Index

- Introduction
- Probabilistic Systems
- Probabilistic Logics
- PRISM
- Performance Evaluation
- Model Checking and Performance Evaluation
- Challenges
Introduction

- Non-probabilistic model checking
  - Qualitative correctness
    - e.g. is there any execution path in which more than one processes are in the critical section?

- Probabilistic model checking:
  - Quantitative correctness:
    - e.g. is the probability that more than one processes are in the critical section below 0.1?
Introduction (Cont.)

- Model checking of systems that exhibit probabilistic behavior:
  - e.g. randomized algorithms/protocols
  - e.g. systems with failures/unreliability

- Markov models
  - Discrete-Time Markov Chains (DTMCs)
  - Markov Decision Processes (MDPs)
  - Continuous-Time Markov Chains (CTMCs)

- Probabilistic Timed Automata (PTAs) (not covered here)
Probabilistic Systems
Discrete-Time Markov Chains (DTMCs)

- State-transition systems augmented with probabilities
- Transitions occur in discrete time steps
Probabilistic Systems
Markov Decision Processes (MDPs)

- DTMCs + non-determinism
Probabilistic Systems
Continuous-Time Markov Chains (CTMCs)

- Transitions
  - Can occur in any real-valued time
  - Modeled using exponential distribution with rate $\lambda$
    - probability triggered before $t$ time units: $1 - e^{-\lambda t}$

- Example:
  - state space: $S = \{s_i \mid i=0..3\}$ where $s_i$ indicates $i$ jobs in queue
  - maximum size of the queue is 3
  - jobs arrive with rate $\lambda$ (i.e. mean inter-arrival time is $\lambda$)
  - jobs are served with rate $\mu$ (i.e. mean service time is $\mu$)
Probabilistic Logics

- Logics for specifying properties of probabilistic systems
- Extensions of CTL
  - PCTL (and PCTL *)
  - CSL
  - PTCTL \((not\ covered\ in\ here)\)
Probabilistic Logics

PCTL

- Temporal logic for describing properties of DTMCs (and MDPs)
- Extends CTL
- Key addition: probabilistic operator $P$

$$
\Phi ::= \text{true} \mid a \mid \neg \Phi \mid \Phi_1 \wedge \Phi_2 \mid P_{\sim p}[\psi] \quad \text{(state formulae)}
$$

$$
\psi ::= X\Phi \mid \Phi_1 \cup \Phi_2 \mid \Phi_1 \cup^{\leq k} \Phi_2 \quad \text{(path formulae)}
$$

$$
s \models P_{\sim p}[\psi] \quad \text{iff} \quad Pr(s \models \psi)_{\sim p},
$$

where $Pr(s \models \psi) = Pr_s\{\pi \in \text{Path}(s) | \pi \models \psi\}$. 
Probabilistic Logics

PCTL

- Examples:
  - $P_{<0.3}[F(Fail_{serverA} \land Fail_{serverB})]$: The probability of situation in which both server A and server B have failed is less than 0.3.
  - $down \rightarrow P_{>0.99}[true \cup_{\leq 10} Up]$: If the service is down, the probability of being repair in 10 time steps is greater than 0.99.
Probabilistic Logics

PCTL (Cont.)

- $P$ is quantitative extension of $A$ and $E$ of CTL
- $P_{\geq 0}[\psi]$ is identical to $E[\psi]$,
- $P_{\geq 1}[\psi]$ is (similar to but) weaker than $A[\psi]$,
- $A[F success]$ is not satisfied
  - Counterexample:
    - $s_0, s_1, s_2, ...$
  - $P_{\geq 1}[F success]$ is satisfied
    - Probability of $s_0, s_1, s_2, ...$ is zero
Probabilistic Logics
CSL

- Temporal logic for describing properties of CTMCs
- Key addition:
  - probabilistic operator $P$
  - steady state operator $S$

\[
\begin{align*}
\Phi & ::= \text{true} \mid a \mid \neg \Phi \mid \Phi_1 \land \Phi_2 \mid P_p[\psi] \mid S_p[\Phi] \quad \text{(State formulae)} \\
\psi & ::= X\Phi \mid \Phi_1 \cup' \Phi_2 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{(Path formulae)}
\end{align*}
\]

\[s \models S_p[\Phi] \quad \text{iff} \quad \sum_{s' \models \Phi} \pi_s(s') \sim p,\]

where $\pi_s(s')$ is the probability of, starting in state $s$, being in state $s'$ in the long run.
Probabilistic Logics

CSL

- Examples:

- $S_{\geq 0.9}[Operational]$: The probability of being in the operational state in the long run is at least 0.9.

- $degraded \rightarrow$

  $P_{>0.99}[Degraded \cup [0,2] Operational]$: When the system is in the degraded state, the probability of changing to full operational state without leaving degraded state within 2 hours is greater than 0.99.
PRISM

- One of the important **probabilistic** model checkers
  - Developed by Oxford University
  - Supports DTMCs, MDPs, CTMCs, PTAs
  - Supports PCTL, CSL, LTL + reward extensions
  - Symbolic model checking techniques
  - GUI + Command line
PRISM
Modeling Language

- Fundamental components
  - Modules: Consisting variables
  - Variables: specifying states of a module
- Model:
  - Consisting of some module interacting with each other
  - Combination of the states of modules determines the global state of the model
- Commands: Behavior of the model

[action] guard -> prob_1: update_1 + ... + prob_N: update_N
PRISM Modeling Language

- DTMC

dtmc
module M1
  S : [0..3] init 0;
  [] s=0 -> (s'=1);
  [] s=1 -> 0.3: (s'=2)+0.7: (s'=3);
  [] s=2 -> (s'=0);
  [] s=3 -> (s'=3);
endmodule
PRISM Modeling Language

- MDP

```plaintext
mdp
module M1
    s : [0..4] init 0;
    [] s=0 -> (s'=1);
    [] s=1 -> 0.3:(s'=2)+0.7:(s'=3);
    [resend] s=2 -> (s'=0);
    [stop] s=2 -> (s'=4);
    [] s=3 -> (s'=3);
    [] s=4 -> (s'=4);
endmodule
```
PRISM Modeling Language

- CTMC

ctmc
cst int N = 3;
cst double mu = ;
cst double lambda = ;
module queue
    q : [0..N];
    [] q<N -> lambda: (q'=q+1);
    [] q>0 -> mu: (q'=q-1);
endmodule
Performance and Dependability Evaluation

- **Performance**: How well a system meets the expectations of its users?
  - Performance measures: response time, queue length, utilization, throughput
  - Ignores failure and repair

- **Dependability**: How much the service is reliable
  - Dependability measures: availability, reliability
  - Focuses on failure and repair

- **Performability**: Performance + Dependability
  - How well a system meets the expectations of its users in the presence of failure and repair?
Performance and Dependability Evaluation (Cont.)

- **Utilization** (in steady-state), $U$: The probability that the server is busy in the long-run.
- **Reliability**, $R(t)$: The probability that the system does not fail in time period $[0,t]$.
- **Instantaneous availability**, $a(t)$: The probability that the system works as expected in time $t$.
- **Steady-state availability**, $A$: The probability that the system works as expected in the long-run.
Performance and dependability measures can be specified using probabilistic logics.

For example, in PRISM:

- **Utilization**: $S \geq 0.9[\text{“Server.Busy”}]$
- **Reliability, $R(t)$**: $P \geq 0.9[ \ G \ t \ ! \ “fail” ]$
- **Instantaneous availability, $a(t)$**: $P \geq 0.99 \ [ G[t,t] \ “up” ]$
- **Steady-state availability, $A$**: $S \geq 0.99 \ [“up”]$
Probabilistic Model Checking for Performance evaluation

- **Advantages**
  - Stating requirements unambiguously
    - It is impossible to consider more than one interpretation for a specification.
  - Flexibility
    - By nesting operators in probabilistic logics, we can define very complex requirements
  - Single algorithm, many different measures
    - No need to develop new methods for new performance or dependability measures.
  - Specifying requirements over the selected paths of the model
    - Not easily possible in traditional performance and dependability evaluation
Challenges

- State space explosion problem
  - More severe than non-probabilistic model checking, as it includes:
    - Model checking algorithms
    - Performance evaluation computations (e.g. linear equation solving)
- Generating and representing the counterexample
  - For many properties, one trace is not enough
    - e.g. probability of a failure is less than 0.1.
Thank You

Questions?