Exercise Sheet 8

General remarks:

- **Due date:** Friday, January 13th 12:30 (before the exercise class).
- This is the last regular exercise sheet.
- Please submit your solutions via MOODLE. Remember to provide your matriculation number. It is necessary to hand in your solutions in groups of **three**. You may use the MOODLE forum to form groups.
- Solutions must be written in English.
- While we will publish sketches of exercise solutions, we do *not* guarantee that these sketches contain all details that are necessary to properly solve an exercise. Hence, it is recommended to attend the exercise classes.
- If you have any questions regarding the lecture or the exercise, please use the forum in MOODLE.

Exercise 1 (Compositionality of Termination) — 25P

(a) [5P] Prove or disprove: If P_1 and P_2 are universally almost-surely terminating (AST) pCGL programs, then $P_1; P_2$ is universally AST as well.

(b) [20P] Prove or disprove: If P_1 is universally PAST and P_2 contains neither loops nor diverge statements, then $P_1; P_2$ is universally PAST.

Exercise 2 (Counting Events with ert) — 25P

(a) [10P] For each of the following situations, explain how you can adapt the definition of ert from the lecture such that for a given pGCL program P, $\mathsf{ert}(P, 0)$ characterizes

1. the expected number of writing accesses to specified variable x during a run of P;
2. the expected number of loop iterations during a run of P.

You do not have to prove your answers.

(b) [15P] Reconsider the familiar random walk program, this time with a drift to the left:

$$\texttt{while}\ (x > 0)\ \{\ \ \{\ x := x - 1\ \}\ [2/3]\ \{\ x := x + 1\ \}\ \ \}$$

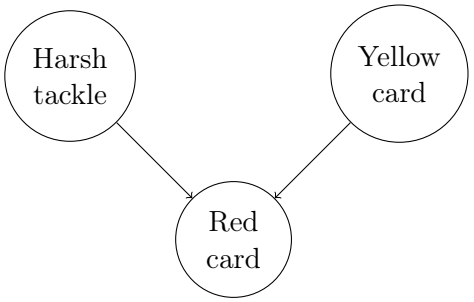
For simplicity, we assume that x is a non-negative integer variable. For all $x \geq 0$ let L_x denote the expected number of *loop iterations* of the above program when started in x. Obviously, $L_0 = 0$. Use your adapted ert from part (a) to prove the following: For all $x \geq 1$,

$$L_x \ \leq\ 3x\ .$$

Remark: In a previous version of this exercise sheet, we asked you to show that for all $x \geq 1$, $L_x \leq 3 + L_{x-1}$. This property is true but the proof is beyond the scope of the lecture. The points of this subexercise are therefore considered bonus points.

Exercise 3 (From BNs to Programs) **25P**

Consider the following Bayesian network modeling the probability that a football referee will show a red card (R), taking into account whether or not the player in question has tackled harshly (H) and if she has already seen a yellow card (Y):



The (conditional) probability tables are given as follows:

H = 0	H = 1
0.8	0.2

Y = 0	Y = 1
0.9	0.1

	R = 0	R = 1
H = 0, Y = 0	0.99	0.01
H = 0, Y = 1	0.8	0.2
H = 1, Y = 0	0.6	0.4
H = 1, Y = 1	0.03	0.97

(a) [15P] Assume the evidence is $Y = 1$ and $R = 1$. Map the network to a `pGCL`-program as in Lecture 19. You may use the `repeat {...} until(...)` statement.

(b) [5P] Is your program from (a) a.s.-terminating? Justify your answer!

(c) [5P] Reconsider your program from (a). Express the probability that the player tackled harshly ($H = 1$) given evidence $Y = 1$, $R = 1$ in terms of *wp* of your program. You do not have to calculate this probability.

Exercise 4 (From pGCL Programs to Bayesian Networks) **25P**

Let B, E, A, J and M be Boolean variables. Moreover, consider the following pGCL program:

$$
\begin{array}{l}
B :\approx 0.001 \cdot [\texttt{true}\rangle + 0.999 \cdot [\texttt{false}\rangle; \\
E :\approx 0.002 \cdot [\texttt{true}\rangle + 0.998 \cdot [\texttt{false}\rangle; \\
\texttt{if}(B)\ \{ \\
\quad \texttt{if}(E)\ \{ \\
\quad\quad A :\approx [\texttt{true}\rangle \cdot 0.95 + [\texttt{false}\rangle \cdot 0.05 \\
\quad \}\ \texttt{else}\ \{ \\
\quad\quad A :\approx [\texttt{true}\rangle \cdot 0.94 + [\texttt{false}\rangle \cdot 0.06 \\
\quad \} \\
\} \texttt{ else } \{ \\
\quad \texttt{if}(E)\ \{ \\
\quad\quad A :\approx [\texttt{true}\rangle \cdot 0.29 + [\texttt{false}\rangle \cdot 0.71 \\
\quad \}\ \texttt{else}\ \{ \\
\quad\quad A :\approx [\texttt{true}\rangle \cdot 0.01 + [\texttt{false}\rangle \cdot 0.99 \\
\quad \} \\
\}; \\
\texttt{if}(A)\ \{ \\
\quad J :\approx [\texttt{true}\rangle \cdot 0.90 + [\texttt{false}\rangle \cdot 0.10 \\
\quad M :\approx [\texttt{true}\rangle \cdot 0.70 + [\texttt{false}\rangle \cdot 0.30 \\
\} \texttt{ else } \{ \\
\quad J :\approx [\texttt{true}\rangle \cdot 0.05 + [\texttt{false}\rangle \cdot 0.95 \\
\quad M :\approx [\texttt{true}\rangle \cdot 0.01 + [\texttt{false}\rangle \cdot 0.99 \\
\}
\end{array}
$$

(a) [15P] Translate the above program into a corresponding Bayesian network. (The Bayesian network must include the DAG describing the probabilistic dependences between variables as well as the corresponding conditional probability table for each of the variables.)

(b) [10P] Using the Bayesian network constructed in the above exercise, determine probability

$$\Pr(\neg B \mid \neg E, A)\ .$$

Exercise 5 (I.i.d-loops) **30P**

For each of the following expectation-program pairs (f, P) prove or disprove that the loop P is i.i.d. with respect to expectation f. If applicable, give the exact runtime $ert(P, 0)$ using the rule from Lecture 18, slide 28.

(a) [10P] $f = x + y + z$ and P is the following loop :

```
while (x + 2y + 4z >= 6) {
    { x:=0 } [1/2] { x:=1 };
    { y:=0 } [1/2] { y:=1 };
    { z:=0 } [1/2] { z:=1 }
}
```

(b) [10P] $f = x$ and P is the following loop:

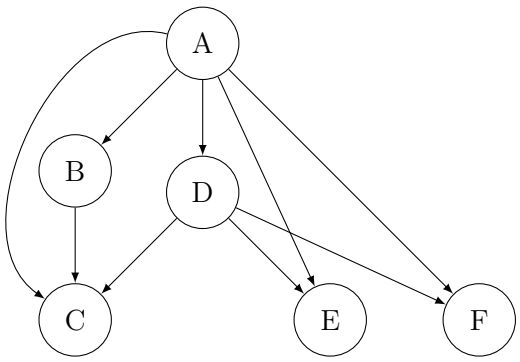
```
while (x > 0) {
    { x:= x - 1 } [1/2] { skip };
}
```

(c) [10P] $f = 1$ and P is the following loop:

```
while (y = 42) {
    { x:= x + 1 } [1/2] { x := x + 2 };
}
```

Exercise 6 (Bayesian Networks) **20P**

Consider the following Bayesian network topology:



(a) [10P] Use the d-separation algorithm to decide whether $B \perp\!\!\!\perp (E \cup F) \mid A$ holds.

(b) [10P] Give the average Markov blanket of the network.

Bonus Exercise 7 (Tools for Probabilistic Program Verification: KIPRO2) **0P**

Note: This is a bonus exercise and will not be corrected. However, we are happy to help with questions about the tool. Send an email to `phisch@cs.rwth-aachen.de`.

Throughout these exercise sheets, we did verification of probabilistic programs by hand. However, a part of our ongoing research is about automating this task. In this exercise, we invite you to play with our prototype tool KIPRO2. It was published as part of the paper "Latticed k-Induction with an Application to Probabilistic Programs" at CAV 2021. As a research prototype, it is intended to demonstrate a new proof rule for loops called *latticed k-induction* that generalizes the induction rule you know from lecture 11, slide 14. In particular, this means we can use KIPRO2 to verify super-invariants for weakest pre-expectations and expected run-times.

Being a prototype, KIPRO2 only supports programs that consist of one loop without nested loops or code before or after the loop. We are working on new tools that do not have these restrictions and that support more proof rules. If you are interested in researching with us, write us an email for Bachelor's/Master's theses topics or Hiwi positions!

The tool is available on GitHub at `https://github.com/moves-rwth/kipro2`. We recommend you use our Docker image on Zenodo. Read a bit of the README in our repository for the necessary commands to run.

(a) [0P] Recall the *geometric loop* program (lecture 11, slide 16, modified here a bit):

$$\texttt{while}\ (f = 1)\ \{\ \{\ f := 0\ \}\ [0.5]\ \{\ c := c + 1\ \}\ \}$$

We can verify automatically that the expected value of c after termination is $c + 1$. Execute `poetry run kipro2 benchmarks/cav21/geo1.pgcl --post c --pre "c+1"`. It uses the `pre` as the invariant, and the output should say it is 2-inductive (using our new proof rule).

(b) [0P] Our familiar induction rule (lecture 11, slide 14) corresponds to 1-inductivity. We instantiate the invariant from slide 16 with $p = 0.5$, i.e. $c+1\cdot[f = 1]$. Run `poetry run kipro2 benchmarks/cav21/geo1.pgcl --post c --pre "c + 1 * [f=1]"`. It should say the property is 1-inductive.

(c) [0P] The expectation $c + 0.5 \cdot [f = 1]$ is *not* a valid lower bound to $wp(P, c)$. The tool can compute a counter-example using a technique called *bounded model checking*: `poetry run kipro2 benchmarks/cav21/geo1.pgcl --post c --pre "c + 0.5 * [f=1]"`

(d) [0P] In the lecture, we have shown that $I = c+[f = 1]\cdot\frac{p}{1-p}$ is a valid super-invariant w.r.t. post-expectation c for the program with probability p instead of just $p = 0.5$. Modify the file and try out some other values for p. To edit the program, run `nano benchmarks/cav21/geo1.pgcl`.