6.0 ECTS/4.5h VU Programm- und Systemverifikation (184.741)

15. Juni 2022

Kennzahl (study id)	Matrikelnummer (student id)	Familienname (family name)	Vorname (first name)	Platz (seat)

1.) Coverage

Consider the following program fragment and test suite:

```
bool prime (unsigned n) {
  unsigned i = 2;
  bool flag = true;
  if (n == 0 || n == 1) {
    flag = false;
  }
  while ((i <= n/2) && flag) {
    if (n % i == 0) {
        flag = false;
    }
    i = i + 1;
  }
  return flag;
}
```

Input (`n`)	Output
0	false
3	true
4	false

(Hint: Assume that division always rounds towards 0.)

(a) **Control-Flow-Based Coverage Criteria**

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above. Assume that the Boolean constants (`true` and `false`) don't constitute decisions, but that `flag` in the return statement does.

	satisfied	
Criterion	yes	no
statement coverage		
decision coverage		
branch coverage		
condition coverage		

For each coverage criterion that is *not* satisfied, explain why this is the case:

(5 points)

(b) **Data-Flow-Based Coverage Criteria**

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (here, the parameter `n` of the function does not constitute a definition, and the `return` statement is a c-use as well as a a p-use):

	satisfied	
Criterion	yes	no
all-defs		
all-c-uses		
all-p-uses		
all-c-uses/some-p-uses		

For each coverage criterion that is not satisfied, explain why this is the case:

(7 points)

(c) Consider the two coverage criteria below.

- If the test-suite from above does not satisfy the coverage criterion, **augment it with** the *minimal* number of **test-cases** such that this criterion is satisfied. If full coverage cannot be achieved, **explain why**.
- If the coverage criterion is already achieved, **explain why**.

all-p-uses/some-c-uses

Input (`n`)	Output

MC/DC

Input (`n`)	Output

(4 points)

(d) **Mutation Testing**

Assume that the condition `(i <= n/2)` is changed to `(i < n/2)`. Provide a test case that **strongly kills** the resulting mutant (i.e., a test case for which the mutant provides a return value different from the one provided by the original program.)

Test Case

Input (`n`)	Output

(2 points)

2.) (a) **Hoare Logic**

Prove the Hoare Triple below. Assume that the domain of all variables in the program are the natural numbers including 0, i.e., $\mathtt{i}, \mathtt{n}, \mathtt{s} \in \mathbb{N}_0$. You need to find a sufficiently strong loop invariant.

Annotate the following code directly with the required assertions. Justify each assertion by stating which Hoare rule you used to derive it, and the premise(s) of that rule. If you **strengthen** or **weaken** conditions, **explain your reasoning**.

Note: No points for assertions that were not clearly derived by using one of the rules from the lecture!

$\{\mathsf{true}\}$

```
if (n % 2 == 0) {

  n = n + 1;

} else {

  skip;

}

i = 0;

s = 0;

while (i != n) {

  i = i + 1;

  s = s + (i%2);

}
```

$\{(\mathsf{s} \leq \mathsf{n})\}$

(7 points)

(b) **Hoare Logic - BONUS TASK (5 bonus points):**
Prove that $\{(1 \leq s) \wedge (s \leq n)\}$ holds at the end of the program!
Important: Prove the original assertion $\{(\mathsf{s} \leq \mathsf{n})\}$ on the previous page first!
Use this page for the **bonus task** only – it can be ignored if you don't want to solve it.

$\{\mathsf{true}\}$

```
if (n % 2 == 0) {

  n = n + 1;

} else {

  skip;

}

i = 0;

s = 0;

while (i != n) {

  i = i + 1;

  s = s + (i%2);

}
```

$\{(1 \leq \mathsf{s}) \wedge (\mathsf{s} \leq \mathsf{n})\}$

(5 points)

3.) Invariants Consider the following program, where `a`, `b`, `t`, `x` and `y` are integer values in $\mathbb{Z}$ (that means no over- or underflow can happen):

```
if (x > y) {
  int t = x; x = y; y = t;
}
if (a > b) {
  x = b; y = a;
}
while (y > x) {
  a = a - 1;
  y = y - 1;
}
```

Consider the formulas below; tick the correct box (☑) to indicate whether they are loop invariants for the program above.

- If the formula is an inductive invariant for the loop, provide an informal argument that the invariant is inductive.
- If the formula P is an invariant that is *not* inductive, give values of `a`, `b`, `x`, and `y` before and after the loop body demonstrating that the Hoare triple

$$\{P \wedge B\} \quad \texttt{a} = \texttt{a} - 1;\ \texttt{y} = \texttt{y} - 1; \quad \{P\}$$

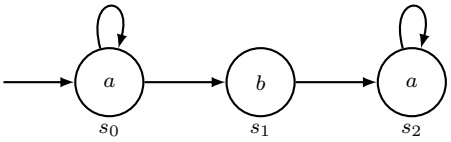
  (where B is `(y > x)`) does not hold.
- Otherwise, provide values of `a`, `b`, `x`, and `y` that correspond to a reachable state showing that the formula is *not* an invariant.

$(a > b) \Rightarrow (y \geq x)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(a > b) \Rightarrow (y > x)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(x > y) \Rightarrow (a > b)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			

(10 points)

4.) Temporal Logic

(a) Consider the following Kripke Structure:



For each formula, give the states of the Kripke structure for which the formula holds. In other words, for each of the states from the set $\{s_0, s_1, s_2\}$, consider the computation trees starting at that state, and for each tree, check whether the given formula holds on it or not.

i. $\mathbf{EG}\, a$

ii. $\mathbf{AF}\, \mathbf{EG}\, a$

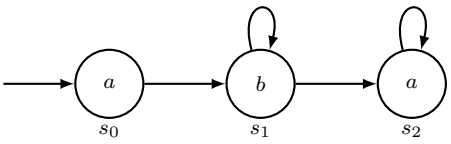
iii. $\mathbf{A}(a \wedge \mathbf{EX}\, b)$

iv. $\mathbf{A}(a\, \mathbf{U}\, b)$

v. $\mathbf{E}(a\, \mathbf{U}\, b)$

(5 points)

(b) Consider the following Kripke Structure with initial state s_0:



Use the **tableaux algorithm** from the lecture to compute the sets of states in which the following formula (and its subformulas) hold!

- For every subformula, compute the states for which it holds!
- For fixpoints, list every step of the computation!

$$\mathbf{EF}\,(\mathbf{EX}\,a)$$

(5 points)

5.) Decision procedures

(a) Consider the following formula in propositional logic; is it satisfiable? If yes, provide a satisfying assignment, if not, **provide the steps of the CDCL algorithm that led to this conclusion:**

- illustrate the conflict graphs for the relevant implication levels and
- provide the learned clauses.

$$
\begin{aligned}
&(\neg x_1 \vee \neg x_2) \wedge (x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge \\
&\quad (\neg x_3 \vee \neg x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (x_4 \vee x_5) \wedge \\
&\quad (\neg x_5 \vee \neg x_6) \wedge (x_5 \vee x_6) \wedge (\neg x_6 \vee \neg x_7) \wedge (x_6 \vee x_7) \wedge \\
&\qquad\qquad (\neg x_1 \vee \neg x_7) \wedge (x_1 \vee x_7) \wedge (x_4 \vee x_5 \vee x_6)
\end{aligned}
$$

(4 points)

(b) Consider the following formulas in Equality Logic with Uninterpreted Functions (EUF); are they satisfiable?

- If yes, provide a **satisfying interpretation**,
- If not,
  i. encode the formula as an **equisatisfiable formula in equality logic without uninterpreted functions** and
  ii. give the reasoning based on equivalence classes that leads to this conclusion.

$$g = h \ \wedge \ a = b \ \wedge \ a = c \wedge e \neq i \ \wedge \ d = e \ \wedge \ f = e \ \wedge \ h = i \ \wedge \ f(a) \neq f(h) \wedge a = i \quad (1)$$

$$g = h \ \wedge \ a = b \ \wedge \ a = c \wedge e \neq i \ \wedge \ d = e \ \wedge \ f = e \ \wedge \ h = i \ \wedge \ f(a) \neq f(d) \quad (2)$$

$$i = j \ \wedge \ j = k \ \wedge \ l \neq n \ \wedge \ m = n \ \wedge \ g(k) = g(l) \ \wedge \ f(i) \neq f(m) \quad (3)$$

(6 points)

6.) General Questions

(a) Indicate whether the following statements are true or false!

Statement	True	False
If a test-suite achieves `all-c-uses/some-p-uses` and `all-p-uses/some-c-uses` coverage, it also achieves `all-uses` coverage.	◯	◯
Let the bit-vectors `x` and `y` be represented by $\mathtt{x}_2$, $\mathtt{x}_1$, $\mathtt{x}_0$ and $\mathtt{y}_2$, $\mathtt{y}_1$, $\mathtt{y}_0$, respectively (where $\mathtt{x}_0$ and $\mathtt{y}_0$ are the least significant bits). Then there exists a BDD with variable order $\mathtt{x}_2 > \mathtt{y}_2 > \mathtt{x}_1 > \mathtt{y}_1 > \mathtt{x}_0 > \mathtt{y}_0$ that represents the operation $\mathtt{x} \neq \mathtt{y}$ whose size is polynomial in the number of variables.	◯	◯
AG F p and **AG EF** p are logically equivalent.	◯	◯
Any unsatisfiable Boolean formula with n variables for which there exists an Ordered Binary Decision Diagram (OBDD) of constant size can be disproved by a SAT Solver in $O(n)$.	◯	◯
All temporal logic formulas in (LTL $\cap$ CTL) can be expressed in CTL* without using the **X** operator.	◯	◯

(5 points)