# Seminar on Probabilistic Programs WS 14/15
# Kick–off Meeting

Joost-Pieter Katoen  Benjamin Kaminski  Nils Jansen
Federico Olmedo

Rheinisch–Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 2

16.10.2014

# Motivation

## Probabilistic Programs

# Motivation

## Probabilistic Programs

- Like ordinary programs, except:
  - Program may toss a (possibly unfair) coin
  - Further execution of the program is influenced by outcome of the coin toss

# Motivation

## Probabilistic Programs

- Like ordinary programs, except:
  - Program may toss a (possibly unfair) coin
  - Further execution of the program is influenced by outcome of the coin toss
- Applications in:
  - Randomized algorithms
  - Security
  - Machine learning

# Aims of this Seminar

## Goals to Achieve

# Aims of this Seminar

## Goals to Achieve

- Independently understand a scientific topic

# Aims of this Seminar

## Goals to Achieve

- Independently understand a scientific topic
- Acquire, read, and understand suitable scientific literature

# Aims of this Seminar

## Goals to Achieve

- Independently understand a scientific topic
- Acquire, read, and understand suitable scientific literature
- Write of your own report on your topic

# Aims of this Seminar

## Goals to Achieve

- Independently understand a scientific topic
- Acquire, read, and understand suitable scientific literature
- Write of your own report on your topic
- Give a didactically appealing oral presentation on your topic

# Aims of this Seminar

## Demands on Your Report

# Aims of this Seminar

## Demands on Your Report

- Independently written report of 14 – 16 pages

# Aims of this Seminar

## Demands on Your Report

- Independently written report of 14 – 16 pages
- Normal font size (11pt or 12pt) with "normal" page layout

# Aims of this Seminar

## Demands on Your Report

- Independently written report of 14 – 16 pages
- Normal font size (11pt or 12pt) with "normal" page layout
- Language: English (german only if you are doing this seminar for your bachelor studies)

# Aims of this Seminar

## Demands on Your Report

- Independently written report of 14 – 16 pages
- Normal font size (11pt or 12pt) with "normal" page layout
- Language: English (german only if you are doing this seminar for your bachelor studies)
- Correct spelling and grammar. More than 10 errors per page lead to abortion of correction

# Aims of this Seminar

## Demands on Your Report

- Independently written report of 14 – 16 pages
- Normal font size (11pt or 12pt) with "normal" page layout
- Language: English (german only if you are doing this seminar for your bachelor studies)
- Correct spelling and grammar. More than 10 errors per page lead to abortion of correction
- Complete list of references to all consulted literature
- Correct citation of important literature. Copying or slightly modifying text blocks without citation constitutes plagiarism and causes immediate exclusion from this seminar!

# Aims of this Seminar

## Demands on Your Talk

# Aims of this Seminar

## Demands on Your Talk

- Talk duration: about 40 minutes. Overtime is bad!

# Aims of this Seminar

## Demands on Your Talk

- Talk duration: about 40 minutes. Overtime is bad!
- Language: English (german only if you are doing this seminar for your bachelor studies)

# Aims of this Seminar

## Demands on Your Talk

- Talk duration: about 40 minutes. Overtime is bad!
- Language: English (german only if you are doing this seminar for your bachelor studies)
- Focus you talk on the audience

# Aims of this Seminar

## Demands on Your Talk

- Talk duration: about 40 minutes. Overtime is bad!
- Language: English (german only if you are doing this seminar for your bachelor studies)
- Focus you talk on the audience
- Prepare descriptive slides:
  - use less than 15 lines of text
  - use (base) colors in a useful manner

# Aims of this Seminar

## Demands on Your Talk

- Talk duration: about 40 minutes. Overtime is bad!
- Language: English (german only if you are doing this seminar for your bachelor studies)
- Focus you talk on the audience
- Prepare descriptive slides:
  - use less than 15 lines of text
  - use (base) colors in a useful manner
- Include slide numbers

# Aims of this Seminar

## Demands on Your Talk

- Talk duration: about 40 minutes. Overtime is bad!
- Language: English (german only if you are doing this seminar for your bachelor studies)
- Focus you talk on the audience
- Prepare descriptive slides:
  - use less than 15 lines of text
  - use (base) colors in a useful manner
- Include slide numbers
- Ask and prepare yourself for questions

# Aims of this Seminar

## Preparation of Your Talk

1

# Aims of this Seminar

## Preparation of Your Talk

- Either send in your slides as PDF or bring your own laptop.

1

# Aims of this Seminar

## Preparation of Your Talk

- Either send in your slides as PDF or bring your own laptop.
  - If you do decide to bring your own laptop, you need to have one of the following connectors available[1]:
    - VGA
    - HDMI (speak to your supervisor a few days in advance)
    - Mac Displayport

[1] We are so strict about this because we have had problems with this in the past.

# Aims of this Seminar

## Preparation of Your Talk

- Either send in your slides as PDF or bring your own laptop.
  - If you do decide to bring your own laptop, you need to have one of the following connectors available[1]:
    - VGA
    - HDMI (speak to your supervisor a few days in advance)
    - Mac Displayport
- Bring multiple copies of your slides (laptop, USB, web, email slides to supervisor, etc.)

[1] We are so strict about this because we have had problems with this in the past.

# The Timeline

## Important Dates/Deadlines

# The Timeline

## Important Dates/Deadlines

| | |
|---|---|
| tomorrow | Obtain required literature and get familiar with it |
| only until 6.11.2014 | Drop out without consequences |
| 17.11.2014 | Table of contents due |
| 15.12.2014 | Preliminary version of report due |
| 5.01.2015 | Final version of report due |
| 19.01.2015 | Preliminary version of slides due |
| 26.01.2015 | Final version of slides due |
| 4.02.2015 | Seminar Talks |
| 5.02.2015 | Seminar Talks (cont'd) |

You can also find these dates on the seminar's website!

# The Timeline

## Important Dates/Deadlines

| | |
|---|---|
| tomorrow | Obtain required literature and get familiar with it |
| only until 6.11.2014 | Drop out without consequences |
| 17.11.2014 | Table of contents due |
| 15.12.2014 | Preliminary version of report due |
| 5.01.2015 | Final version of report due |
| 19.01.2015 | Preliminary version of slides due |
| 26.01.2015 | Final version of slides due |
| 4.02.2015 | Seminar Talks |
| 5.02.2015 | Seminar Talks (cont'd) |

You can also find these dates on the seminar's website!

All deadlines are firm!

Missing a deadline causes immediate exclusion from this seminar!

# Topic Selection Procedure

# Topic Selection Procedure

1. We hand out a sheet with list of topics with a topic number

# Topic Selection Procedure

1. We hand out a sheet with list of topics with a topic number
2. We give a short presentation of the topics

# Topic Selection Procedure

1. We hand out a sheet with list of topics with a topic number
2. We give a short presentation of the topics
3. You indicate your name, your language preferences, and your three priorities for topics on that sheet

# Topic Selection Procedure

1. We hand out a sheet with list of topics with a topic number
2. We give a short presentation of the topics
3. You indicate your name, your language preferences, and your three priorities for topics on that sheet
4. We do our best to find a good student–topic–matching (Disclaimer: no guarantee for an optimal solution)

# Topic Selection Procedure

1. We hand out a sheet with list of topics with a topic number
2. We give a short presentation of the topics
3. You indicate your name, your language preferences, and your three priorities for topics on that sheet
4. We do our best to find a good student–topic–matching (Disclaimer: no guarantee for an optimal solution)
5. We announce the matching ASAP on the seminar's website. There you will also find out who your supervisor is

# Topic Selection Procedure

1. We hand out a sheet with list of topics with a topic number
2. We give a short presentation of the topics
3. You indicate your name, your language preferences, and your three priorities for topics on that sheet
4. We do our best to find a good student–topic–matching (Disclaimer: no guarantee for an optimal solution)
5. We announce the matching ASAP on the seminar's website. There you will also find out who your supervisor is
6. You contact your supervisor to get things started

# Stand by while your sheet is being handed out to you...

# The Topics

# 1: Expressing and Verifying Probabilistic Assertions

# 1: Expressing and Verifying Probabilistic Assertions

- Traditional invariants are predicates whose validity is not changed by one loop iteration
  - **assert** $e$
  - Property $e$ must hold on every program execution

# 1: Expressing and Verifying Probabilistic Assertions

- Traditional invariants are predicates whose validity is not changed by one loop iteration
  - **assert** $e$
  - Property $e$ must hold on every program execution
- Probabilistic programs have probabilistic outcomes — traditional assertions not suitable

# 1: Expressing and Verifying Probabilistic Assertions

- Traditional invariants are predicates whose validity is not changed by one loop iteration
  - **assert** $e$
  - Property $e$ must hold on every program execution
- Probabilistic programs have probabilistic outcomes — traditional assertions not suitable
- More appropriate concept: probabilistic assertions:
  - **passert** $e$, $p$, $c$
  - Property $e$ must hold with probability $p$ at confidence $c$

# 1: Expressing and Verifying Probabilistic Assertions

- Traditional invariants are predicates whose validity is not changed by one loop iteration
  - **assert** $e$
  - Property $e$ must hold on every program execution
- Probabilistic programs have probabilistic outcomes — traditional assertions not suitable
- More appropriate concept: probabilistic assertions:
  - **passert** $e$, $p$, $c$
  - Property $e$ must hold with probability $p$ at confidence $c$
- Paper presents probabilistic evaluation scheme to efficiently verify probabilistic assertions

# 2: Verifying Prob. Programs Using a Hoare Like Logic

# 2: Verifying Prob. Programs Using a Hoare Like Logic

- Traditional Hoare Logic
  - Hoare triple $\{E\}\ P\ \{F\}$ is valid iff...

# 2: Verifying Prob. Programs Using a Hoare Like Logic

- Traditional Hoare Logic
  - Hoare triple $\{E\}\ P\ \{F\}$ is valid iff...
  - ...predicate $E$ is a sufficient precondition such that after termination of program $P$ the predicate $F$ holds

# 2: Verifying Prob. Programs Using a Hoare Like Logic

- Traditional Hoare Logic
  - Hoare triple $\{E\}\ P\ \{F\}$ is valid iff...
  - ...predicate $E$ is a sufficient precondition such that after termination of program $P$ the predicate $F$ holds
  - Predicates here are expressed in first–order logic

# 2: Verifying Prob. Programs Using a Hoare Like Logic

- Traditional Hoare Logic
  - Hoare triple $\{E\}\ P\ \{F\}$ is valid iff...
  - ...predicate $E$ is a sufficient precondition such that after termination of program $P$ the predicate $F$ holds
  - Predicates here are expressed in first–order logic
- Approach of this paper:
  - Lift notion of first–order predicates to probabilistic predicates

# 2: Verifying Prob. Programs Using a Hoare Like Logic

- Traditional Hoare Logic
  - Hoare triple $\{E\}\ P\ \{F\}$ is valid iff...
  - ...predicate $E$ is a sufficient precondition such that after termination of program $P$ the predicate $F$ holds
  - Predicates here are expressed in first–order logic
- Approach of this paper:
  - Lift notion of first–order predicates to probabilistic predicates
  - Probabilistic predicates evaluate to some value in the interval $[0,\ 1]$ instead of **True** or **False**
  - Do Hoare–style reasoning on probabilistic programs

# 3: Proof Rules for Probabilistic Loops

# 3: Proof Rules for Probabilistic Loops

- Traditional programs:
  - Weakest precondition: **wp**.$P.E$

# 3: Proof Rules for Probabilistic Loops

- Traditional programs:
  - Weakest precondition: **wp**.$P$.$E$
  - $F =$ **wp**.$P$.$E$ if $F$ is the weakest precondition, such that starting in a program state satisfying $F$ the execution of the program $P$ terminates and ends in a state satisfying the postcondition $E$

# 3: Proof Rules for Probabilistic Loops

- Traditional programs:
  - Weakest precondition: **wp**.$P$.$E$
  - $F = \textbf{wp}.P.E$ if $F$ is the weakest precondition, such that starting in a program state satisfying $F$ the execution of the program $P$ terminates and ends in a state satisfying the postcondition $E$
- Probabilistic programs:
  - Weakest pre–expectation: **wp**.$P$.$f$

# 3: Proof Rules for Probabilistic Loops

- Traditional programs:
  - Weakest precondition: **wp**.$P$.$E$
  - $F = \textbf{wp}.P.E$ if $F$ is the weakest precondition, such that starting in a program state satisfying $F$ the execution of the program $P$ terminates and ends in a state satisfying the postcondition $E$
- Probabilistic programs:
  - Weakest pre–expectation: **wp**.$P$.$f$
  - $f$ is a random variable mapping program states to real numbers

# 3: Proof Rules for Probabilistic Loops

- Traditional programs:
  - Weakest precondition: $\mathbf{wp}.P.E$
  - $F = \mathbf{wp}.P.E$ if $F$ is the weakest precondition, such that starting in a program state satisfying $F$ the execution of the program $P$ terminates and ends in a state satisfying the postcondition $E$
- Probabilistic programs:
  - Weakest pre–expectation: $\mathbf{wp}.P.f$
  - $f$ is a random variable mapping program states to real numbers
  - $g = \mathbf{wp}.P.f$ if $g$ expresses the least expected value of the random variable $f$ after executing the program $P$

# 3: Proof Rules for Probabilistic Loops

- Traditional programs:
  - Weakest precondition: **wp**.$P.E$
  - $F = \textbf{wp}.P.E$ if $F$ is the weakest precondition, such that starting in a program state satisfying $F$ the execution of the program $P$ terminates and ends in a state satisfying the postcondition $E$
- Probabilistic programs:
  - Weakest pre–expectation: **wp**.$P.f$
  - $f$ is a random variable mapping program states to real numbers
  - $g = \textbf{wp}.P.f$ if $g$ expresses the least expected value of the random variable $f$ after executing the program $P$
- Paper presents proof rules for proving properties of probabilistic programs using weakest pre–expectation calculus

# 4: Expectation Invariants for Prob. Loops as Fixed Points

# 4: Expectation Invariants for Prob. Loops as Fixed Points

- Traditional programs:
  - Hoare–style logic over first order predicates
  - Use invariants to prove correctness

# 4: Expectation Invariants for Prob. Loops as Fixed Points

- Traditional programs:
  - Hoare–style logic over first order predicates
  - Use invariants to prove correctness
- Approach of this paper:
  - Use invariants involving expected values of program variables
    - Expectation Invariants

# 4: Expectation Invariants for Prob. Loops as Fixed Points

- Traditional programs:
  - Hoare–style logic over first order predicates
  - Use invariants to prove correctness
- Approach of this paper:
  - Use invariants involving expected values of program variables
    - Expectation Invariants
  - Express expectation invariants as fixed–points of the pre–expectation operator

# 4: Expectation Invariants for Prob. Loops as Fixed Points

- Traditional programs:
  - Hoare–style logic over first order predicates
  - Use invariants to prove correctness
- Approach of this paper:
  - Use invariants involving expected values of program variables
    - Expectation Invariants
  - Express expectation invariants as fixed–points of the pre–expectation operator
  - Use concepts from abstract interpretation for analyzing probabilistic programs by means of expectation invariants

# 5: Proving Termination of Prob. Programs Using Patterns

# 5: Proving Termination of Prob. Programs Using Patterns

- Traditional programs:
  - Either terminate or do not terminate
  - Termination problem is semi–decidable

# 5: Proving Termination of Prob. Programs Using Patterns

- Traditional programs:
  - Either terminate or do not terminate
  - Termination problem is semi–decidable
- Probabilistic programs:
  - Terminate with a certain probability

# 5: Proving Termination of Prob. Programs Using Patterns

- Traditional programs:
  - Either terminate or do not terminate
  - Termination problem is semi–decidable
- Probabilistic programs:
  - Terminate with a certain probability
  - Problem of almost–sure termination: Does the program terminate with probability 1?

# 5: Proving Termination of Prob. Programs Using Patterns

- Traditional programs:
  - Either terminate or do not terminate
  - Termination problem is semi–decidable
- Probabilistic programs:
  - Terminate with a certain probability
  - Problem of almost–sure termination: Does the program terminate with probability 1?
  - Almost–sure termination problem is harder to decide
  - Hence, more sophisticated approaches needed!

# 5: Proving Termination of Prob. Programs Using Patterns

- Traditional programs:
  - Either terminate or do not terminate
  - Termination problem is semi–decidable
- Probabilistic programs:
  - Terminate with a certain probability
  - Problem of almost–sure termination: Does the program terminate with probability 1?
  - Almost–sure termination problem is harder to decide
  - Hence, more sophisticated approaches needed!
- Paper presents a refinement–based approach for generating terminating patterns, sets of program runs with probability 1

# 6: Probabilistic Program Analysis with Martingales

# 6: Probabilistic Program Analysis with Martingales

- A sequence of random variables $M_0, M_1, \ldots$ is a martingale, if

$$\mathsf{E}(M_{n+1} \mid M_0, \ldots, M_n) = M_n$$

# 6: Probabilistic Program Analysis with Martingales

- A sequence of random variables $M_0, M_1, \ldots$ is a martingale, if

$$\mathsf{E}(M_{n+1} \mid M_0, \ldots, M_n) = M_n$$

- Almost corresponds to the Markov property

# 6: Probabilistic Program Analysis with Martingales

- A sequence of random variables $M_0, M_1, \ldots$ is a martingale, if

$$\mathsf{E}(M_{n+1} \mid M_0, \ldots, M_n) = M_n$$

- Almost corresponds to the Markov property
- $M_0, M_1, \ldots$ is a supermartingale, if

$$\mathsf{E}(M_{n+1} \mid M_0, \ldots, M_n) \leq M_n$$

# 6: Probabilistic Program Analysis with Martingales

- A sequence of random variables $M_0, M_1, \ldots$ is a martingale, if

$$\mathsf{E}(M_{n+1} \mid M_0, \ldots, M_n) = M_n$$

- Almost corresponds to the Markov property
- $M_0, M_1, \ldots$ is a supermartingale, if

$$\mathsf{E}(M_{n+1} \mid M_0, \ldots, M_n) \leq M_n$$

- Define supermartingale ranking functions for probabilistic programs

# 6: Probabilistic Program Analysis with Martingales

- A sequence of random variables $M_0, M_1, \ldots$ is a martingale, if

$$\mathsf{E}(M_{n+1} \mid M_0, \ldots, M_n) = M_n$$

- Almost corresponds to the Markov property
- $M_0, M_1, \ldots$ is a supermartingale, if

$$\mathsf{E}(M_{n+1} \mid M_0, \ldots, M_n) \leq M_n$$

- Define supermartingale ranking functions for probabilistic programs
- Paper uses known results on martingales to reason about properties of probabilistic programs, e.g. almost–sure termination

# 7: Slicing Probabilistic Programs

# 7: Slicing Probabilistic Programs

- A probabilistic program $P$ returns a distribution over return values

# 7: Slicing Probabilistic Programs

- A probabilistic program $P$ returns a distribution over return values
- Goal: Obtain a simpler program $Slice(P)$

# 7: Slicing Probabilistic Programs

- A probabilistic program $P$ returns a distribution over return values
- Goal: Obtain a simpler program $Slice(P)$
  - $Slice$ should be correct: Slicing should preserve the distribution over the return values

# 7: Slicing Probabilistic Programs

- A probabilistic program $P$ returns a distribution over return values
- Goal: Obtain a simpler program $\mathit{Slice}(P)$
  - $\mathit{Slice}$ should be correct: Slicing should preserve the distribution over the return values
  - $\mathit{Slice}$ should be efficient: Slicing should be done as fast as possible

# 7: Slicing Probabilistic Programs

- A probabilistic program $P$ returns a distribution over return values
- Goal: Obtain a simpler program $\mathit{Slice}(P)$
  - $\mathit{Slice}$ should be correct: Slicing should preserve the distribution over the return values
  - $\mathit{Slice}$ should be efficient: Slicing should be done as fast as possible
- Traditional program slicing techniques do not suffice for probabilistic programs as they are not correct in this setting

# 7: Slicing Probabilistic Programs

- A probabilistic program $P$ returns a distribution over return values
- Goal: Obtain a simpler program $Slice(P)$
  - $Slice$ should be correct: Slicing should preserve the distribution over the return values
  - $Slice$ should be efficient: Slicing should be done as fast as possible
- Traditional program slicing techniques do not suffice for probabilistic programs as they are not correct in this setting
- Paper presents correct and efficient approach for probabilistic program slicing

# 8: The Satisfiability Problem for Probabilistic CTL

# 8: The Satisfiability Problem for Probabilistic CTL

- Probabilistic CTL (PCTL) is a logic for reasoning about DTMCs

# 8: The Satisfiability Problem for Probabilistic CTL

- Probabilistic CTL (PCTL) is a logic for reasoning about DTMCs
- Probabilistic version of Computation Tree Logic (CTL)

# 8: The Satisfiability Problem for Probabilistic CTL

- Probabilistic CTL (PCTL) is a logic for reasoning about DTMCs
- Probabilistic version of Computation Tree Logic (CTL)
- Satisfiability problem for CTL is decidable

# 8: The Satisfiability Problem for Probabilistic CTL

- Probabilistic CTL (PCTL) is a logic for reasoning about DTMCs
- Probabilistic version of Computation Tree Logic (CTL)
- Satisfiability problem for CTL is decidable
- Still an open problem: Is the satisfiability problem for PCTL decidable?

# 8: The Satisfiability Problem for Probabilistic CTL

- Probabilistic CTL (PCTL) is a logic for reasoning about DTMCs
- Probabilistic version of Computation Tree Logic (CTL)
- Satisfiability problem for CTL is decidable
- Still an open problem: Is the satisfiability problem for PCTL decidable?
- Paper presents a partial solution in 2008: The satisfiability problem for the qualitative fragment of PCTL is decidable (ExpTime–complete)

# 8: The Satisfiability Problem for Probabilistic CTL

- Probabilistic CTL (PCTL) is a logic for reasoning about DTMCs
- Probabilistic version of Computation Tree Logic (CTL)
- Satisfiability problem for CTL is decidable
- Still an open problem: Is the satisfiability problem for PCTL decidable?
- Paper presents a partial solution in 2008: The satisfiability problem for the qualitative fragment of PCTL is decidable (ExpTime–complete)
- Gentle warning: The matter is a little difficult to digest. Good knowledge of DTMCs and PCTL is definitely recommended

Thank you for your attention!
Prioritize your favorite topics NOW!