Computation of distributions
   - \+ Highly accurate results
   - \- Intractable for systems with very large state space
2. Statistical Model Checking (SMC)
   - Based on: Sampling (by simulation or by measurement) and Statistical Methods for verification
   - \+ Low memory requirements
   - \- Expensive if high accuracy is required

Solution Methods

1. Numerical Model Checking (NMC)
   - Based on: Computation of distributions
   - \+ Highly accurate results
   - \- Intractable for systems with very large state space
2. Statistical Model Checking (SMC)
   - Based on: Sampling (by simulation or by measurement) and Statistical Methods for verification
   - \+ Low memory requirements
   - \- Expensive if high accuracy is required

Solution Methods

1. Numerical Model Checking (NMC)
   - Based on: Computation of distributions
   - \+ Highly accurate results
   - \- Intractable for systems with very large state space
2. Statistical Model Checking (SMC)
   - Based on: Sampling (by simulation or by measurement) and Statistical Methods for verification
   - \+ Low memory requirements
   - \- Expensive if high accuracy is required

Existing Tools

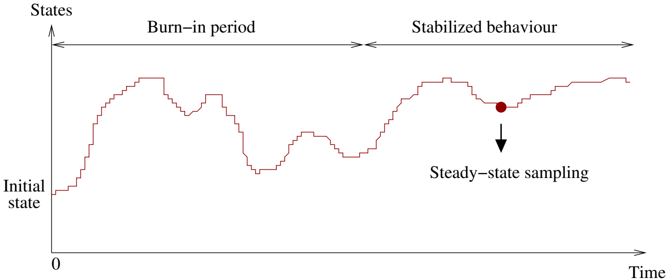
- PRISM tool: Numerical (memory limit)
- MRMC tool: Statistical (simulation by regeneration method, same memory limit problem as PRISM)
- Ymer, VESTA tools: Statistical (*transient properties*)
- APMC tool: Statistical (*transient properties*)

Existing Tools

- PRISM tool: Numerical (memory limit)
- MRMC tool: Statistical (simulation by regeneration method, same memory limit problem as PRISM)
- Ymer, VESTA tools: Statistical (*transient properties*)
- APMC tool: Statistical (*transient properties*)

Outline

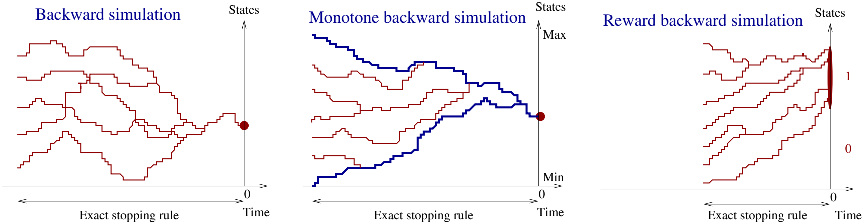
Stochastic simulation idea



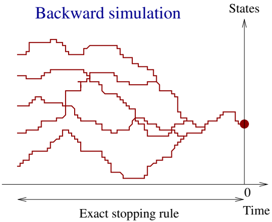
- Drawbacks of forward simulation
  - Dependence on the initial state
  - Burn-in period estimation

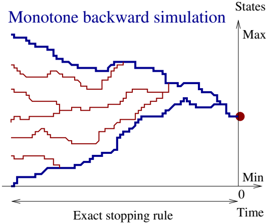
    $\Rightarrow$ Biased sampling
- Alternatives
  - Regeneration (MRMC tool)
  - Perfect sampling (Ψ^2 tool)

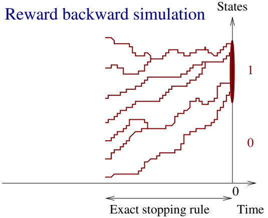
Backward Simulation Schemes

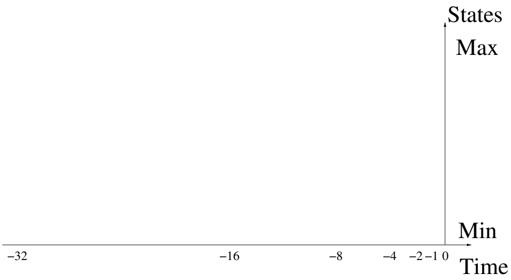


Backward Simulation Schemes

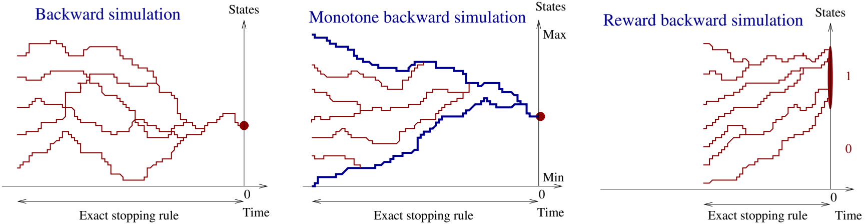


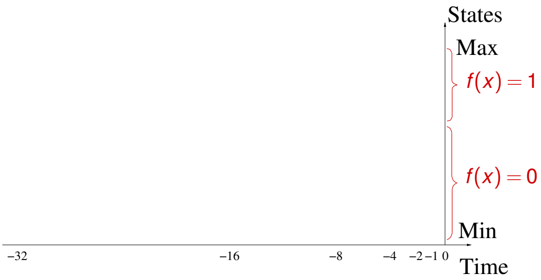




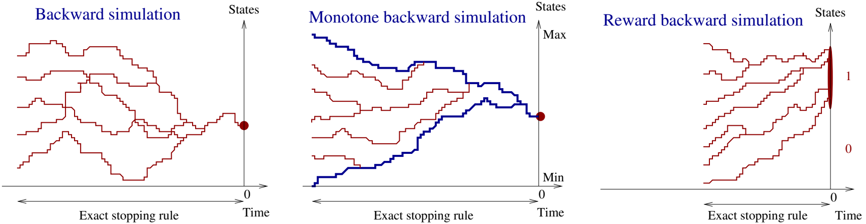


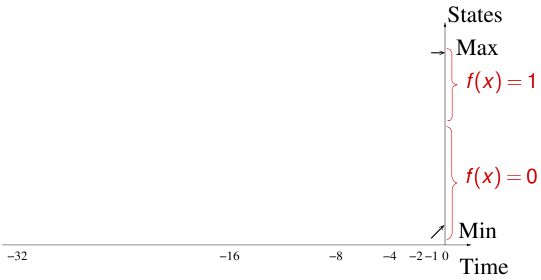
Backward Simulation Schemes



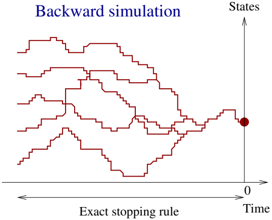


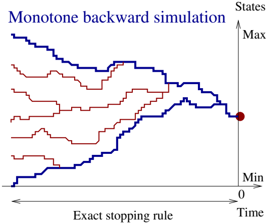
Backward Simulation Schemes

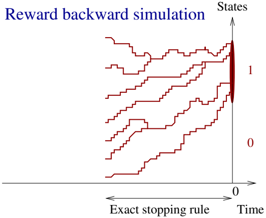


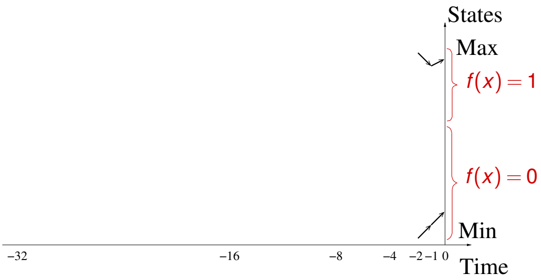


Backward Simulation Schemes

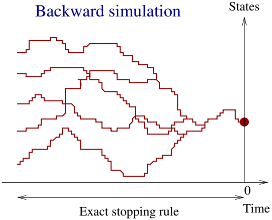


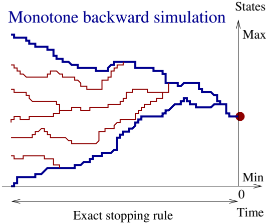


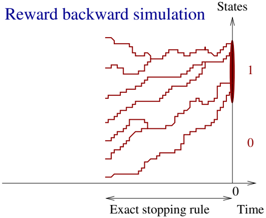


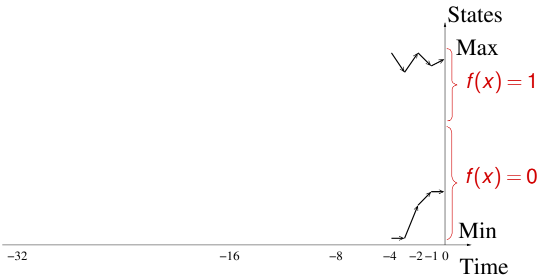


Backward Simulation Schemes

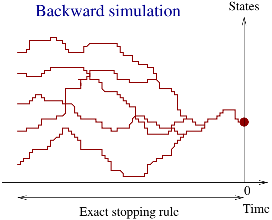


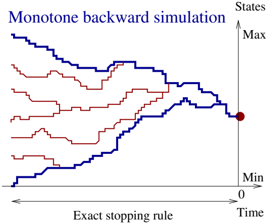


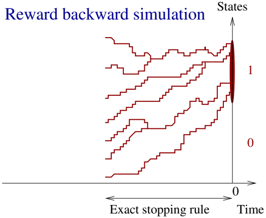


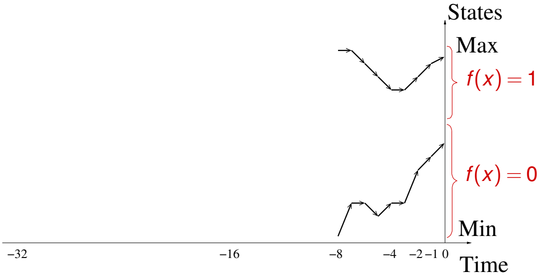


Backward Simulation Schemes

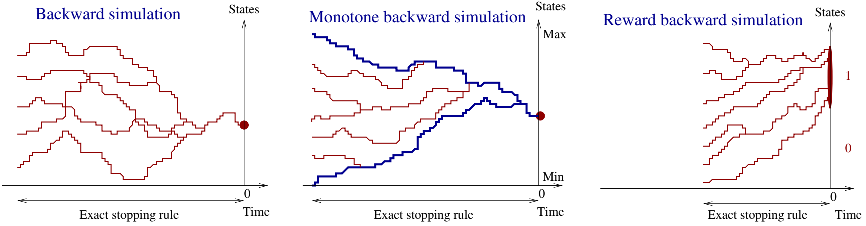


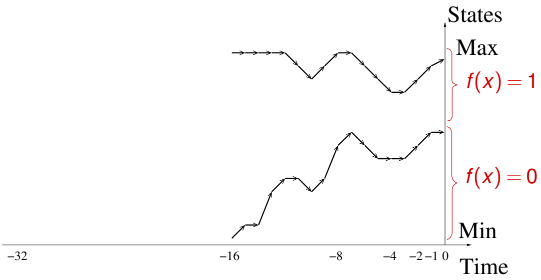




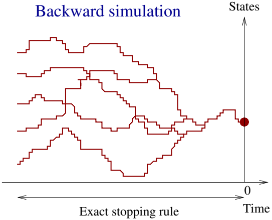


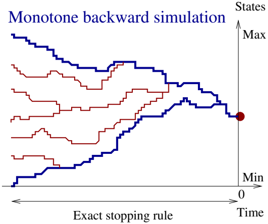
Backward Simulation Schemes

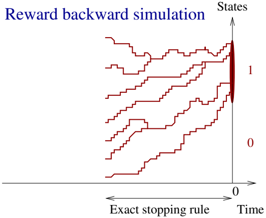


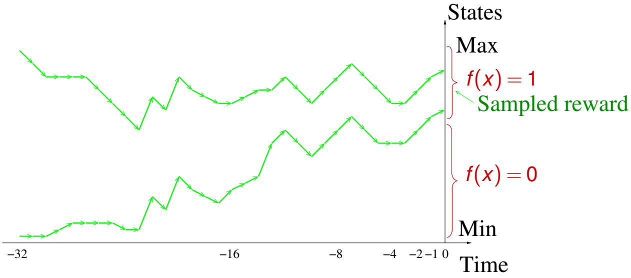


Backward Simulation Schemes

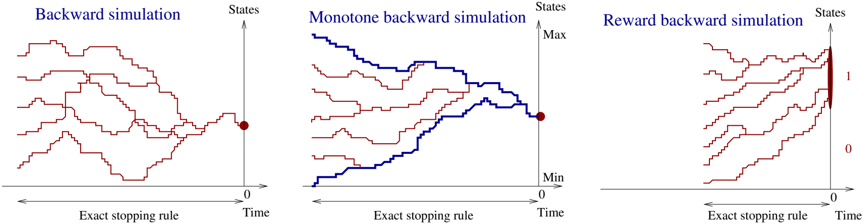


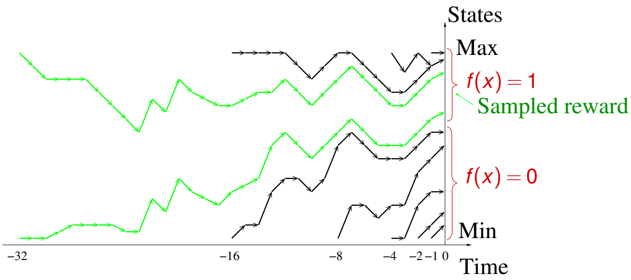






Backward Simulation Schemes





Synthesis

- Advantages
  - Unbiased sampling of the steady-state
  - Very efficient under monotonicity
  - Very efficient for rare probability estimation (reward sampling)
- Drawbacks
  - To study the monotonicity of a system can be complex
    - If it is monotone, that has to be proven
    - If not, we may involve "non-monotone techniques" (Ex: extended sandwiching approach, called envelopes)
- A perfect sampler ψ^2 proposed in MESCAL Project
  - Samples rewards of the stationary distribution of large Markov chains

Synthesis

- Advantages
  - Unbiased sampling of the steady-state
  - Very efficient under monotonicity
  - Very efficient for rare probability estimation (reward sampling)
- Drawbacks
  - To study the monotonicity of a system can be complex
    - If it is monotone, that has to be proven
    - If not, we may involve "non-monotone techniques" (Ex: extended sandwiching approach, called envelopes)
- A perfect sampler ψ^2 proposed in MESCAL Project
  - Samples rewards of the stationary distribution of large Markov chains

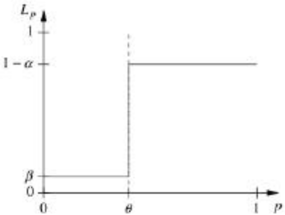
Synthesis

- Advantages
  - Unbiased sampling of the steady-state
  - Very efficient under monotonicity
  - Very efficient for rare probability estimation (reward sampling)
- Drawbacks
  - To study the monotonicity of a system can be complex
    - If it is monotone, that has to be proven
    - If not, we may involve "non-monotone techniques" (Ex: extended sandwiching approach, called envelopes)
- A perfect sampler ψ^2 proposed in MESCAL Project
  - Samples rewards of the stationary distribution of large Markov chains

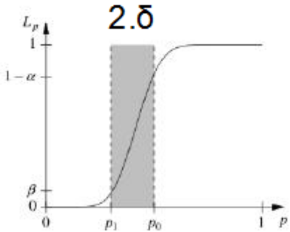
Outline

Statistical Hypothesis Testing (SHT)

- Estimate the probability p that φ of a given formula $S_{<\theta}(\varphi)$ is satisfied on sample paths
- Formula verification: Test H : p $\geq \theta$ against K : p < θ
- For specified indifference region δ and error bounds (α,β)



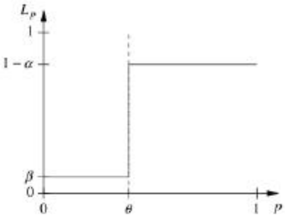
(a) Prob. of accepting H (ideal)



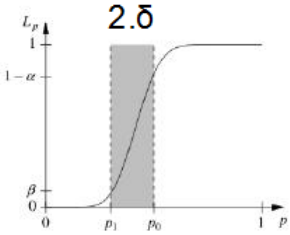
(b) Prob. of accepting H (with indifference region)

Statistical Hypothesis Testing (SHT)

- Estimate the probability p that φ of a given formula $S_{<\theta}(\varphi)$ is satisfied on sample paths
- Formula verification: Test H : p $\geq \theta$ against K : p < θ
- For specified indifference region δ and error bounds (α,β)



(a) Prob. of accepting H (ideal)



(b) Prob. of accepting H (with indifference region)

Decision Method

1. Inspired from the Single Sampling Plan (SHT method used by Younes et al.)
2. Check samples and compute number of positive samples (Y)

   $H_0 : p \geq \theta + \delta$ $\qquad$ $H_1 : p < \theta - \delta$

   - If $Y \geq m$ then accepting H_0 (YES)
   - Else If $Y < m$ then accepting H_1 (NO)
   - where m is the acceptance threshold of the statistical test
3. Statistical test strength (n, m) depends on (α, β) and on δ where n is the total sample size

Decision Method

1. Inspired from the Single Sampling Plan (SHT method used by Younes et al.)
2. Check samples and compute number of positive samples (Y)

   $H_0 : p \geq \theta + \delta \qquad H_1 : p < \theta - \delta$

   - If $Y \geq m$ then accepting H_0 (YES)
   - Else If $Y < m$ then accepting H_1 (NO)
   - where m is the acceptance threshold of the statistical test
3. Statistical test strength (n, m) depends on (α, β) and on δ where n is the total sample size

Decision Method

1. Inspired from the Single Sampling Plan (SHT method used by Younes et al.)
2. Check samples and compute number of positive samples (Y)

   $H_0 : p \geq \theta + \delta \qquad H_1 : p < \theta - \delta$

   - If $Y \geq m$ then accepting H_0 (YES)
   - Else If $Y < m$ then accepting H_1 (NO)
   - where m is the acceptance threshold of the statistical test
3. Statistical test strength (n, m) depends on (α, β) and on δ where n is the total sample size

Decision Method

1. Inspired from the Single Sampling Plan (SHT method used by Younes et al.)
2. Check samples and compute number of positive samples (Y)

   $H_0 : p \geq \theta + \delta \qquad H_1 : p < \theta - \delta$

   - If $Y \geq m$ then accepting H_0 (YES)
   - Else If $Y < m$ then accepting H_1 (NO)
   - where m is the acceptance threshold of the statistical test
3. Statistical test strength (n, m) depends on (α, β) and on δ where n is the total sample size

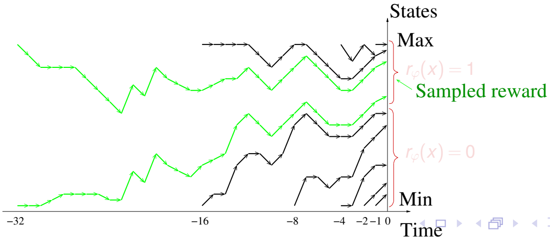
Outline

Verification of CSL Steady State Formula

SMC of ψ=$\mathcal{S}_{\geq\theta}(\varphi)$ by using functional and/or monotone perfect simulation

- Check if the obtained steady-state samples (x) satisfies φ or not
- By associating reward $r_{\varphi}(x)$ to each state x for the given property φ:

$$\begin{aligned} r_{\varphi}(x) &= 1, \text{ if } x \models \varphi \\ r_{\varphi}(x) &= 0, \text{ otherwise } x \not\models \varphi \end{aligned} \quad (1)$$

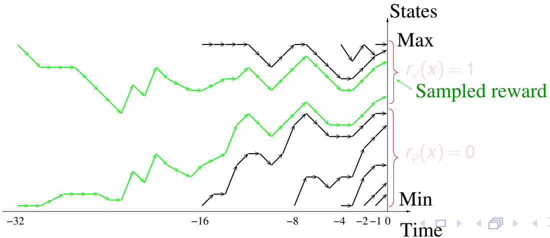


Verification of CSL Steady State Formula

SMC of $\psi = \mathcal{S}_{\geq\theta}(\varphi)$ by using functional and/or monotone perfect simulation

- Check if the obtained steady-state samples (x) satisfies φ or not
- By associating reward $r_\varphi(x)$ to each state x for the given property φ:

$$\begin{aligned} r_\varphi(x) &= 1, \text{ if } x \models \varphi \\ r_\varphi(x) &= 0, \text{ otherwise } x \not\models \varphi \end{aligned} \quad (1)$$

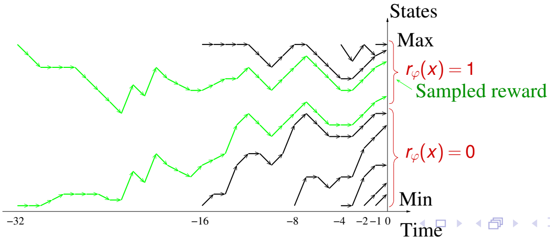


Verification of CSL Steady State Formula

SMC of ψ=$\mathcal{S}_{\geq\theta}(\varphi)$ by using functional and/or monotone perfect simulation

- Check if the obtained steady-state samples (x) satisfies φ or not
- By associating reward $r_\varphi(x)$ to each state x for the given property φ:

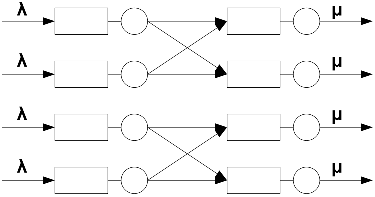
$$
\begin{aligned}
r_\varphi(x) &= 1, \text{ if } x \models \varphi \\
r_\varphi(x) &= 0, \text{ otherwise } x \not\models \varphi
\end{aligned}
\tag{1}
$$



Outline

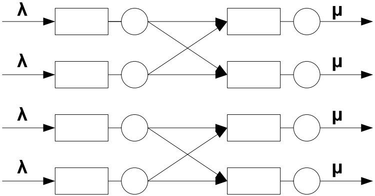
Models

1. Tandem network with 4 queues (TN)
   - Monotone model (ψ^2 benchmark)
2. Multistage delta queueing network with 8 queues (MDN)
   - Monotone model (ψ^2 benchmark)

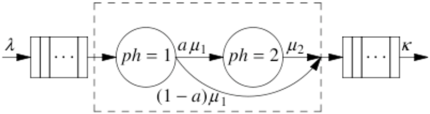


Models

1. Tandem network with 4 queues (TN)
   - Monotone model (ψ^2 benchmark)
2. Multistage delta queueing network with 8 queues (MDN)
   - Monotone model (ψ^2 benchmark)



- Tandem Queuing Network with coaxian server (TQN-Cox)
  - Non monotone model (PRISM benchmark)
  - Implemented in ψ^2 using envelopes
  - Extended sandwiching approach (envelopes) are very efficient for this example



Verified Properties (1)

1. AP $a_i(k)$: True if $N_i > k$, False otherwise
   - N_i: number of customers in the i^{th} queue
   - $0 \leq k \leq N_{max}$ and N_{max}: maximum queue size
2. Define different saturation and availability measures for the underlying models
   - Ex: Saturation property in the i^{th} buffer, $S_{<\theta}(a_i(N_{max}))$, also check availability property $S_{\geq 1-\theta}(\neg\, a_i\, (N_{max}))$

Verified Properties (1)

1. AP $a_i(k)$: True if $N_i > k$, False otherwise
   - N_i: number of customers in the i^{th} queue
   - $0 \leq k \leq N_{max}$ and N_{max}: maximum queue size
2. Define different saturation and availability measures for the underlying models
   - Ex: Saturation property in the i^{th} buffer, $S_{<\theta}(a_i(N_{max})$, also check availability property $S_{\geq 1-\theta}(\neg\ a_i\ (N_{max}))$

Verified Properties (2)

1. Tandem network with 4 queues (TN)
   - 4^{th} buffer is full
2. Multistage delta queueing network with 8 queues (MDN)
   - At least one queue of the second stage of MDN is full
3. Tandem Queuing Network with coaxian server (TQN-Cox)
   - The overall system is full

Outline

1. PRISM tool (Numerical)
   - Computes probabilities for each reachable state
   - Solves system of linear equations to find probabilities with convergence precision ϵ
2. ψ^2 with SHT tool (Statistical)
   - Perfect sampling (Functional)
   - Verification by Statistical Hypothesis Testing with (α, β, δ) precision parameters
3. Comparison study
   - For fair comparison we take $\epsilon = 2.\delta$
   - (ϵ, δ)={$(10^{-3}/2, 10^{-3}/4), (10^{-4}, 10^{-4}/2)$} and $\alpha = \beta = 10^{-2}$
   - PRISM: memory is proportional to the number of states
   - ψ^2 with SHT: memory is never exhausted

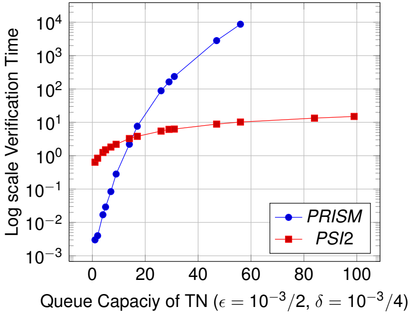
1. PRISM tool (Numerical)
   - Computes probabilities for each reachable state
   - Solves system of linear equations to find probabilities with convergence precision ϵ
2. ψ^2 with SHT tool (Statistical)
   - Perfect sampling (Functional)
   - Verification by Statistical Hypothesis Testing with (α, β, δ) precision parameters
3. Comparison study
   - For fair comparison we take $\epsilon = 2.\delta$
   - (ϵ, δ)={$(10^{-3}/2, 10^{-3}/4), (10^{-4}, 10^{-4}/2)$} and $\alpha = \beta = 10^{-2}$
   - PRISM: memory is proportional to the number of states
   - ψ^2 with SHT: memory is never exhausted

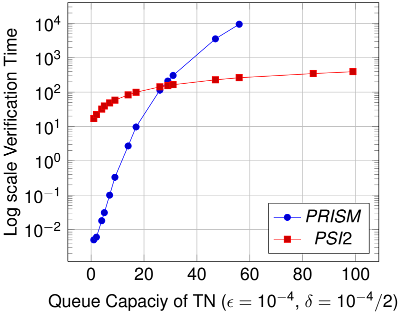
1. PRISM tool (Numerical)
   - Computes probabilities for each reachable state
   - Solves system of linear equations to find probabilities with convergence precision ϵ
2. ψ^2 with SHT tool (Statistical)
   - Perfect sampling (Functional)
   - Verification by Statistical Hypothesis Testing with (α, β, δ) precision parameters
3. Comparison study
   - For fair comparison we take $\epsilon = 2.\delta$
   - (ϵ, δ)={$(10^{-3}/2, 10^{-3}/4), (10^{-4}, 10^{-4}/2)$} and $\alpha = \beta = 10^{-2}$
   - PRISM: memory is proportional to the number of states
   - ψ^2 with SHT: memory is never exhausted

Outline

Tandem Network (TN)

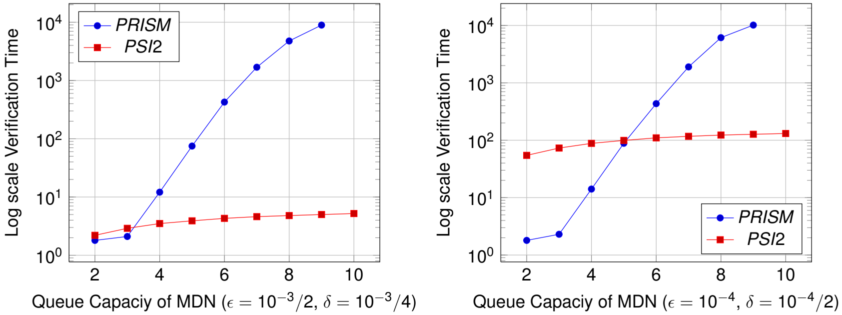
- Model and property: λ =0.9, $\mu_i = 1,\ 1 \leq i \leq 4$, $S_{<\theta}$ (*last-full*) where $\theta = 0.001$





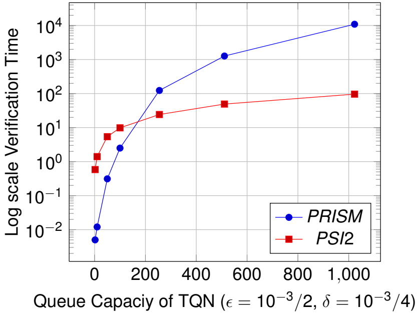
Multistage Delta Network (MDN)

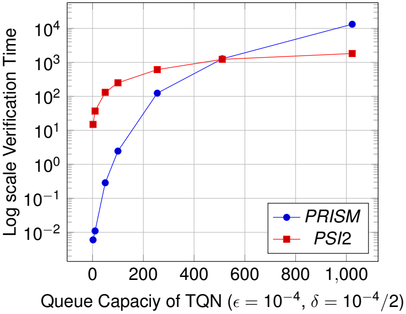
- Model and property: 2 *stages* and 4 *buffers*/*stage*, $\lambda = 0.9$, $\mu = 1$, $(\tau_{rout1}, \tau_{rout2}) = (0.8, 0.6)$, $S_{<\theta}$ (*last-stage-full*) where $\theta = 0.001$



Tandem Queueing Network (TQN)

- Model and property: $\lambda = 4 \times N_{max}$, $\mu_1 = 2$, $\mu_2 = 2$, $a = 0.1$ and κ =4, $S_{<\theta}$ (*sys-full*) where $\theta = 0.001$





Discussions

1. Variation of precision parameters ϵ (numerical) and δ (statistical)
   - Numerical verification time dependence on ϵ is negligible
   - Statistical verification time dependence on δ is considerable
2. Dependence on state space size is negligible in ψ^2 (functional)
3. Memory limit:
   - TN case: For $N_{max} = 99$ ($|X| = 10^8$)
   - MDN case: For $N_{max} = 10$ ($|X| = 1.1 * 10^8$)
   - TQN case: For $N_{max} = 7500$ ($|X| = 2.1 * 10^8$)
4. MDN case: For 4 *stages* and 8 *buffers*/*stage*
   - Efficient results using Ψ^2 while not possible using PRISM (memory problem for N_{max}=1, $O((N_{max} + 1)^{32})$)

Discussions

1. Variation of precision parameters ϵ (numerical) and δ (statistical)
   - Numerical verification time dependence on ϵ is negligible
   - Statistical verification time dependence on δ is considerable
2. Dependence on state space size is negligible in ψ^2 (functional)
3. Memory limit:
   - TN case: For $N_{max} = 99$ ($|X| = 10^8$)
   - MDN case: For $N_{max} = 10$ ($|X| = 1.1 * 10^8$)
   - TQN case: For $N_{max} = 7500$ ($|X| = 2.1 * 10^8$)
4. MDN case: For 4 *stages* and 8 *buffers*/*stage*
   - Efficient results using Ψ^2 while not possible using PRISM (memory problem for N_{max}=1, $O((N_{max} + 1)^{32})$)

Discussions

1. Variation of precision parameters ϵ (numerical) and δ (statistical)
   - Numerical verification time dependence on ϵ is negligible
   - Statistical verification time dependence on δ is considerable
2. Dependence on state space size is negligible in ψ^2 (functional)
3. Memory limit:
   - TN case: For $N_{max} = 99$ ($|X| = 10^8$)
   - MDN case: For $N_{max} = 10$ ($|X| = 1.1 * 10^8$)
   - TQN case: For $N_{max} = 7500$ ($|X| = 2.1 * 10^8$)
4. MDN case: For 4 *stages* and 8 *buffers*/*stage*
   - Efficient results using Ψ^2 while not possible using PRISM (memory problem for N_{max}=1, $O((N_{max} + 1)^{32})$)

Conclusion

1. Empirical comparison of numerical and statistical solutions
   - PRISM vs. ψ^2 with SHT
   - Focus on CSL steady state formulas
2. We have found that:
   - ψ^2 with SHT scales better with the state space size (no limiting memory problem)
   - ψ^2 with SHT is faster than PRISM for large models (greater than 10^5)
   - Memory problem: Limiting state space sizes using PRISM for the considered case studies

Conclusion

1. Empirical comparison of numerical and statistical solutions
   - PRISM vs. ψ^2 with SHT
   - Focus on CSL steady state formulas
2. We have found that:
   - ψ^2 with SHT scales better with the state space size (no limiting memory problem)
   - ψ^2 with SHT is faster than PRISM for large models (greater than 10^5)
   - Memory problem: Limiting state space sizes using PRISM for the considered case studies

Conclusion

1. Empirical comparison of numerical and statistical solutions
   - PRISM vs. ψ^2 with SHT
   - Focus on CSL steady state formulas
2. We have found that:
   - ψ^2 with SHT scales better with the state space size (no limiting memory problem)
   - ψ^2 with SHT is faster than PRISM for large models (greater than 10^5)
   - Memory problem: Limiting state space sizes using PRISM for the considered case studies

Conclusion

1. Empirical comparison of numerical and statistical solutions
   - PRISM vs. ψ^2 with SHT
   - Focus on CSL steady state formulas
2. We have found that:
   - ψ^2 with SHT scales better with the state space size (no limiting memory problem)
   - ψ^2 with SHT is faster than PRISM for large models (greater than 10^5)
   - Memory problem: Limiting state space sizes using PRISM for the considered case studies

Conclusion

1. Empirical comparison of numerical and statistical solutions
   - PRISM vs. ψ^2 with SHT
   - Focus on CSL steady state formulas
2. We have found that:
   - ψ^2 with SHT scales better with the state space size (no limiting memory problem)
   - ψ^2 with SHT is faster than PRISM for large models (greater than 10^5)
   - Memory problem: Limiting state space sizes using PRISM for the considered case studies

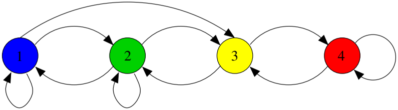
Future works

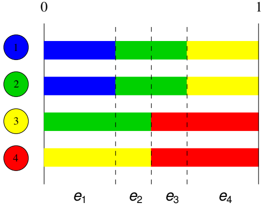
1. Compare ψ^2 with SHT tool with the MRMC tool
2. SMC of CSL time unbounded until formulas

Future works

1. Compare ψ^2 with SHT tool with the MRMC tool
2. SMC of CSL time unbounded until formulas

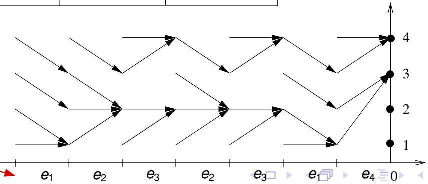
Event modelling of a Markov chain





event	e_1	e_2	e_3	e_4
probabililty	$\frac{2}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{2}{6}$
Transition function $\Phi(x, .)$	$4 \to 3$, $3 \to 2$, $2 \to 1$, $1 \to 1$	$4 \to 3$, $3 \to 2$, $2 \to 2$, $1 \to 2$	$4 \to 4$, $3 \to 4$, $2 \to 2$, $1 \to 2$	$4 \to 4$, $3 \to 4$, $2 \to 3$, $1 \to 3$

Sample paths are driven by
the same source of randomness
(inovation process of events)



Monotonicity

Monotone event

- let $\preceq$ be a partial order on a multi-dimensional state space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_K$ (usually a lattice).

$$x \preceq y \Leftrightarrow x^i \leq y^i \ \ \forall i$$

- An event e is monotone if it preserves the partial ordering $\preceq$ on $\mathcal{X}$

$$\forall (x, y) \in \mathcal{X} \quad x \preceq y \ \Rightarrow \ \Phi(x, e) \preceq \Phi(y, e)$$

Monotonicity of systems

A Markov chain is monotone if all events are monotone