DD2552 - Seminars on Theoretical Computer Science, Programming Languages and Formal Methods, Seminar 1

Karl Palmskog (palmskog@kth.se)

2021-08-30

Course Contents

- course run is about specification and verification of systems with stochastic behavior
- includes specification **formalisms** for, and mathematical **models** of, stochastic systems
- includes **methods** and **tools** for verifying model properties
- for large systems with complex behavior, we leverage **statistical** approaches (sampling)
- most areas covered are highly active research topics!

Course Overview

- highly research-based course for 7.5 ECTS credits
- assumes knowledge of logic, programming, and probability
- seminars twice every week in period 1
  - Mondays 13-15
  - Thursday 10-12
- all seminars take place in room 4523

Course Examination

- examination is by:
  - homework set (graded, determines course grade)
- seminar participation *strongly* recommended
- seminars driven by topic and research papers ...
- ... but the aim is to have lots of interaction

About Me

- M.Sc. Computer Science and Engineering, KTH, 2007
- Ph.D. Computer Science, KTH, 2014
- researcher at U. of Illinois and U. of Texas until 2019
- researcher at KTH since 2019
- main interests:
  - formal verification using proof assistants (Coq, HOL4)
  - programming language metatheory
  - distributed systems
- https://setoid.com

Randomness in Computer Science and Engineering

- domain is inherently random (e.g., networks, biological systems)
  - random arrival of requests
  - random interaction among actors
- processes execute a randomized algorithm
  - flip coin to determine next action
  - run random function on whether to act (blockchains)
- requirement: behavior captured by **probability distribution**

Functional and Non-functional Requirements

- system requirements can be *functional* or *non-functional*
- functional: what is a system supposed to **do**?
  - e.g., specification of output given knowledge of input
  - "if x is nonnegative, then the output is a prime larger than x"
  - studied in other courses
- non-functional: what is a system supposed to **be** like?
  - e.g., not leak confidential information
  - e.g., provide an answer within a time limit
  - "every received request is answered within t seconds"
  - focus of this course

Non-functional Requirements Mostly Outside Scope

- security & privacy properties
- *hard* temporal constraints, e.g., WCET
- asymptotic behavior, such as execution time growth with input

Non-functional requirements we want to consider

- what happens in the *typical* or *average* case?
- how low is the chance of a crash?
- how high is the chance of responding quickly to a request?

"within time t, the probability that the number of messages in the queue q will be greater than 5 is less than 0.01"

"within time t, if a network node crashes, the probability that it will recover within 5000 steps is between 0.9 and 0.99"

General Approach in This Course

- specify the desired property as a formula ϕ in a logic
- consider a model $\mathcal{M}$ of the system
- determine (using deduction/algorithm/tool) whether $\mathcal{M}$ satisfies ϕ, i.e., whether $\mathcal{M} \models \phi$
- problem: logic may have to be extremely expressive
- problem: system can have large state space
- problem: system can be inaccessible(!)

Course Trajectory

- stochastic logics: PCTL, QuaTex, ...
- models: DTMC, CTMC, ...
- verification: deductive, symbolic, statistical, ...
- tools: PRISM, Ymer, UPPAAL, ...

Main course literature:

`Gul Agha and Karl Palmskog`
`A Survey of Statistical Model Checking`
`TOMACS, 28(1):6:1-6:39, January 2018`

https://doi.org/10.1145/3158668

Reminder: CTL, a Non-Stochastic Temporal Logic

$$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi$$
$$\psi ::= \phi \mid X\,\phi \mid \phi\,U\,\phi$$

- a: atomic proposition
- $\neg\phi$: negation
- $\phi \wedge \phi'$: conjunction
- $X\,\phi$: next
- $\phi\,U\,\phi'$: until

Reminder: Non-Stochastic Transition Systems

$\mathcal{M} = (S, \to, L)$ where

- S is a (finite) set of states
- $\to \subseteq S \times S$
- $L \colon S \mapsto 2^{\mathrm{AP}}$ (AP are atomic propositions)

Transitions from state to state are "taken" non-deterministically.

Reminder: CTL Model Checking Problem

$$M, s \models \phi$$

- can be determined using CTL's semantics (tedious, no termination guarantee)
- can be determined using an efficient CTL model checking algorithm (but PSPACE-hard problem)

CTL with Bounded Until

$$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi$$

$$\psi ::= \phi \mid X\,\phi \mid \phi\,U\,\phi \mid \phi\,U^{\leq t}\,\phi$$

$$t \in Z^{\geq 0}$$

CTL with Bounded Until Example

Consider formula $\phi \; U^{\leq t} \; \phi'$ and path starting with state s where it holds:

$$\begin{array}{ccccccc} s & s_1 & \ldots & s_k & s_{k+1} & \ldots & s_t \\ \phi & \phi & \ldots & \phi & \phi' & \ldots & \top \end{array}$$

Probabilistic CTL

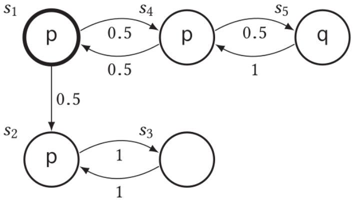
$$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi \mid P_{\geq\theta}(\psi)$$
$$\psi ::= \phi \mid X\,\phi \mid \phi\,U\,\phi \mid \phi\,U^{\leq t}\,\phi$$
$$t \in Z^{\geq 0} \quad \theta \in [0,1]$$

Reminder: Discrete Time Markov Chains

$\mathcal{M} = (S, s_i, M, L)$ where

- S is a (finite) set of states
- $s_i \in S$ is the initial state
- $M\colon S \times S \mapsto [0,1]$ defines transition probabilities, where
  - for all $s \in S$, $\sum_{s' \in S} M(s, s') = 1$
- $L\colon S \mapsto 2^{\mathrm{AP}}$ (AP are atomic propositions)

Example Discrete Time Markov Chain



M	s_1	s_2	s_3	s_4	s_5
s_1	0	0.5	0	0.5	0
s_2	0	0	1	0	0
s_3	0	1	0	0	0
s_4	0.5	0	0	0	0.5
s_5	0	0	0	1	0

- $S = \{s_1, s_2, s_3, s_4, s_5\}$
- $L(s_1) = \{\mathrm{p}\}$
- $L(s_2) = \{\mathrm{p}\}$
- $L(s_3) = \{\}$
- $L(s_4) = \{\mathrm{p}\}$
- $L(s_5) = \{\mathrm{q}\}$

Discrete Time Markov Chains and PCTL

Slogan: "$P_{\geq\theta}(\psi)$ is true in s when the probability that ψ holds on paths starting from s is greater than or equal to θ"

Example formula:

$$P_{\geq 0.98}(\texttt{pending}\, U^{\leq 10}\, \texttt{done})$$

Intuition: "With probability 0.98 or more, `pending` holds until `done` holds within 10 steps"