Steady State Dependability Verification by Perfect Sampling

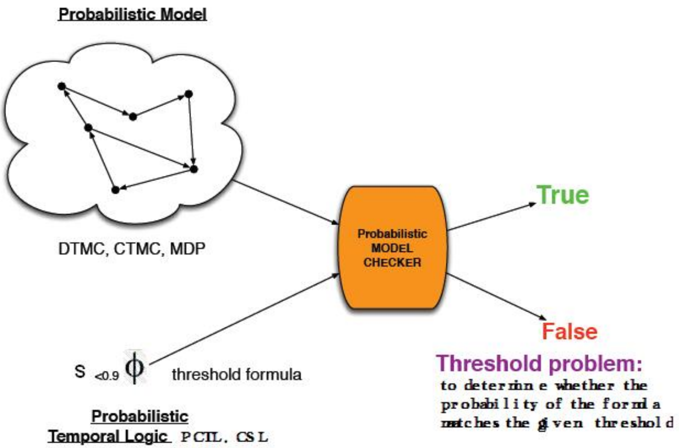
Diana EL RABIH [(1)], Gael Gorgo [(2)], Nihal PEKERGIN [(1)], Jean-Marc Vincent [(2)]

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

Outline

Outline

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: $\mathcal{S}_{\geq\theta}(\phi)$

     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: $\mathcal{S}_{\geq\theta}(\phi)$

     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 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 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 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 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 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 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 Model Checkers

1. PRISM tool: Numerical
   - Matrix representation: memory limit
2. MRMC tool: Statistical
   - Simulation by regeneration method
   - Same memory limit problem as PRISM
3. Ymer, VESTA tools: Statistical
   - *transient properties*
4. APMC tool: Statistical
   - *transient properties*
   - integrated in PRISM

Existing Model Checkers

1. PRISM tool: Numerical
   - Matrix representation: memory limit
2. MRMC tool: Statistical
   - Simulation by regeneration method
   - Same memory limit problem as PRISM
3. Ymer, VESTA tools: Statistical
   - *transient properties*
4. APMC tool: Statistical
   - *transient properties*
   - integrated in PRISM

Existing Model Checkers

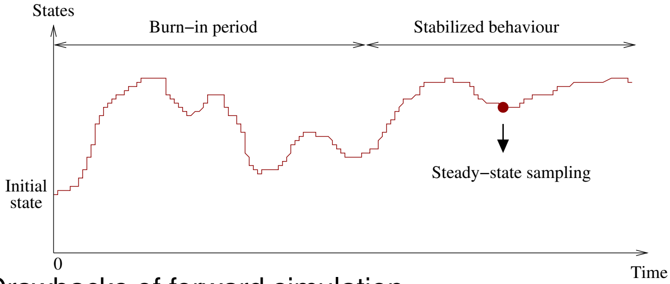
1. PRISM tool: Numerical
   - Matrix representation: memory limit
2. MRMC tool: Statistical
   - Simulation by regeneration method
   - Same memory limit problem as PRISM
3. Ymer, VESTA tools: Statistical
   - *transient properties*
4. APMC tool: Statistical
   - *transient properties*
   - integrated in PRISM

Existing Model Checkers

1. PRISM tool: Numerical
   - Matrix representation: memory limit
2. MRMC tool: Statistical
   - Simulation by regeneration method
   - Same memory limit problem as PRISM
3. Ymer, VESTA tools: Statistical
   - *transient properties*
4. APMC tool: Statistical
   - *transient properties*
   - integrated in PRISM

Outline

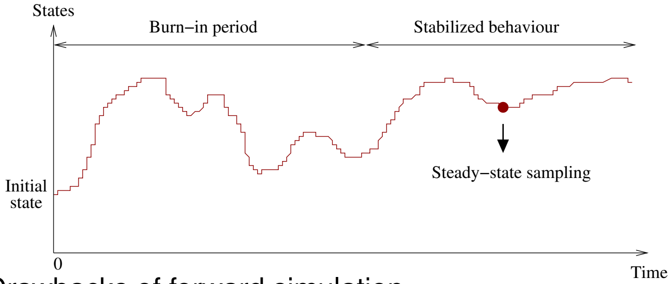
Stochastic simulation idea



- Drawbacks of forward simulation
  - Steady state is not exact
  - Dependence on the initial state
  - Burn-in period estimation

    ⇒ Biased sampling
- Alternatives
  - Regeneration (MRMC tool)
  - Perfect sampling (Ψ^2 tool)

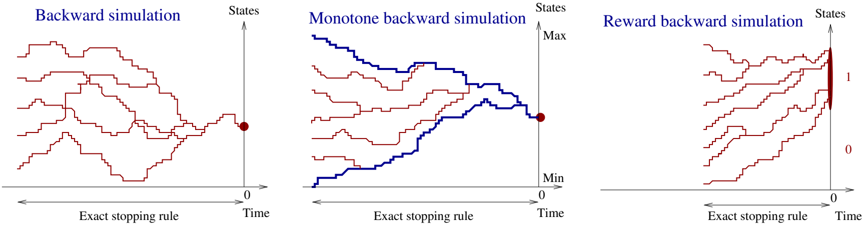
Stochastic simulation idea

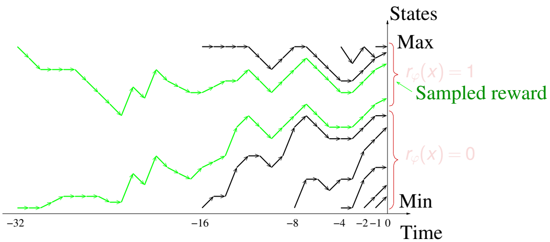


- Drawbacks of forward simulation
  - Steady state is not exact
  - Dependence on the initial state
  - Burn-in period estimation

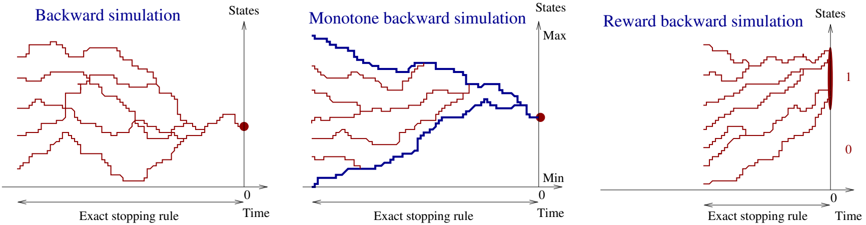
    $\Rightarrow$ Biased sampling
- Alternatives
  - Regeneration (MRMC tool)
  - Perfect sampling (Ψ^2 tool)

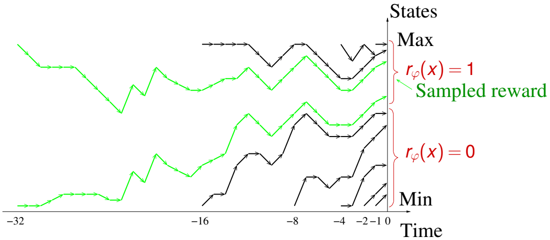
Backward Simulation Schemes





Backward Simulation Schemes





Synthesis

- Advantages
  - Steady state is exact (perfect sample)
  - Unbiased sampling of the steady-state
  - Very efficient under monotonicity
  - Very efficient for rare probability verification
  - Generic events (monotone and not) implemented in ψ^2 enabling to describe a wide range of systems
- Drawbacks
  - Monotonicity study of a system
    - If system is monotone: has to be proven
    - If not, "extended sandwiching technique": envelopes (not always efficient)
- A perfect sampler ψ^2 proposed by MESCAL INRIA Team
  - Samples rewards of the stationary distribution of large Markov chains

Synthesis

- Advantages
  - Steady state is exact (perfect sample)
  - Unbiased sampling of the steady-state
  - Very efficient under monotonicity
  - Very efficient for rare probability verification
  - Generic events (monotone and not) implemented in ψ^2 enabling to describe a wide range of systems
- Drawbacks
  - Monotonicity study of a system
    - If system is monotone: has to be proven
    - If not, "extended sandwiching technique": envelopes (not always efficient)
- A perfect sampler ψ^2 proposed by MESCAL INRIA Team
  - Samples rewards of the stationary distribution of large Markov chains

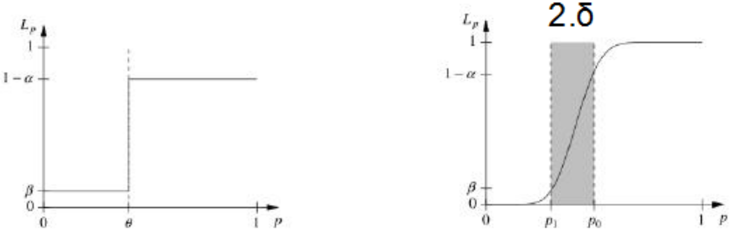
Synthesis

- Advantages
  - Steady state is exact (perfect sample)
  - Unbiased sampling of the steady-state
  - Very efficient under monotonicity
  - Very efficient for rare probability verification
  - Generic events (monotone and not) implemented in ψ^2 enabling to describe a wide range of systems
- Drawbacks
  - Monotonicity study of a system
    - If system is monotone: has to be proven
    - If not, "extended sandwiching technique": envelopes (not always efficient)
- A perfect sampler ψ^2 proposed by MESCAL INRIA Team
  - 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 $S_{\geq\theta}(\varphi)$ 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)

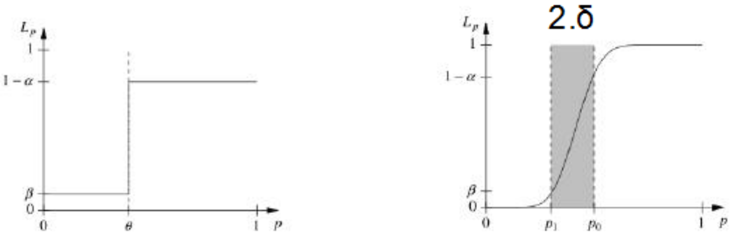


(a) Prob. of accepting H (ideal)

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

Statistical Hypothesis Testing (SHT)

- Estimate the probability p that φ of a given formula $S_{\geq\theta}(\varphi)$ 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)

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

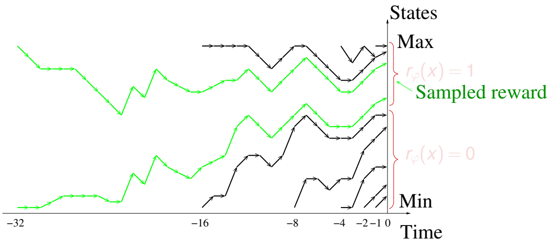
   - If $Y \geq m$ then accepting H_0 (YES)
   - Else If $Y < m$ then accepting H_1 (NO)
   - where m is the acceptance threshold of the statistical test
3. Statistical test strength (n, m) depends on (α, β) and on δ where n is the total sample size

Outline

Verification of CSL Steady State Formula

- SMC of ψ=$\mathcal{S}_{\geq\theta}(\varphi)$ by functional and/or monotone perfect simulation
- Check if the steady-state samples (x) satisfies φ or not
- By associating reward $r_\varphi(x)$ to each state x for the given property φ:

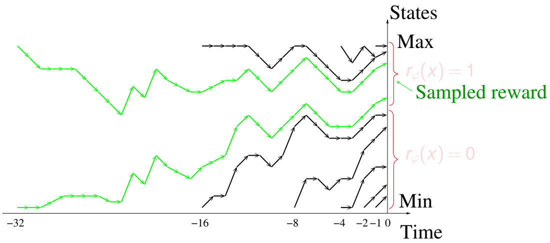
$$r_\varphi(x) = 1, \text{ if } x \models \varphi \quad (1)$$
$$r_\varphi(x) = 0, \text{ otherwise } x \not\models \varphi$$



Verification of CSL Steady State Formula

- SMC of $\psi = \mathcal{S}_{\geq\theta}(\varphi)$ by functional and/or monotone perfect simulation
- Check if the steady-state samples (x) satisfies φ or not
- By associating reward $r_\varphi(x)$ to each state x for the given property φ:

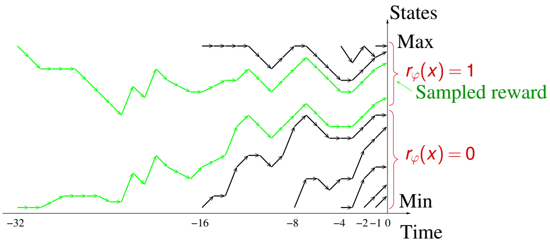
$$\begin{aligned} r_\varphi(x) &= 1, \text{ if } x \models \varphi \\ r_\varphi(x) &= 0, \text{ otherwise } x \not\models \varphi \end{aligned} \quad (1)$$



Verification of CSL Steady State Formula

- SMC of $\psi = \mathcal{S}_{\geq\theta}(\varphi)$ by functional and/or monotone perfect simulation
- Check if the steady-state samples (x) satisfies φ or not
- By associating reward $r_\varphi(x)$ to each state x for the given property φ:

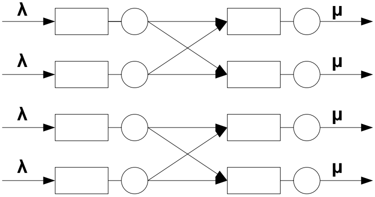
$$\begin{aligned} r_\varphi(x) &= 1, \text{ if } x \models \varphi \\ r_\varphi(x) &= 0, \text{ otherwise } x \not\models \varphi \end{aligned} \quad (1)$$



Outline

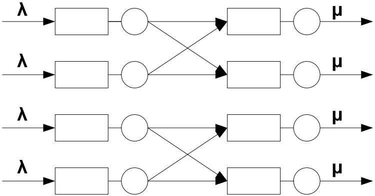
Models

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)



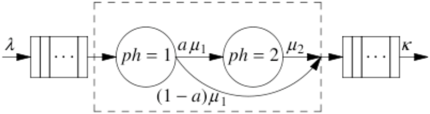
Models

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



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 ($< \theta$ or not at steady state)
2. Multistage delta queueing network with 8 queues (MDN)
   - At least one queue of the second stage of MDN is full ($< \theta$ or not at steady state)
3. Tandem Queuing Network with coaxian server (TQN-Cox)
   - The overall system is full ($< \theta$ or not at steady state)

Outline

1. PRISM tool (Numerical MC, Oxford University)
   - Computes probabilities for each reachable state
   - Solves system of linear equations to find probabilities with convergence precision ϵ
2. ψ^2 with SHT tool (SMC, Grenoble and UPEC Universities)
   - Perfect sampling (Functional)
   - Verification by Statistical Hypothesis Testing with precision (α, β, δ)
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 $\alpha = \beta = 10^{-2}$
   - Rare probability dependability properties: $\theta = 0.001$

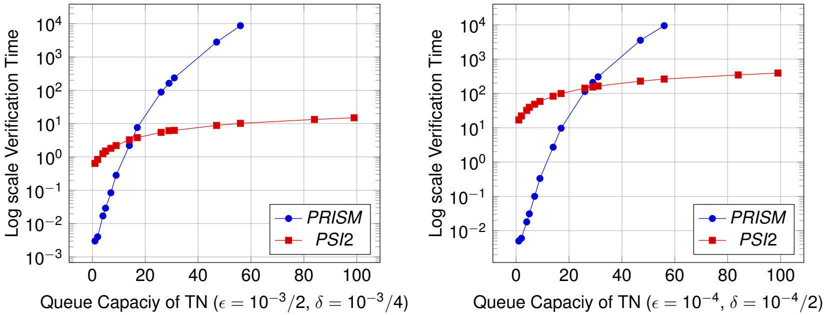
1. PRISM tool (Numerical MC, Oxford University)
   - Computes probabilities for each reachable state
   - Solves system of linear equations to find probabilities with convergence precision ϵ
2. ψ^2 with SHT tool (SMC, Grenoble and UPEC Universities)
   - Perfect sampling (Functional)
   - Verification by Statistical Hypothesis Testing with precision (α, β, δ)
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 $\alpha = \beta = 10^{-2}$
   - Rare probability dependability properties: $\theta = 0.001$

1. PRISM tool (Numerical MC, Oxford University)
   - Computes probabilities for each reachable state
   - Solves system of linear equations to find probabilities with convergence precision ϵ
2. ψ^2 with SHT tool (SMC, Grenoble and UPEC Universities)
   - Perfect sampling (Functional)
   - Verification by Statistical Hypothesis Testing with precision (α, β, δ)
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 $\alpha = \beta = 10^{-2}$
   - Rare probability dependability properties: $\theta = 0.001$

Outline

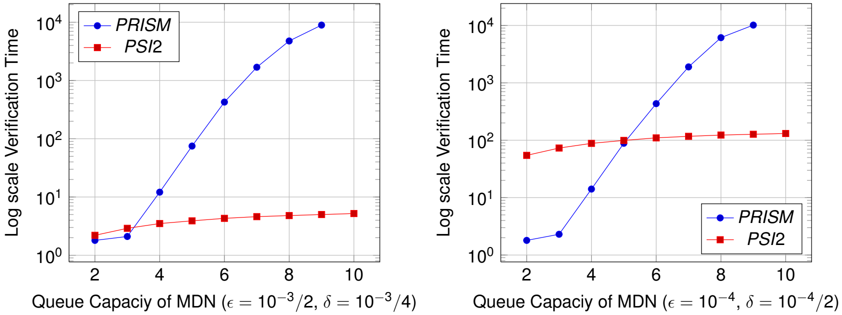
Tandem Network (TN)

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



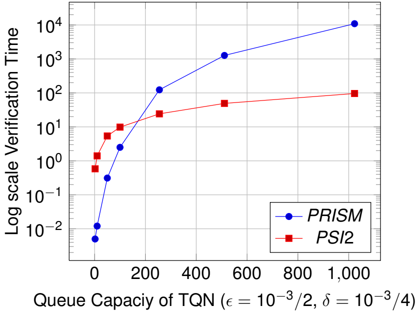
Multistage Delta Network (MDN)

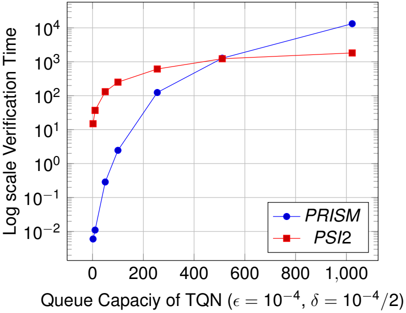
- Model and property: 2 *stages* and 4 *buffers*/*stage*, $\lambda = 0.9$, $\mu = 1$, $(\tau_{rout1}, \tau_{rout2}) = (0.8, 0.6)$, $S_{<\theta}$ (*last-stage-full*) where $\theta = 0.001$



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 $\theta = 0.001$





Synthesis and Discussions

1. Variation of precision parameters ϵ (numerical) and δ (statistical)
   - Verification time dependence on on δ is considerable but on ϵ is negligible
2. Variation of state space size (Max. queue capacity)
   - Verification time dependence on state space size is negligible in ψ^2 (functional) but is considerable in PRISM
3. Memory limitation problem
   - Memory is never exhausted in ψ^2 but is proportional to the number of states in PRISM

Synthesis and Discussions

1. Variation of precision parameters ϵ (numerical) and δ (statistical)
   - Verification time dependence on on δ is considerable but on ϵ is negligible
2. Variation of state space size (Max. queue capacity)
   + Verification time dependence on state space size is negligible in ψ^2 (functional) but is considerable in PRISM
3. Memory limitation problem
   + Memory is never exhausted in ψ^2 but is proportional to the number of states in PRISM

Synthesis and Discussions

1. Variation of precision parameters ϵ (numerical) and δ (statistical)
   - Verification time dependence on on δ is considerable but on ϵ is negligible
2. Variation of state space size (Max. queue capacity)
   + Verification time dependence on state space size is negligible in ψ^2 (functional) but is considerable in PRISM
3. Memory limitation problem
   + Memory is never exhausted in ψ^2 but is proportional to the number of states in PRISM

1. Memory limits obtained in PRISM:
   - TN case: For $N_{max} = 99$ ($|X| = 10^8$)
   - MDN case: For $N_{max} = 10$ ($|X| = 1.1 * 10^8$)
   - TQN case: For $N_{max} = 7500$ ($|X| = 2.1 * 10^8$)
2. 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, $O((N_{max} + 1)^{32})$)
3. TQN case (Non monotone model)
   + Efficient results for this example when using envelopes

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)
   - + PRISM have memory problem (limiting state space sizes)

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)
   - + PRISM have memory problem (limiting state space sizes)

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)
   + PRISM have memory problem (limiting state space sizes)

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)
   - + PRISM have memory problem (limiting state space sizes)

Future works

1. Compare ψ^2 with SHT tool with the MRMC tool (Current)
   - Perfect Simulation vs. Regeneration Simulation
   - SHT vs. Confidence Intervals
   - Memory limitation problem in MRMC tool
2. SMC of CSL time unbounded until formulas

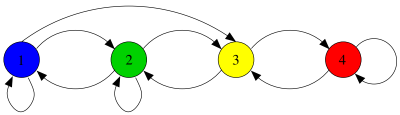
Future works

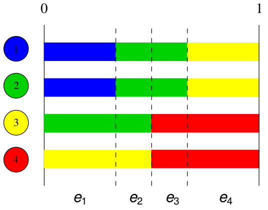
1. Compare ψ^2 with SHT tool with the MRMC tool (Current)
   - Perfect Simulation vs. Regeneration Simulation
   - SHT vs. Confidence Intervals
   - Memory limitation problem in MRMC tool
2. SMC of CSL time unbounded until formulas

Future works

1. Compare ψ^2 with SHT tool with the MRMC tool (Current)
   - Perfect Simulation vs. Regeneration Simulation
   - SHT vs. Confidence Intervals
   - Memory limitation problem in MRMC tool
2. SMC of CSL time unbounded until formulas

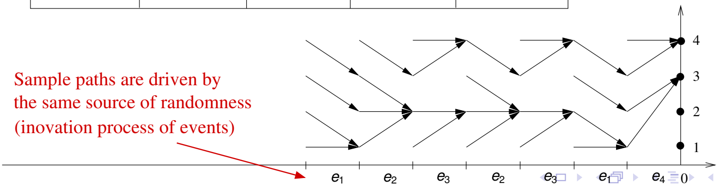
Event modelling of a Markov chain





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 → 3, 3 → 2, 2 → 1, 1 → 1	4 → 3, 3 → 2, 2 → 2, 1 → 2	4 → 4, 3 → 4, 2 → 2, 1 → 2	4 → 4, 3 → 4, 2 → 3, 1 → 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