Seminar on Probabilistic Programming (SS16)

— Introductory Meeting —

Christian Denhert Sebastian Junges Benjamin Kaminski
Joost-Pieter Katoen Christoph Matheja Federico Olmedo

RWTH Aachen University
Software Modeling and Verification Group

April 13th, 2016

Outline

Outline

Probabilistic Programs – Definition

What is a Probabilistic Program?

- An ordinary program,
  - imperative
  - functional
  - logical
- that allows for probabilistic choices,
  - randomly choose a process with which communicate
  - select a prime number in $[1..n^2]$
  - flip a (fair/biased) coin
- whose outcomes determine the program behaviour
  - input-output relation
  - runtime

Example

Probabilistic program that simulates a geometric distribution:

```
n := 0;
repeat
   n := n + 1;
   c := coin_flip(0.5)
until (c = heads);
return n
```

Relevance of Randomization

- Algorithms speed-up
  - probabilistic Quicksort
  - Rabin–Miller primality test
  - verification of matrix multiplication
- Solution to problems where deterministic techniques fail
  - dinning philosopher problems [Lehmann & Rabin '81]
  - leader election [Angluin '80]
- Multiple application domains
  - cryptography
  - communication
  - data mining
  - computer vision

Outline

Goals

Aims of this seminar

- Independent understanding of a scientific topic
- Preparation of your own report on this topic
- Oral presentation of the topic

Requirements on Report

Layout

- Length of $\approx$ 15 pages, with font size 12pt using "standard" page layout
- Use ordinary word processor or, preferably, LaTeX
  - LaTeX template available on seminar web page.
  - Recommended reading: The Not So Short Introduction to LaTeX 2ϵ, Tobias Oetiker
- Language: English (German only possible for bachelor students)

Requirements on Report

Content

- Independent writing (e.g. own examples)
    - Plagiarism: taking text blocks (from literature or web) without source indication causes immediate exclusion from this seminar
- Discuss content with your supervisor
- Include references to all consulted literature

Readability

- Correct usage of spelling and grammar
    - ≥ 10 errors per page $\Longrightarrow$ abortion of correction
- Clear, cohesive and concise
    - Recommended reading: Learn Technical Writing in Two Hours per Week, Norman Ramsey
- Formal language

Requirements on Talk

Talk

- Language: English (German only possible for bachelor students)
- Duration: 40 minutes (+5' of Q&A). Overtime is bad!
- Focus your talk on the audience
- Use descriptive (enumerated) slides:
  - ≤ 15 lines of text
  - use colors in a useful manner
- Ask and prepare yourself for questions. Prepare backup slides to anticipated questions
- Read this HowTo on Presentations before preparing the slides
- Discuss with your supervisor the possibility of a rehearsal

Requirements on Talk

Day of the Talk

- Either send slides as PDF to your supervisor or bring your own laptop. If so you must ensure you have one of the following connectors:
  - VGA
  - HDMI
  - Mac Displayport
- Students are required to attend all talks

Important Dates

Deadlines

23rd May	Report due
19th June	Slides due
29-30th June	Seminar

- Missing a deadline causes immediate exclusion from the seminar
- Report and slide hand-in is done by email to your supervisor

Outline

Selecting your Topic

Procedure

- We hand out a sheet with list of topics with topic number
- We give a short presentation of the topics
  - Topics 1–9: master students
  - Topics 10–13: bachelor students
- You indicate the preference of your topics (first, second, third).
- We do our best to find an adequate topic-student assignment. (Disclaimer: no guarantee for an optimal solution)
- Assignment will be published on website by 18th April
- You contact your supervisor to get things started

- **Problem:** Approximate the probability that a program establishes a given assertion ϕ.

- **Solution Overview:** Infer the whole program behaviour from finitely many executions
  - Choose (by program simulation) a *finite set of executions with overall high probability* (e.g. 0.95)
  - Compute the probability of ϕ within this set of executions using *symbolic execution*
  - Use this probability to give guaranteed bounds for the probability of ϕ in the whole program
  - Instead of computing exact probabilities, approximate them using *branch–and–bound techniques over polyhedra* (bounding boxes)

```
n := 0;
repeat
   n := n + 1;
   c := coin_flip(0.5)
until (c = heads);
return n
```

- **Problem:** establish *probabilistic assertions of loops* upon termination

$$x := 0;\ \texttt{while}\ (|x| < 100)\ \{x := x{+}1\ \ [1/2]\ \ x{-}1\}$$

Show that upon loop termination, $Pr[x = 100] = Pr[x = -100]$.

- **Solution Overview:** synthesise a *martingale* and apply the optinal stopping theorem (OST)

X_n: value of x after n-th iteration τ: loop stopping time

$$\mathbb{E}[X_{n+1}|X_n, .., X_0] = \tfrac{1}{2}(X_n{+}1) + \tfrac{1}{2}(X_n{-}1) = X_n$$ ◀ X_n matingale

$$\mathbb{E}[X_\tau] = \mathbb{E}[X_0]$$ ◀ OST

$$100 \cdot Pr[x = 100] + (-100) \cdot Pr[x = -100] = 0$$

- **Problem:** Establish *"relational" properties* between pair of probabilistic processes

```
n1 := 0;                                   n2 := 0;
for (i1 := 0; i1 < k; i1++)                for (i2 := 0; i2 < k; i2++)
  b1 := p1 · ⟨tt⟩ + (1−p1) · ⟨ff⟩;           b2 := p2 · ⟨tt⟩ + (1−p2) · ⟨ff⟩;
  if (b1) then n1 ++;                        if (b2) then n2 ++;
return n1;                                 return n2;
```

$$p_1 \geq p_2 \implies \forall a.\ \mathtt{Pr}[n_1 \geq a] \geq \mathtt{Pr}[n_2 \geq a]$$

- **Overview solution:** exploit *connection* between *probabilistic couplings* and *relational Hoare logic*
  - Translate a proof argument based on couplings into a deductive argument using (relational) Hoare logic

- **Problem:** Bound the probability that a program fails to satisfy its specification
- **Solution Overview:** Use a probabilistic Hoare logic, where triples are augmented with the failure probability

$$\vdash_\beta \{P\}\, c\, \{Q\}$$

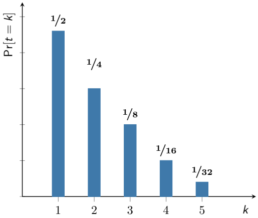
c:	probabilistic program
P/Q:	pre–/post–condition
β:	bound on the probability of failing to establish Q

  - Proof system
  - Application: verification of accuracy for differential privacy mechanisms

- **Problem:** Determine the *average runtime* of a probabilistic program.

```
repeat
  {b := heads} [1/2] {b := tail};
until (b := heads)
```

$$\mathbb{E}[t] = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + \cdots = \mathbf{2}$$



- **Solution Overview:** Use a *continuation passing style* through runtime transformer

$$\mathsf{ert}[c] : \mathbb{T} \to \mathbb{T}$$

t is the runtime of the computation following c $\Longrightarrow$ $\mathsf{ert}[c](t)$ is the runtime of c, plus the computation following c

6: Game-Based Abstraction Refinement

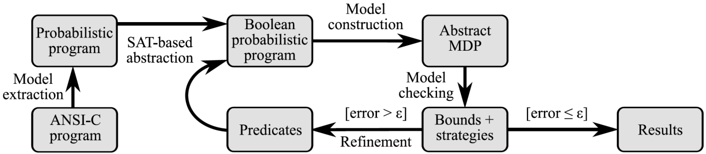
```
    bool fail = false;
    int c = 0;

    int main() {
1:    c = num_to_send();
2:    while (!fail && c>0) {
3:       fail = send_msg();
4:       c--;
      }
5:    assert(!fail);
    }
    int num_to_send() {
      return ndet(3);
    }
    bool send_msg() {
      return (coin(0.1)==1);
    }
```

input: ANSI-C program with probabilistic features

output: probability to reach certain states

approach: start with coarse overapproximation and refine as necessary



- PCTL: The probabilistic version of Computation Tree Logic
- PCTL* model-checking well-studied for *finite-state* MDPs
- This paper: Develop a *proof system* to verify PCTL* properties for *countable-state* systems

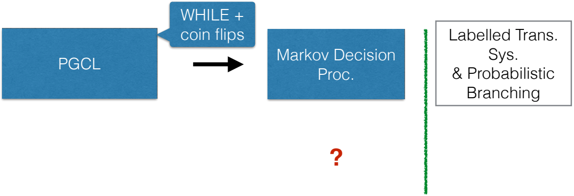
probabilistic program P

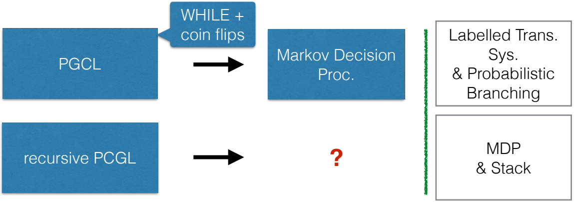
$P \vdash \Phi$

PCTL* formula Φ

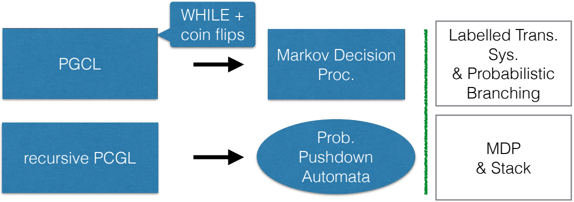
- *Soundness*: $P \vdash \Phi$ implies $P \models \Phi$
- *Completeness* for finite-state systems: $P \models \Phi$ implies $P \vdash \Phi$

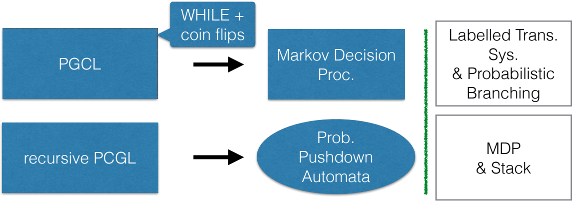
8: Analysis of Probabilistic Pushdown Automata




PGCL
WHILE +
coin flips
Markov Decision
Proc.
Labelled Trans.
Sys.
& Probabilistic
Branching
recursive PCGL
?
MDP
& Stack

8: Analysis of Probabilistic Pushdown Automata





- Probability of reaching a state (with an empty stack) ?
  - with probability 1 ?
  - with probability p ?
- Expected number of steps until reaching a state ?
  - finite?
  - bounded?

.....

- KAT = **K**leene **A**lgebra with **T**ests
  - Algebraic approach to program verification
  - Enables equational reasoning about program equivalence
  - E.g. Kleene Normal Form Theorem (one while loop suffices) provable by purely algebraic means without knowing anything about the program using KAT
- NetKAT extends KAT by communication primitives
- Probabilistic NetKAT extends NetKAT by probability measures on communication histories

10: Symmetry Reduction

```
nondeterministic
module process1
  s1 : [0..2] init 2;
  [] s1=2 -> 1:(s1'=0);
  [] s1=2 -> 1:(s1'=1);
  [] s1=0 & (s1!=2 & s2!=2 & s3!=2) & (s1=0 & s2=0 & s3=0) ->
                                 0.5:(s1'=0) + 0.5:(s1'=1);
  [] s1=0 & (s1!=2 & s2!=2 & s3!=2) & (s2=1 | s3=1) -> 1:(s1'=0);
  [] s1=1 & (s1!=2 & s2!=2 & s3!=2) & (s2=1 | s3=1) ->
                                 0.5:(s1'=0) + 0.5:(s1'=1);
  [] s1=1 & (s1!=2 & s2!=2 & s3!=2) & (s2=0 & s3=0) -> 1:(s1'=1);
endmodule

module process2 = process1 [ s1 = s2, s2 = s1 ] endmodule
module process3 = process1 [ s1 = s3, s3 = s1 ] endmodule
```

observation: lots of symmetry in (PRISM) models

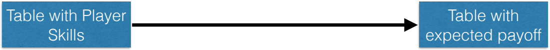
idea: avoid state space explosion by exploiting symmetries

approach: construct new program that captures symmetries

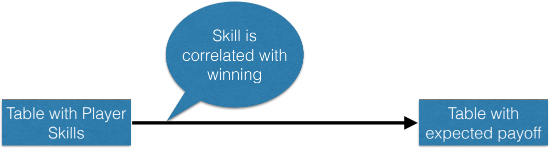
11: Probabilistic Assertions

- Traditional programs:
  - `assert expr`
  - `expr` must hold on each program execution passing `assert`
- Probabilistic programs have probabilistic outcomes – Traditional assertions not suitable
- Approach of this paper:
  - `passert expr`, p, c
  - `expr` must hold with *proability* p at *confidence* c
  - Efficient evaluation scheme to verify probabilistic assertions
- Bachelor students preferred

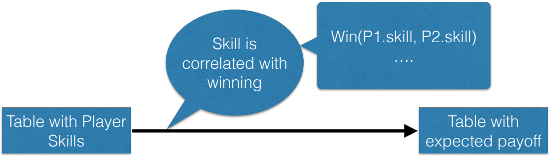
Placing a bet:



Placing a bet:

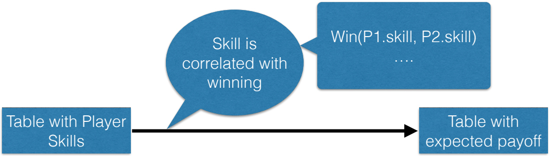


Placing a bet:



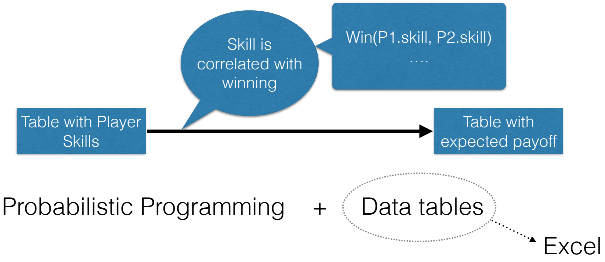
12: Bringing Probabilistic Programming to MS Excel

Placing a bet:

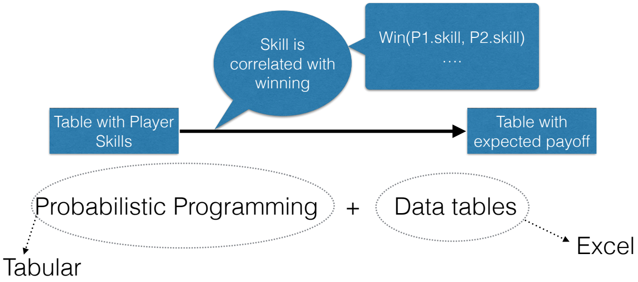


Probabilistic Programming + Data tables

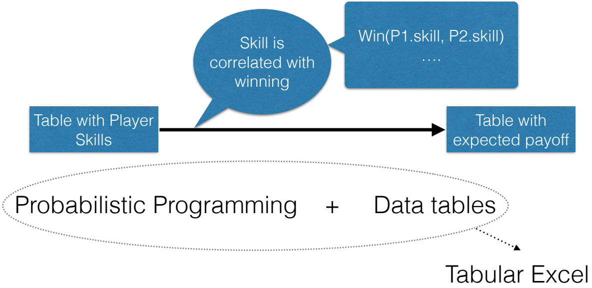
Placing a bet:



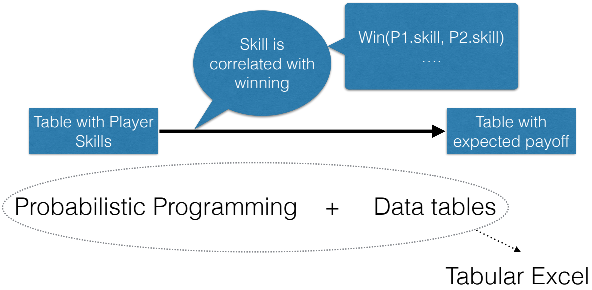
Placing a bet:



Placing a bet:



Placing a bet:



Semantics?

Reduction to Tabular & Dependent Type System

- Non–prob. programs terminate if its computations are finite
- Probabilistic programs may admit infinite computations which occur only with probability 0
- More appropriate notion: Almost–sure termination
  - Program terminates with probability 1
  - More difficult to decide than termination problem for non–probabilistic programs
- In this paper:
  - New algorithm for (semi)deciding almost–sure termination
  - Construct terminating patterns that have probability one
  - Algorithm is complete for a certain class of programs called weakly finite programs

Selecting your Topic

Please, choose your three preferred topics

Seminar Withdrawal

Withdrawal

- You have up to 3 weeks to refrain from participating in the seminar, once you are assigned your topic
- Later cancellation causes a not passed grade for the seminar

Outline

Final Remarks

Resources and Contact Information

- Seminar webpage: https://moves.rwth-aachen.de/teaching/ss-16/pp/
  - Kick-off meeting slides
  - LaTeX template for the report
  - Topic-Student assignment
  - Support resources: LaTeX, witting scientific papers, giving presentations
- Further inquiries: federico.olmedo@cs.rwth-aachen.de