Probabilistic Verification of Discrete Event Systems

Håkan L. S. Younes

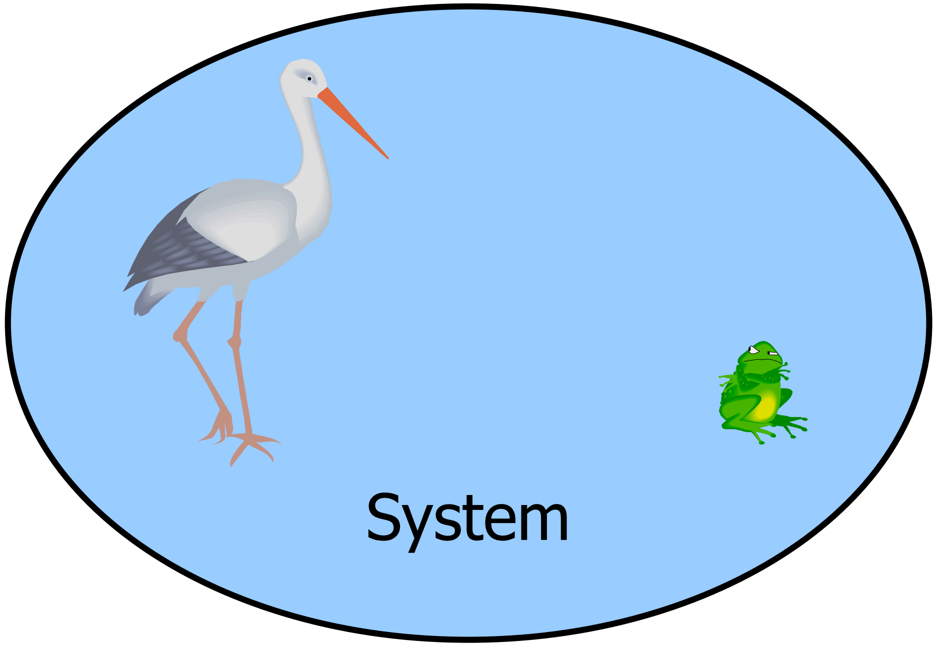
Reid G. Simmons

(initial work performed at HTC, Summer 2001)

Introduction

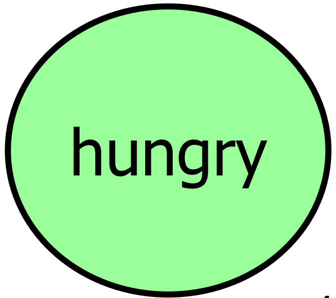
- Goal: Verify temporal properties of general discrete event systems
  - Probabilistic, real-time properties
  - Expressed using CSL
- Approach: Acceptance sampling
  - Guaranteed error bounds
  - Any-time properties

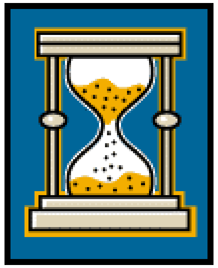
"The Hungry Stork"



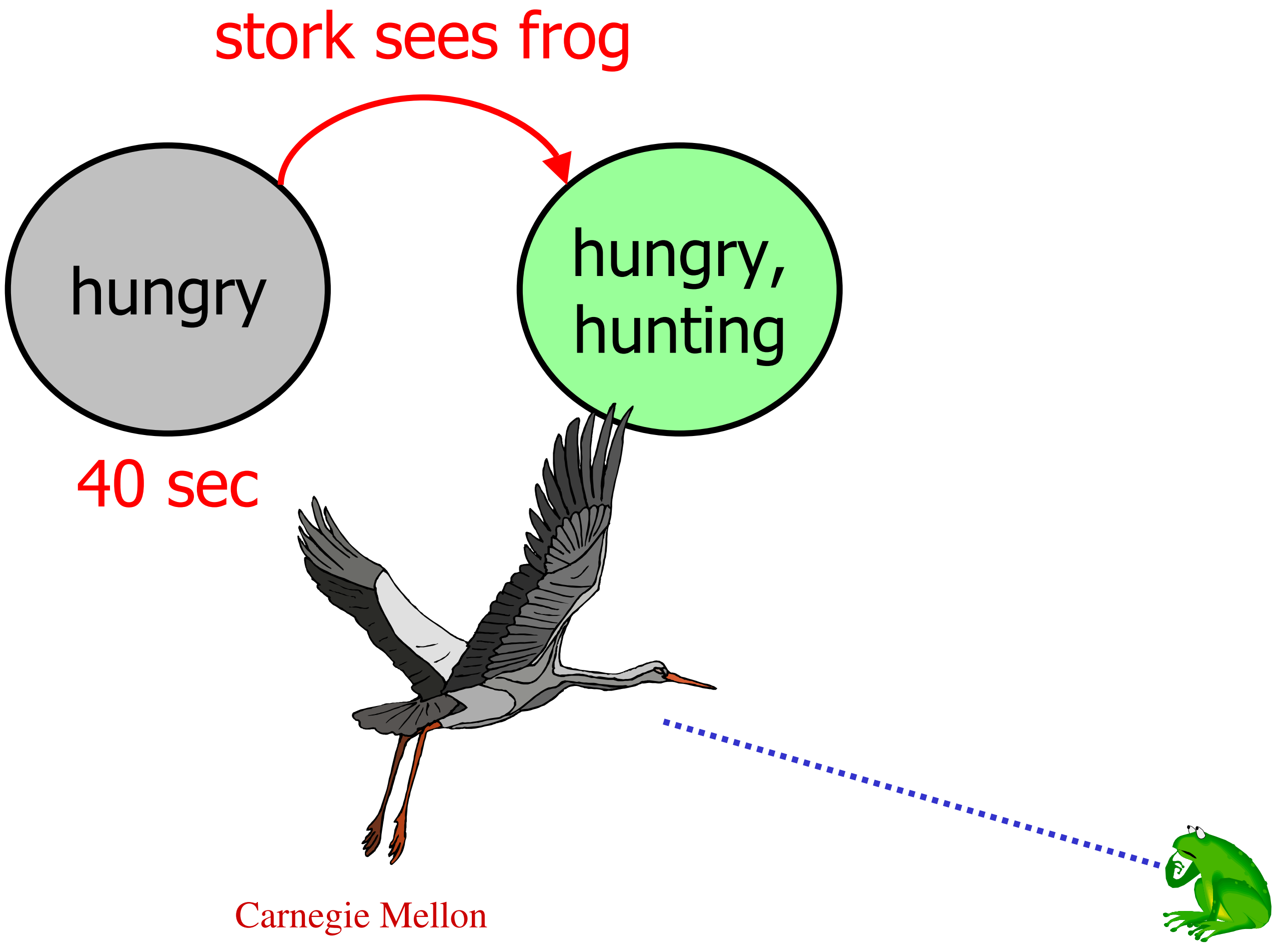
"The *probability is at least 0.7* that the stork satisfies its hunger *within 180 seconds*"

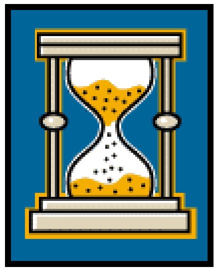
“The Hungry Stork” as a Discrete Event System



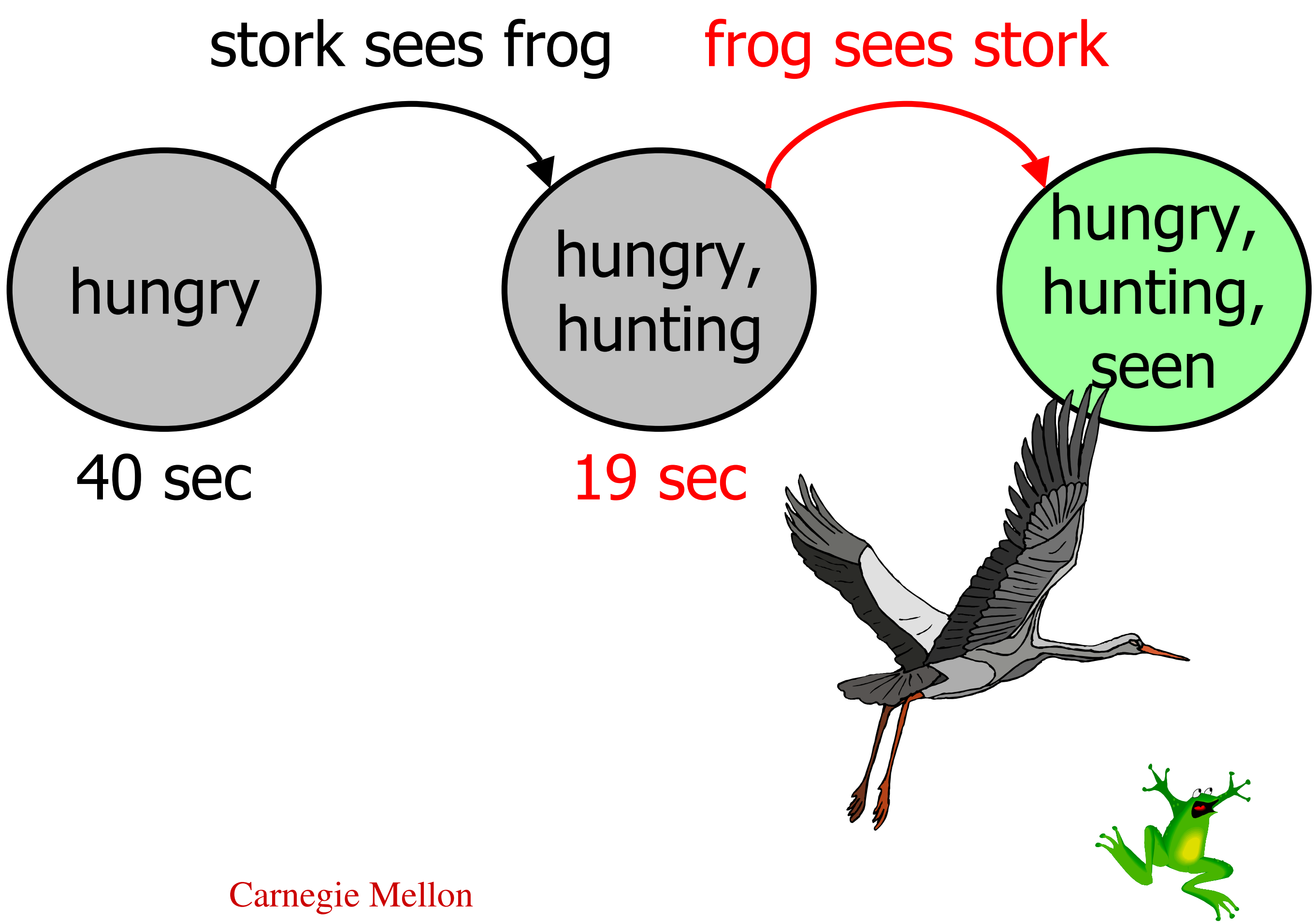


"The Hungry Stork" as a Discrete Event System

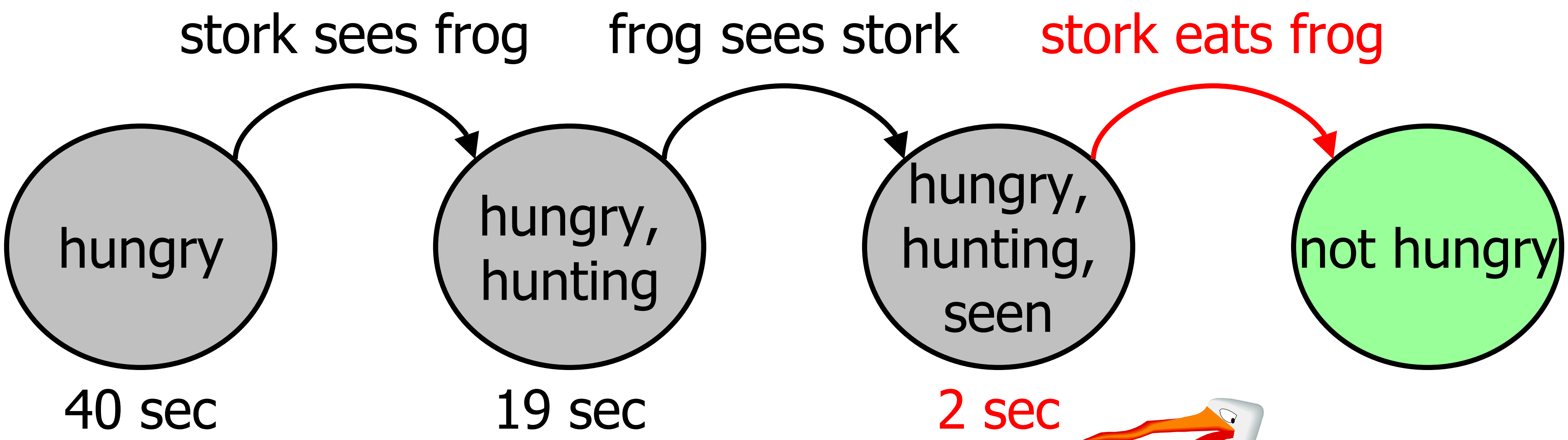


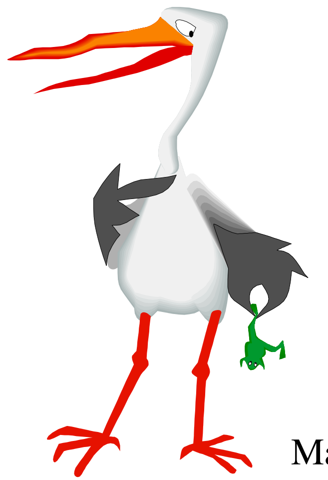


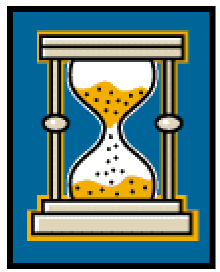
"The Hungry Stork" as a Discrete Event System



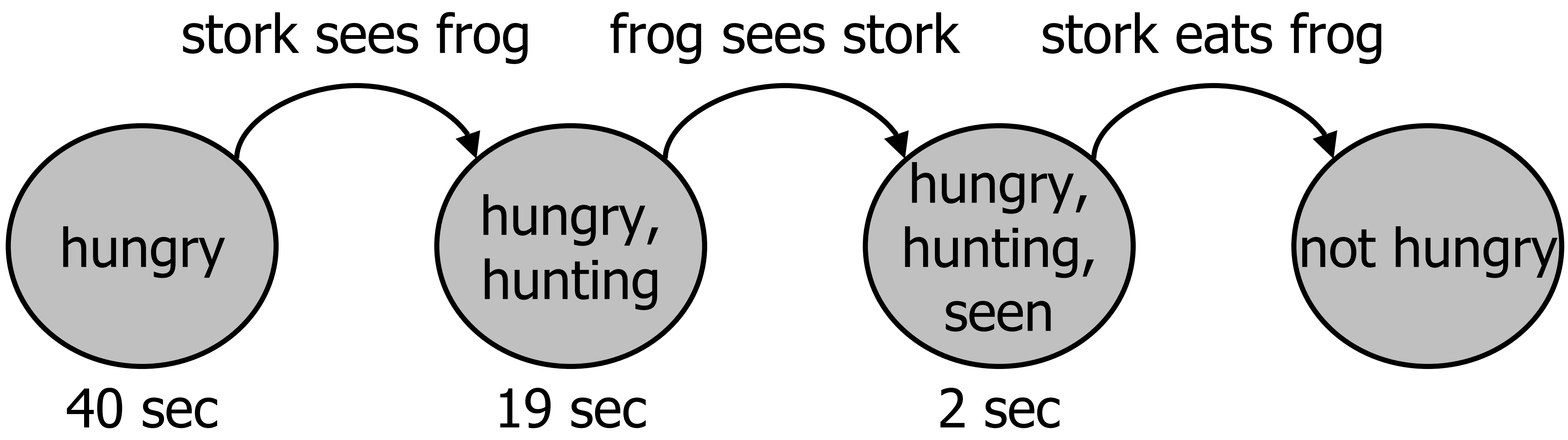
"The Hungry Stork" as a Discrete Event System







"The Hungry Stork" as a Discrete Event System



For this execution path, at least, the property holds…

(total time < 180 sec)

Verifying Probabilistic Properties

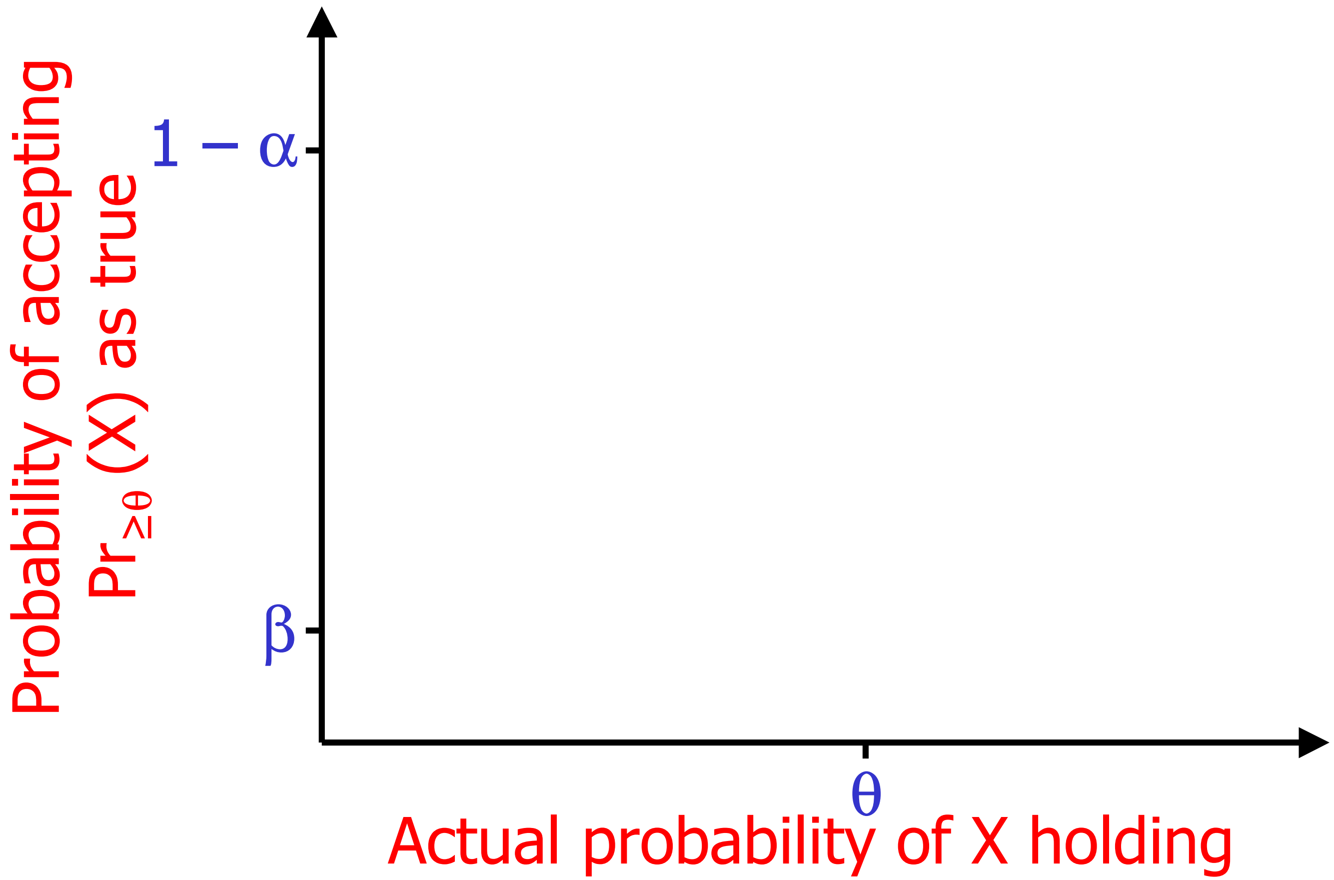
- Properties of the form: $Pr_{\geq\theta}(X)$
- Symbolic Methods
  - \+ Exact solutions
  - \- Works for a restricted class of systems
- Sampling
  - \+ Works for all systems that can be simulated
  - \- Solutions not guaranteed

Our Approach: Acceptance Sampling

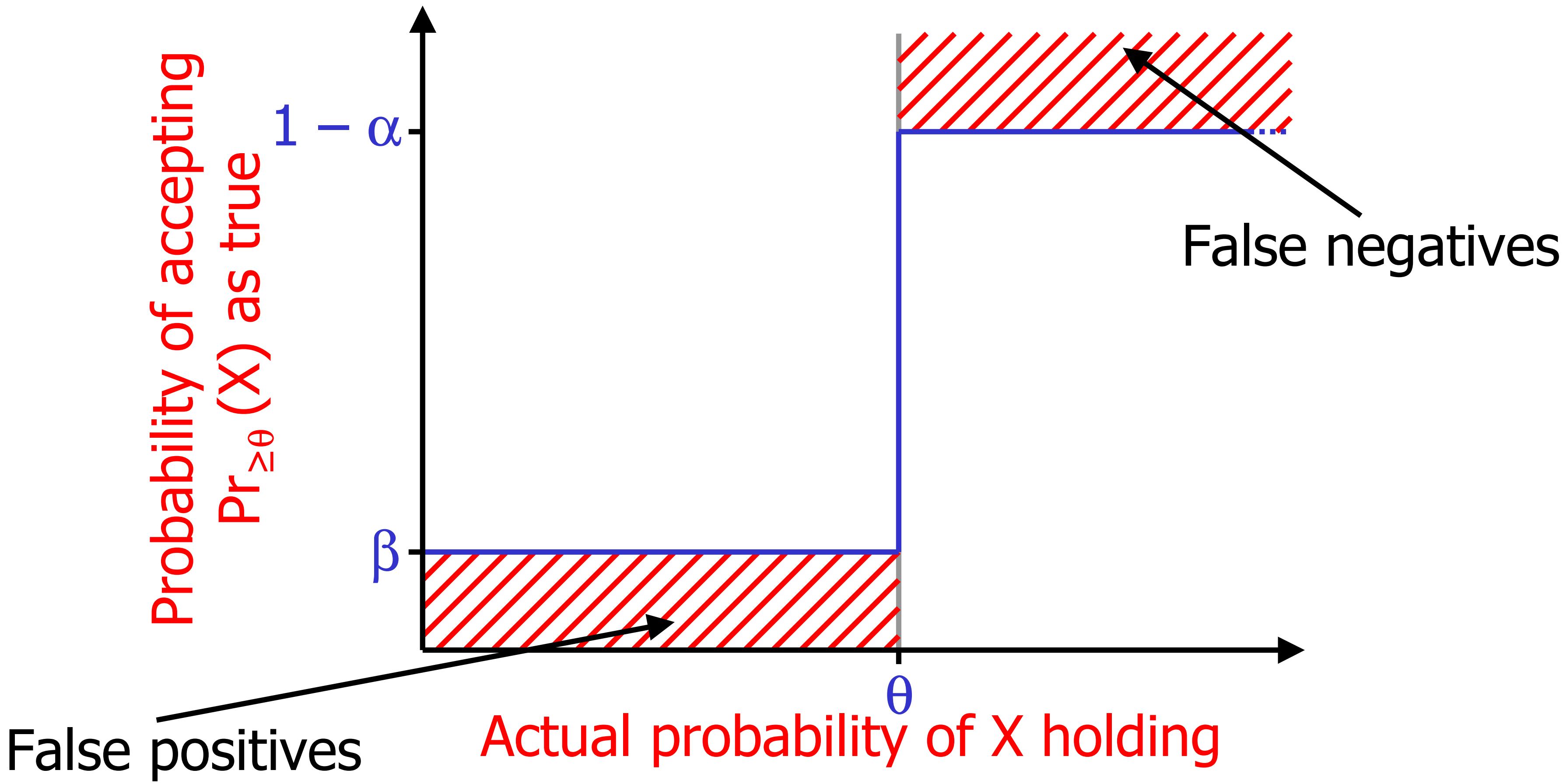
- Use *simulation* to generate sample execution paths
  - Samples based on stochastic discrete event models

- How many samples are "enough"?
  - Probability of false negatives $\leq \alpha$
  - Probability of false positives $\leq \beta$

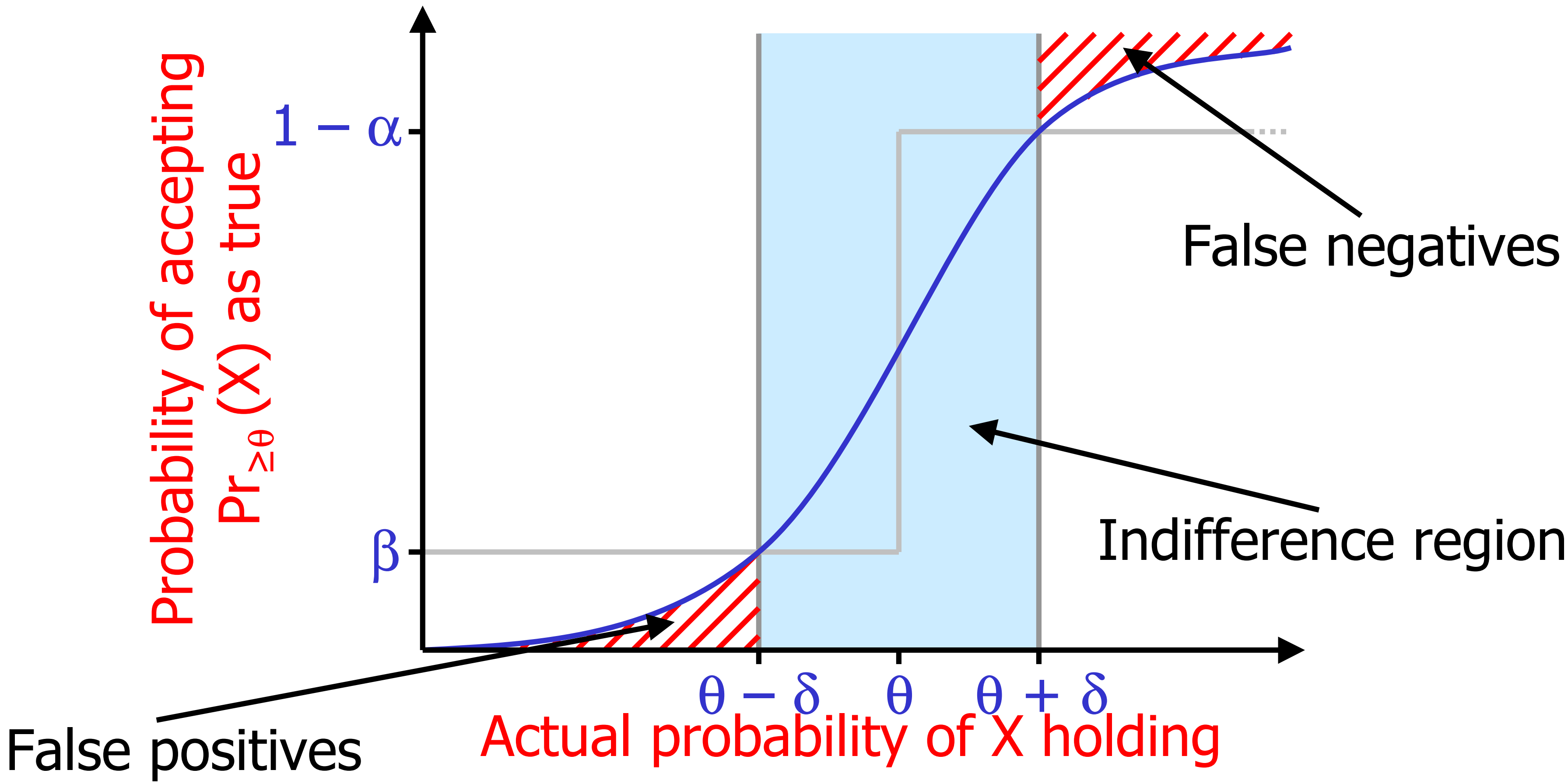
Performance of Test



Ideal Performance

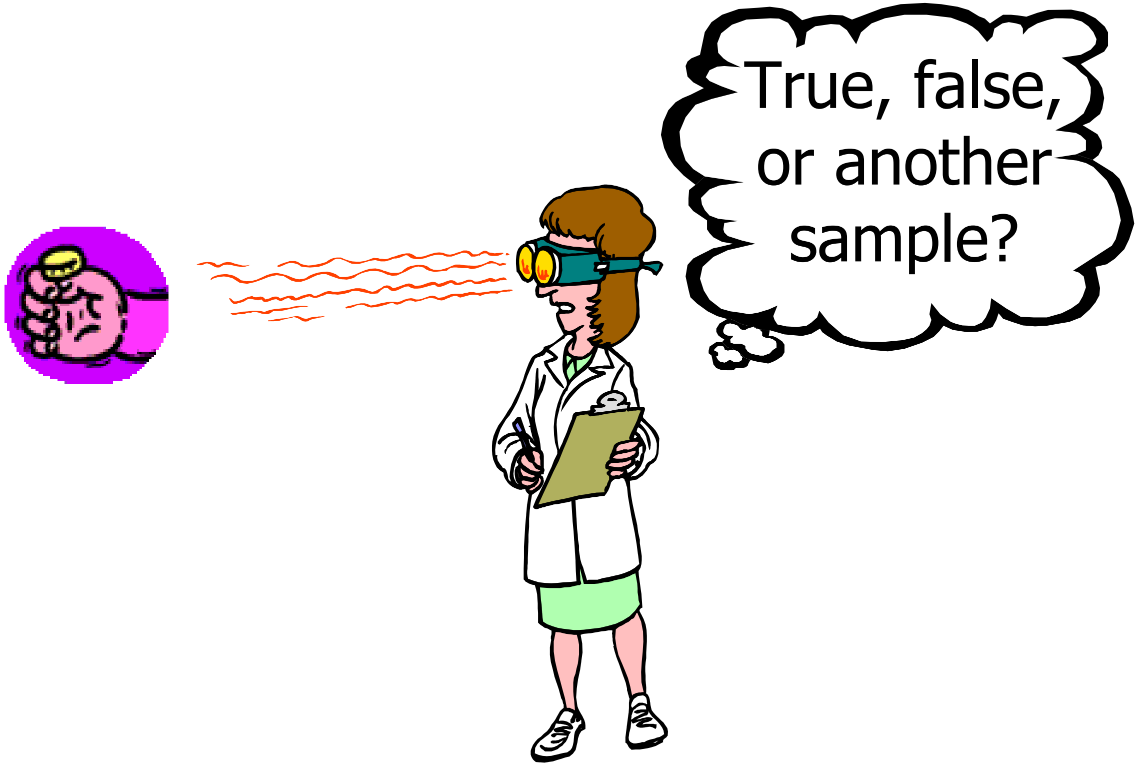


Actual Performance

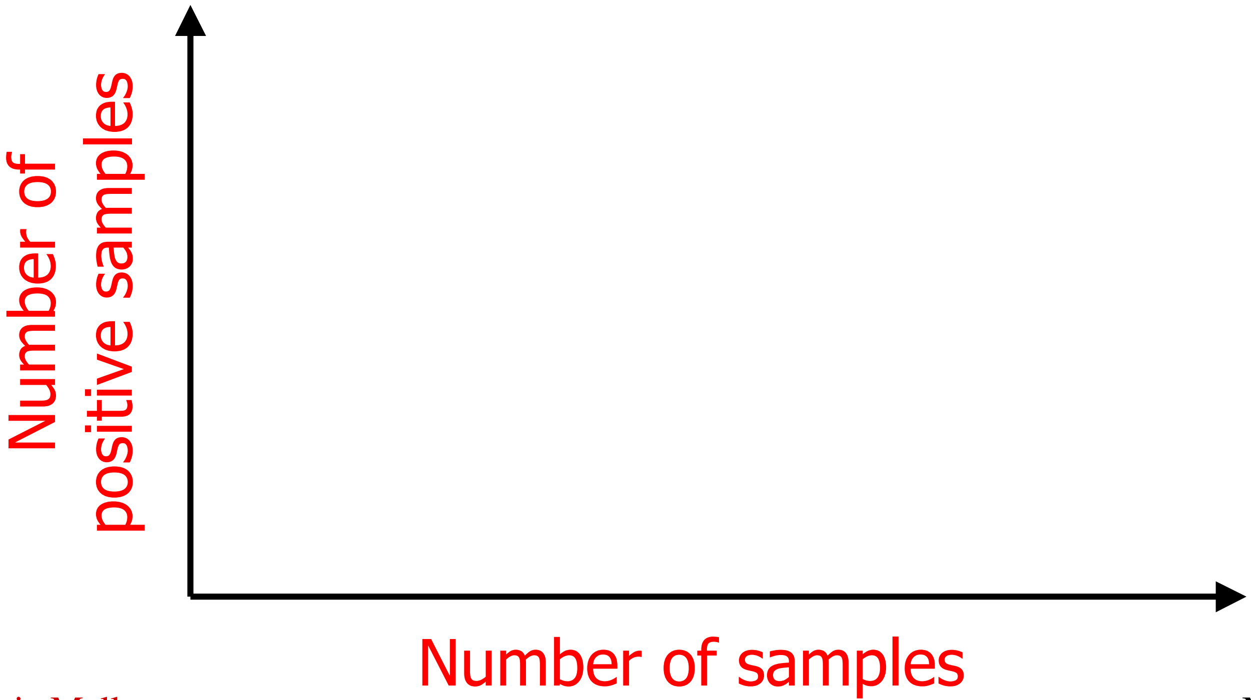


Sequential Acceptance Sampling

- Hypothesis: $Pr_{\geq\theta}(X)$

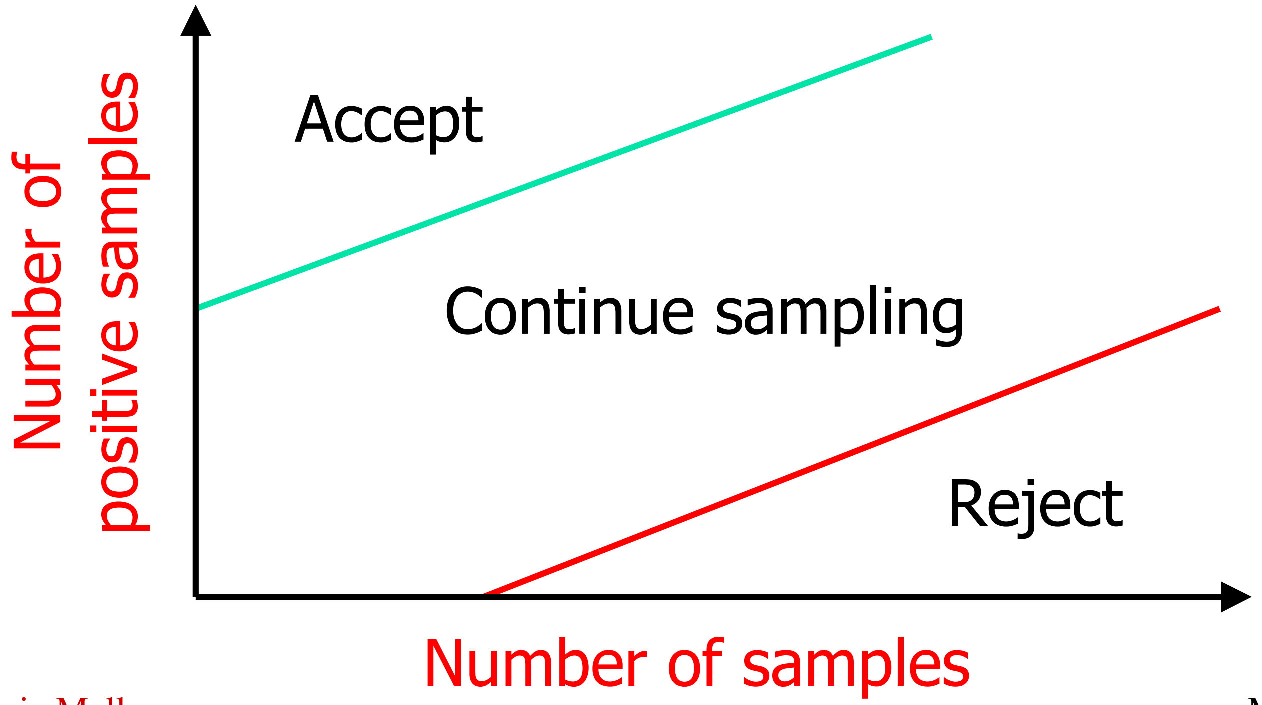


Graphical Representation of Sequential Test

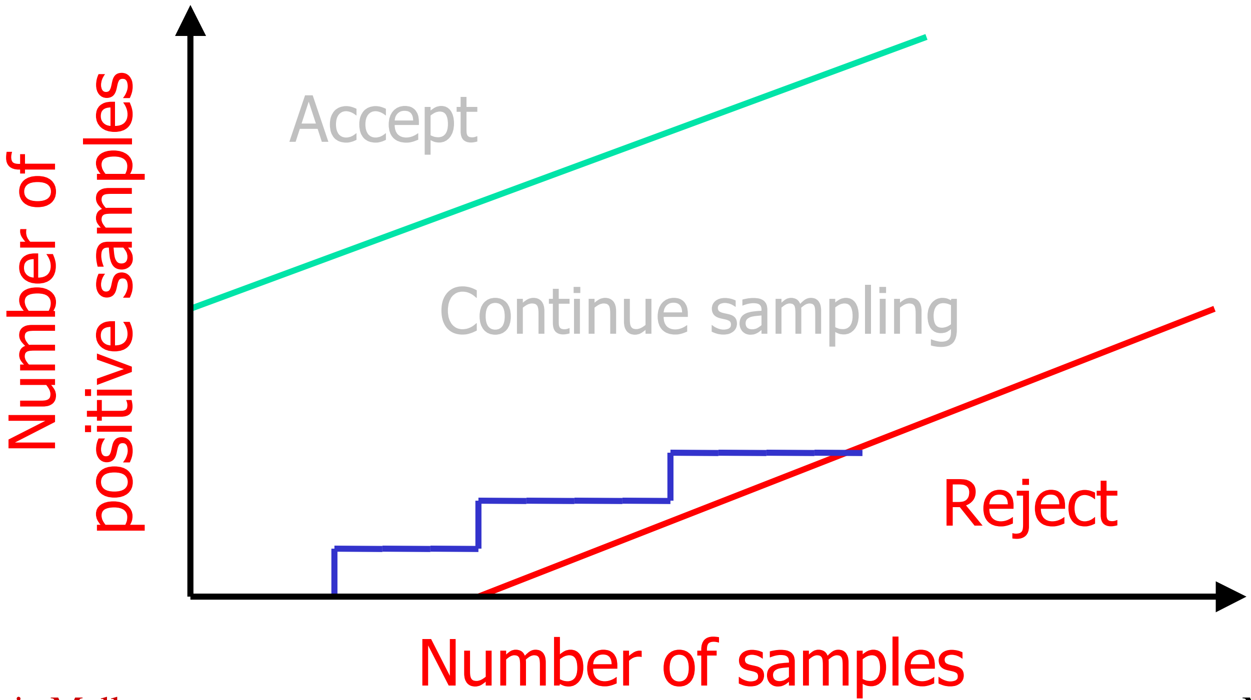


Graphical Representation of Sequential Test

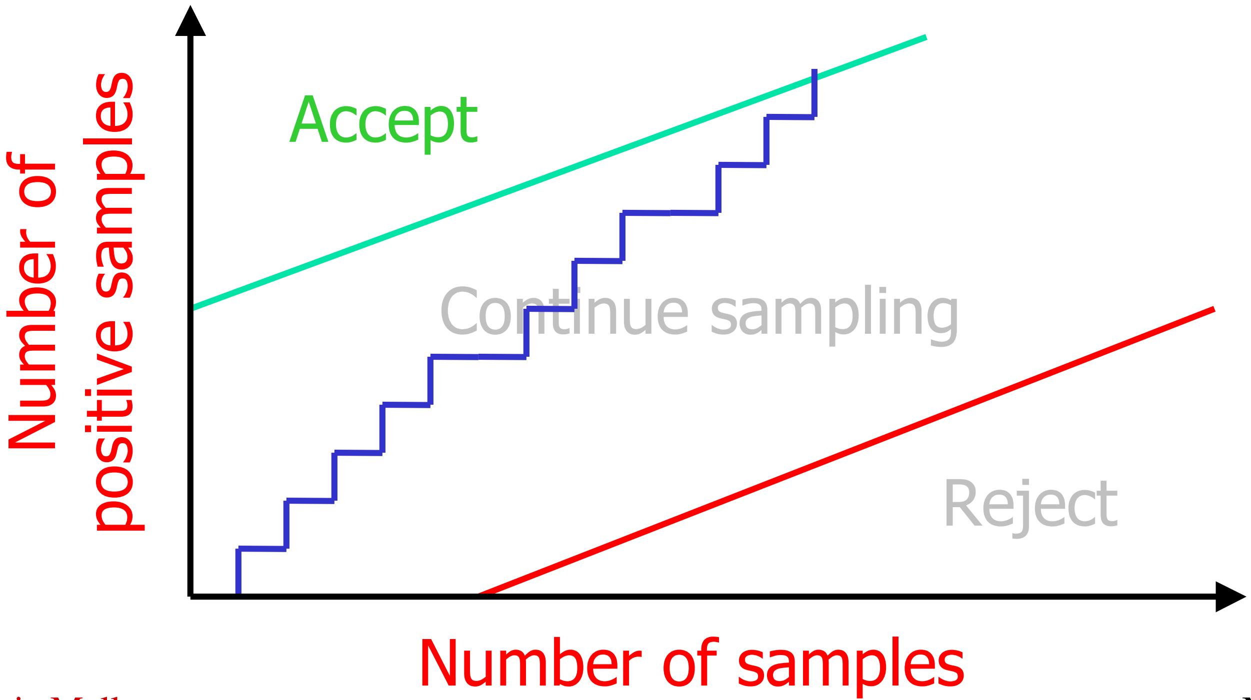
- We can find an acceptance line and a rejection line given θ, δ, α, and β



Graphical Representation of Sequential Test



Graphical Representation of Sequential Test



Verifying Properties

- Verify $Pr_{\geq\theta}(\rho)$ with error bounds α and β
  - Generate sample execution paths using simulation
  - Verify ρ over each sample execution path
    - If ρ is true, then we have a positive sample
    - If ρ is false, then we have a negative sample
  - Use sequential acceptance sampling to test the hypothesis $Pr_{\geq\theta}(\rho)$
- How to express probabilistic, real-time temporal properties as acceptance tests?

Continuous Stochastic Logic (CSL)

- State formulas
  - Standard logic operators: $\neg\varphi$, $\varphi_1 \wedge \varphi_2$ …
  - Probabilistic operator: $Pr_{\geq\theta}(\rho)$
- Path formulas
  - Time-bounded Until: $\varphi_1 \, U^{\leq t} \, \varphi_2$

- $Pr_{\geq 0.7}(\text{true } U^{\leq 180} \neg\text{hungry})$
- $Pr_{\geq 0.9}(Pr_{\leq 0.1}(\text{queue-full}) \, U^{\leq 60} \text{ served})$

Verification of Conjunction

- Verify $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$ with error bounds α and β

- What error bounds to choose for the φ_i's?
  - Naïve: $\alpha_i = \alpha/n$, $\beta_i = \beta/n$
  - Accept if all conjuncts are true
  - Reject if some conjunct is false

Verification of Conjunction

- Verify $\varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n$ with error bounds α and β
  1. Verify each φ_i with error bounds α and β'
  2. Return **false** as soon as any φ_i is verified to be false
  3. If all φ_i are verified to be true, verify each φ_i again with error bounds α and β/n
  4. Return **true** iff all φ_i are verified to be true

"Fast reject"

Verification of Conjunction

- Verify $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$ with error bounds α and β
  1. Verify each φ_i with error bounds α and β'
  2. Return **false** as soon as any φ_i is verified to be false
  3. If all φ_i are verified to be true, verify each φ_i again with error bounds α and β/n
  4. Return **true** iff all φ_i are verified to be true

"Rigorous accept"

Verification of Path Formulas

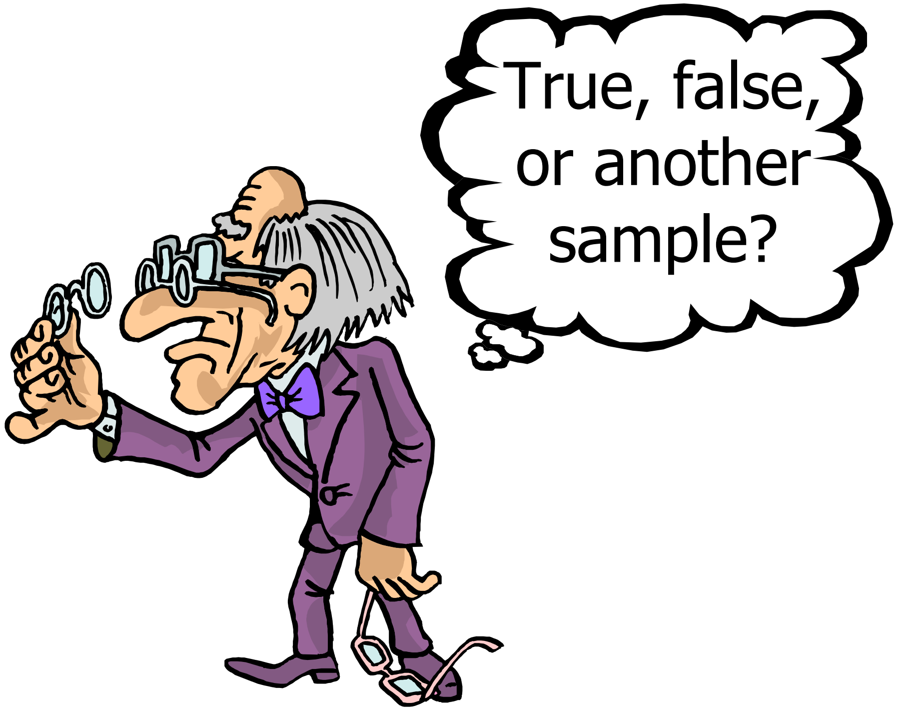
- To verify $\varphi_1\ U^{\leq t}\ \varphi_2$ with error bounds α and β
  - Convert to disjunction
    - $\varphi_1\ U^{\leq t}\ \varphi_2$ holds if φ_2 holds in the first state, or if φ_2 holds in the second state and φ_1 holds in all prior states, or …

More on Verifying Until

- Given $\varphi_1 \; U^{\leq t} \; \varphi_2$, let n be the index of the first state more than t time units away from the current state
- Disjunction of n conjunctions c_1 through c_n, each of size i
- Simplifies if φ_1 or φ_2, or both, do not contain any probabilistic statements

Verification of Nested Probabilistic Statements

- Suppose ρ, in $Pr_{\geq\theta}(\rho)$, contains probabilistic statements

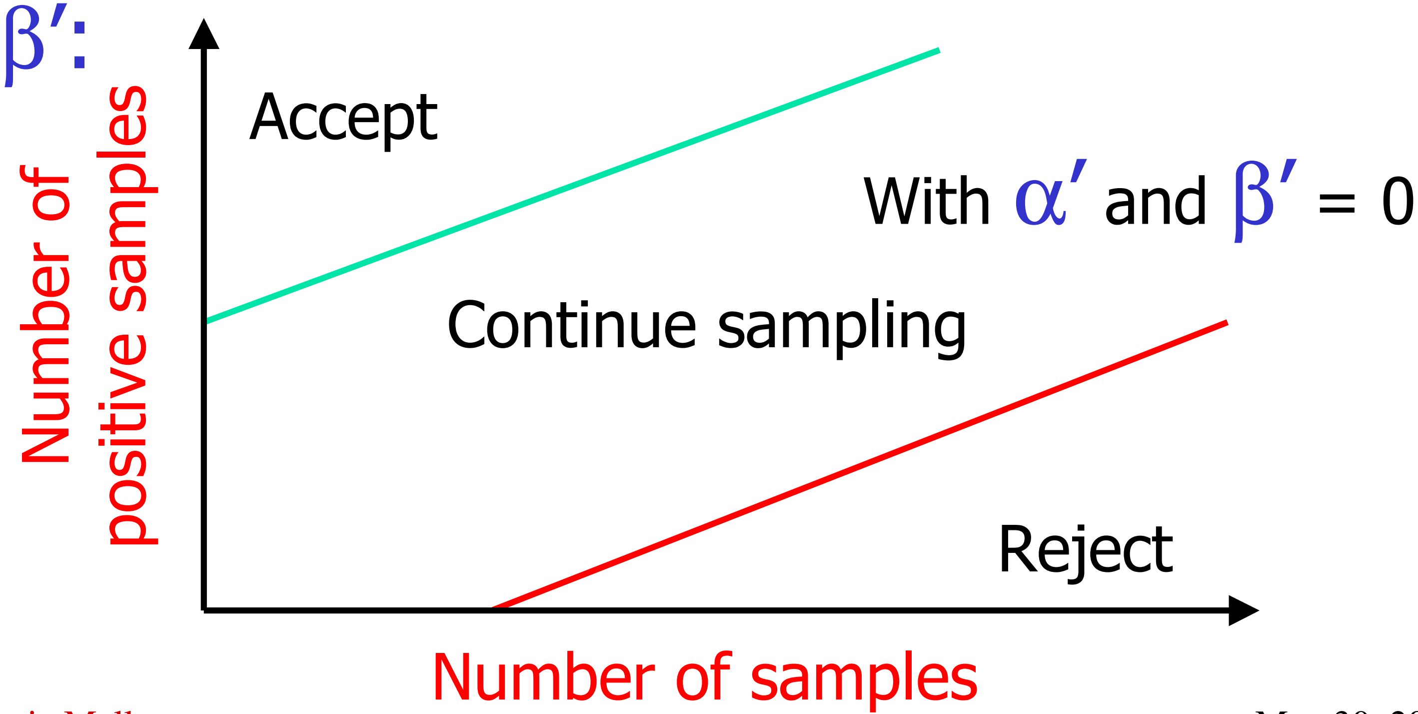


Verification of Nested Probabilistic Statements

- Suppose ρ, in $Pr_{\geq\theta}(\rho)$, contains probabilistic statements
  - $Pr_{\geq 0.9}(Pr_{\leq 0.1}(\text{queue-full})\ U^{\leq 60}\ \text{served})$
  - How to specify the error bounds α' and β' when verifying ρ?

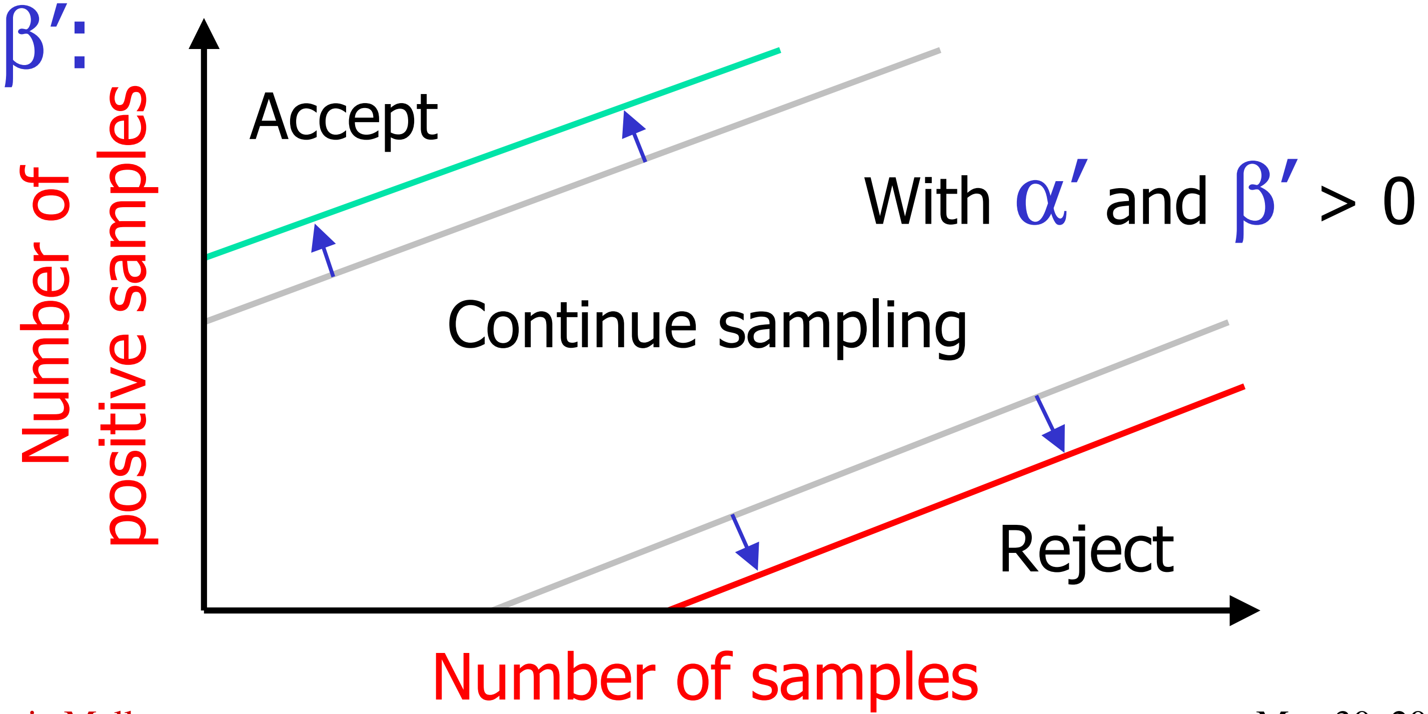
Modified Test

- find an acceptance line and a rejection line given θ, δ, α, β, α', and β':

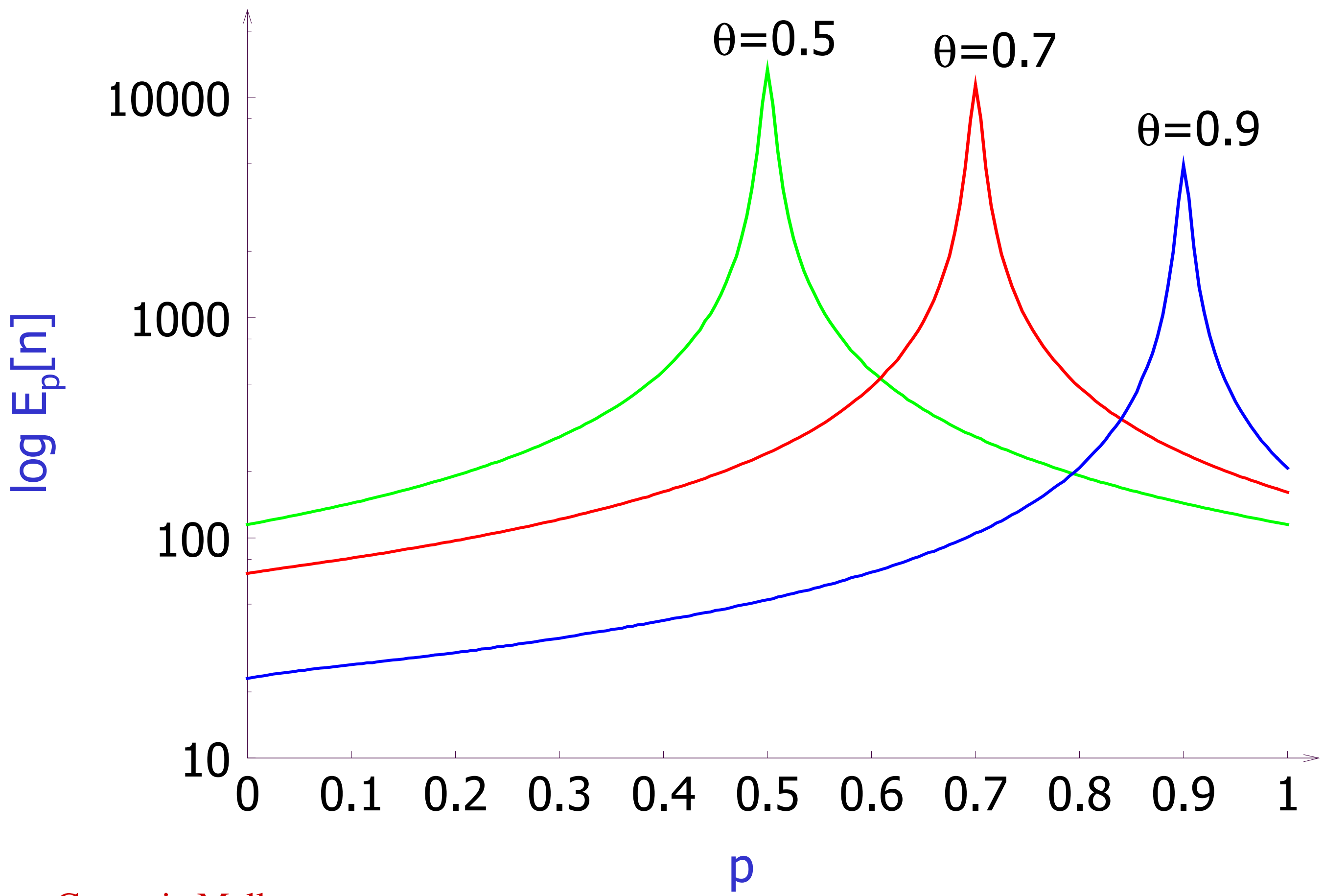


Modified Test

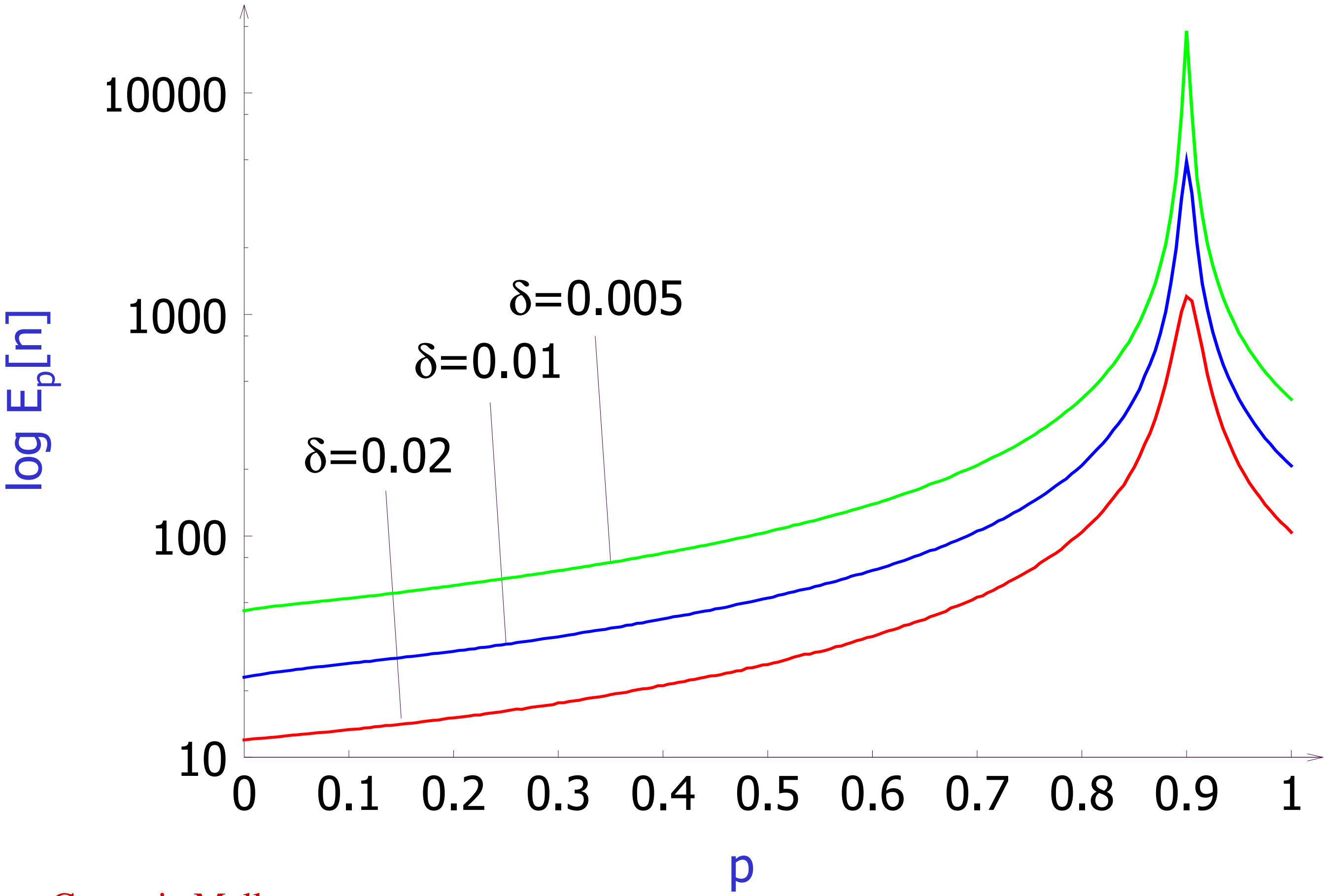
- find an acceptance line and a rejection line given θ, δ, α, β, α', and β':



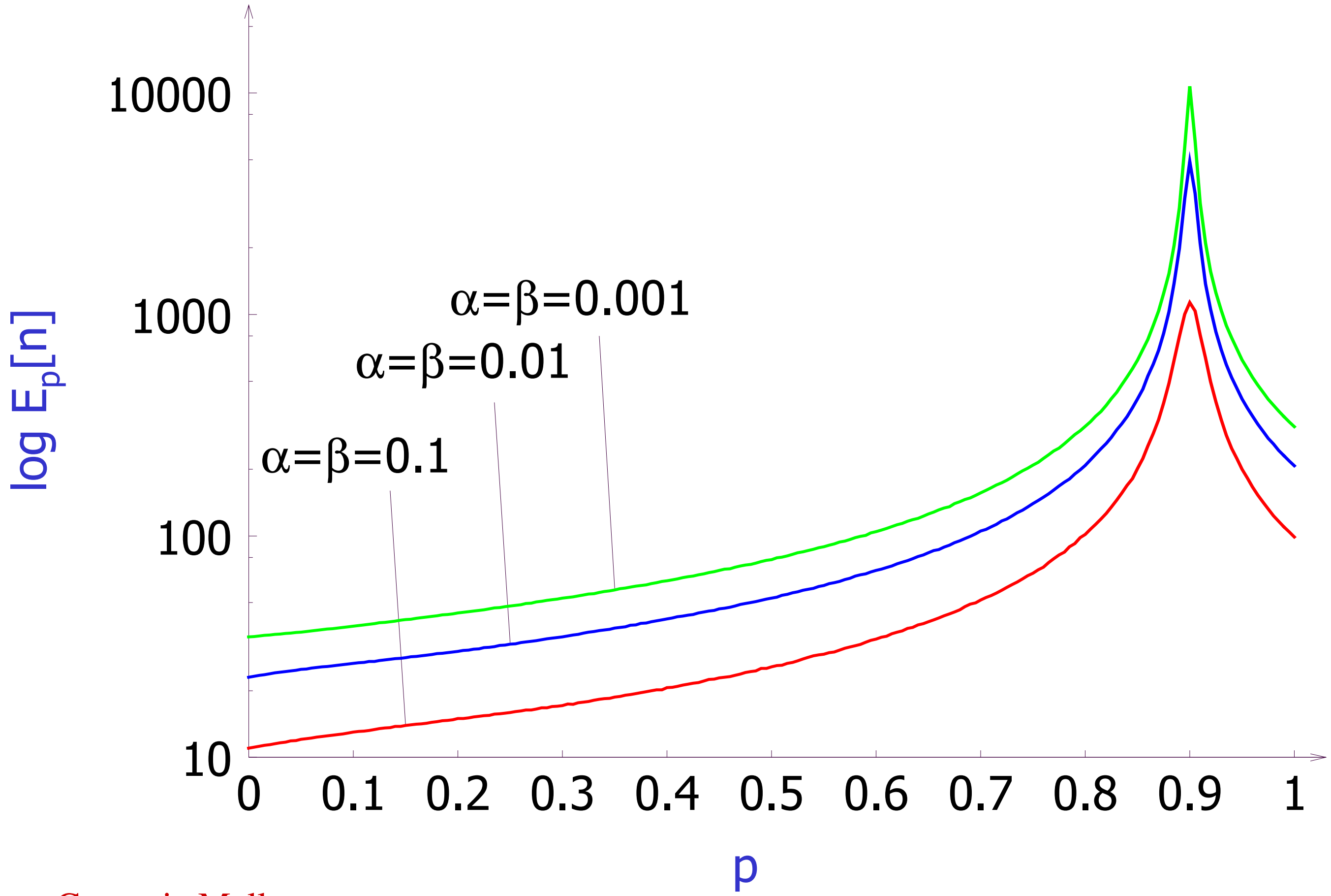
Performance



Performance



Performance



Summary

- Algorithm for probabilistic verification of discrete event systems
- Sample execution paths generated using simulation
- Probabilistic properties verified using sequential acceptance sampling
- Properties specified using CSL

Future Work

- Apply to hybrid dynamic systems
- Develop heuristics for formula ordering and parameter selection
- Use verification to aid policy generation for real-time stochastic domains