Probabilistic Systems (Lecture 21)

Analysis of Software Artifacts

Outline

- Markov chains: notions and analysis
- Probabilistic CTL
- Model Checking for PCTL
- Probabilistic Verus and symbolic model checking with MTBDDs

Probabilistic Systems

Stochastic systems: events occur with certain probabilities.

Examples: manufacturing systems, controllers, environment, etc.

Properties of interest

- steady-state probability
- mean time to failure
- average lifetime
- reliability
- probability of reaching a state after a given time

Stochastic and Markov Processes

discrete random variable: probability mass function
$p(x) = P[X = x]$

continuous rnd. var.: cumulative distribution function
$F(x) = P[X \leq x]$

$p(x) \in [0,1]; \sum_{x \in D} p(x) = 1;$ $F(x) \in [0,1];$
$F(-\infty) = 0; F(+\infty) = 1$

stochastic process = collection of rnd. var. indexed by time $t \in \mathcal{T}$

Stochastic and Markov Processes

discrete-time process ($\mathcal{T} = \mathbb{N}$) or *continuous-time* process ($\mathcal{T} = \mathbb{R}$)

$\{X(t)\}$ is a *Markov process* iff is has the *memoryless* property:
future probabilistically determined by present, independent of past

$$P[X(t_{k+1}) \leq x_{k+1} | X(t_k) = x_k, \ldots, X(t_0) = x_0] = P[X(t_{k+1}) \leq x_{k+1} | X(t_k) = x_k]$$

For discrete state space: *Markov chain*:

$$P[X(t_{k+1}) = x_{k+1} | X(t_k) = x_k, \ldots, X(t_0) = x_0] = P[X(t_{k+1}) = x_{k+1} | X(t_k) = x_k]$$

Transition Probabilities

Consider: discrete-time Markov chain, with integers as state space.

Define: $p_{ij}(k) = P[X_{k+1} = j \mid X_k = i]$. Then, $\sum_j p_{ij}(k) = 1$.

n-step transition probability: $p_{ij}(k, k+n) = P[X_{k+n} = j \mid X_k = i]$.

$p_{ij}(k, k+n) = \sum_{r \in D} p_{ir}(k, u) p_{rj}(u, k+n)$, $k \leq u \leq k+n$ (Chapman-Kolmogorov equation)

Transition Probabilities

homogeneous Markov chain if $p_{ij}(k)$ independent of $k \Rightarrow$ simply p_{ij}

Define *transition probability matrix* $T = [p_{ij}]$

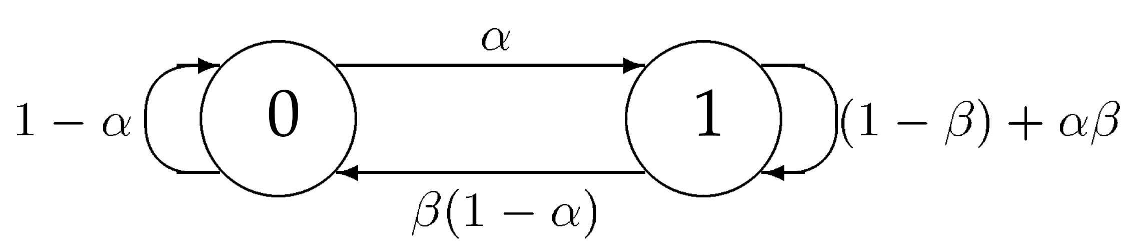
All rows sum up to 1: *stochastic* matrix.

Example: A Simple Markov Chain

Telephone call process with discrete time slots:

- at most one call in one slot; probability α
- calls complete with probability β
- if phone busy, call is lost
- process call arrival if it
occurs at time of call completion

State transition diagram



Analyzing Markov Chains

Classification of states:

- state j reachable from i if $p_{ij}(k) > 0$ for some k.
- state set S *closed* iff $p_{ij} = 0 \; \forall i \in S, j \notin S$
- state is *absorbing* iff $p_{ii} = 1$
- *irreducible* closed set if any state
reachable from any other

Analyzing Markov Chains

- *recurrent* state if probability of returning to it is 1
- *transient* state if not recurrent
- *positive recurrent* state if mean recurrence time $< \infty$
- positive recurrent state: *periodic* or *aperiodic*

$\Rightarrow$ with these notions, can analyze *steady-state behavior*

Steady-State Analysis

- state probability $\pi_j(k) = P[X_k = j]$
- state probability vector $\pi(k) = [\pi_0(k), \pi_1(k), \ldots]$

Question: does $\lim_{k\to\infty} \pi(k)$ exist ?

Theorem: In an irreducible aperiodic Markov chain, the limit $\lim_{k\to\infty} \pi(k)$ always exists and is independent of the initial state probability vector.

Temporal Logic Analysis: Computation Model

Computation model: labelled Markov chain (S, s^0, T, μ) where:

- S = finite state space
- s^0 = initial state
- $T : S \times S \rightarrow [0, 1]$ = transition relation with $\sum_{s' \in S} T(s, s') = 1$

Temporal Logic Analysis: Computation Model

- $\mu : S \to 2^{AP}$ = labeling function (AP = set of atomic propositions)

T induces probability measure on set of execution traces.

Probabilistic CTL

[Hansson and Jonsson, RTSS89]

Syntax: State formulas sf and path formulas pf.

$$sf ::= prop \mid \neg sf \mid sf_1 \wedge sf_2 \mid [pf]_{\geq p} \mid [pf]_{>p}$$

$$pf ::= sf_1 \mathbf{U}^{\leq t} sf_2 \mid sf_1 \mathcal{U}^{\leq t} sf_2$$

strong until   weak until

Probabilistic CTL

Semantics:

- $\sigma \models sf_1 \mathbf{U}^{\leq t} sf_2$ iff $\exists i \leq t . \sigma[i] \models sf_2$ and $\forall j \in [0, i] . \sigma[j] \models sf_1$
- $\sigma \models sf_1 \mathcal{U}^{\leq t} sf_2$ iff $\sigma \models sf_1 \mathbf{U}^{\leq t} sf_2$ or $\forall j \in [0, t] . \sigma[j] \models sf_1$
- $s \models [pf]_{>p}$ iff prob. measure of paths that satisfy pf exceeds p

Properties in PCTL

- $\mathbf{F}^{\leq t}_{\geq p} sf = true \mathbf{U}^{\leq t}_{\geq p} sf$
- $\mathbf{G}^{\leq t}_{\geq p} sf = \neg \mathbf{F}^{\leq t}_{> 1-p} \neg sf$

Examples:

- $req \mathbf{U}^{\leq t}_{\geq p} ack$: there is probability at least p that there is an *ack* within t units and that *req* stays true until *ack* becomes true
- $\mathbf{G}^{\leq t}_{\geq p} fail$: there is no failure for t time units with probability at least p
- $\mathbf{F}^{\leq t}_{\geq p} alarm$: an alarm occurs with probability at least p

within time t

Model Checking for PCTL: Until Operator

Probability of a path from s satisfying $f_1 \mathbf{U}^{\leq t} f_2$ defined inductively:

- $p(s, 0)$ = if $s \models f_2$ then 1 else 0
- $p(s, t)$ = if $s \models f_2$ then 1 else if $s \not\models f_1$ then 0 `/* directly */`

  else $\sum_{s' \in S} T(s, s')p(s', t-1)$ `/* recursive */`

Model Checking

- build transition probability matrix P from T as follows:
- $P[s_k, s_l] = T[s_k, s_l]$ if $s_k \models f_1 \wedge s_k \not\models f_2$ `/* recursive case */`
- $P[s_k, s_l] = 1$ if $\neg(s_k \models f_1 \wedge s_k \not\models f_2) \wedge (k = l)$ `/* keep same */`
- $P[s_k, s_l] = 0$ otherwise
- compute vector $\bar{p}(s, t) = P^t \bar{p}(s, 0)$
- $M \models f_1 \mathbf{U}^{\leq t}_{\geq p} f_2$ iff $\bar{p}(s^0, t) \geq p$

Probabilistic Verus

[Hartonas-Garmhausen, Campos, Clarke 1998]

- introduce probabilistic selection statement:

$\texttt{pselect}(p_1 : stmt_1; \ldots p_m : stmt_m)$ with $p_i \in [0,1]$, $\sum_{i=1}^{m} p_i = 1$

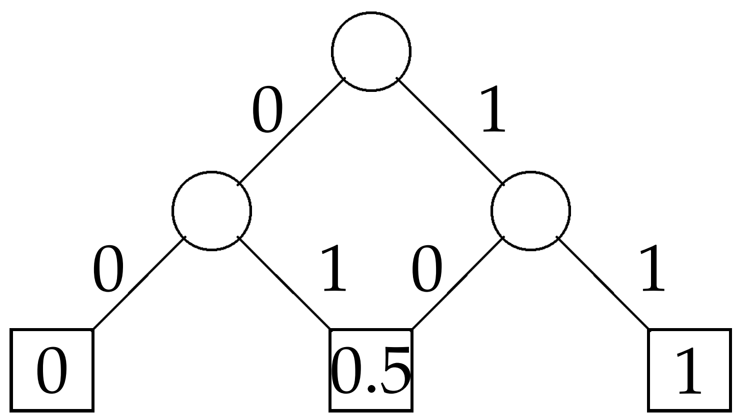
- compilation of program yields probabilistic transition matrix
- implemented symbolically using MTBDDs (multi-terminal BDDs)
- fully symbolic efficient representation and algorithms

Multi-Terminal BDDs

- MTBDDs are binary decision diagrams with arbitrary values
(from a finite set) in the terminal nodes
- matrix representation: BDD variables are row and column indices;
terminal values are matrix elements
- if matrix is sparse or presents regularity $\Rightarrow$ compact representation

Multi-Terminal BDDs

- matrix operation implemented recursively using BDD *apply* operator



Practical Results with ProbVerus

- manufacturing systems: verify downtime probabilities
- also transportation; fault-tolerant industrial process control
- largest example: a safety-critical system for railway stations
- safety, liveness, response times, probabilities of reaching unsafe states

Practical Results with ProbVerus

- complexity: 10^{27} states; about 5 minutes per specification
- deadlock discovered and located through counterexample trace