6.0 ECTS/4.5h VU Programm- und Systemverifikation (184.741)
29 September 2023

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

1.) Coverage

Consider the following program fragment and test suite:

```
int multiply_modulo (long long a,
                     long long b,
                     long long mod) {
long long res = 0;
a = a % mod;
while (b != 0) {
  if (b & 1)
      res = (res + a) % mod;
  a = (a << 1) % mod;
  b = b >> 1;
}
return res;
}
```

a	b	mod	Output
1	1	2	1
2	2	2	0

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

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above.

	satisfied	
Criterion	yes	no
statement coverage		
path coverage		
decision coverage		

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

(4 points)

1	2	3	4	5	6	∑
/18	/10	/10	/09	/06	/07	/60

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

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (here, the parameters `a`, `b`, and `mod` of the function do not constitute a definition, but their re-assignment does. The `return` statement is a c-use):

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

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

(8 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-uses

Input (`n`)	Output

MC/DC

Input (`n`)	Output

(4 points)

(d) **Mutation Testing**

Assume that the assignment `i = b >> 1;` is changed to `i = i / 2;`. Either 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 and specified by the test case), **or** explain why no such test case can exist (i.e., the mutant is *equivalent*).

Test Case

Input (`n`)	Output

(2 points)

2.) 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., $\texttt{a}, \texttt{b}, \texttt{x}, \texttt{y}, \texttt{res} \in \mathbb{N}_0$. You need to find a sufficiently strong loop invariant.

Note: `y/2` denotes *integer division* (i.e., always rounds down).

Hint: What is the relation between `(a * b)` and `(x * y)`?

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 not clearly derived by using one of the rules from the lecture!

$\{\mathsf{true}\}$

```
res = 0;

x = a;

y = b;

while (y != 0) {

  if (y % 2 == 1)

    res = res + x;

  else

    skip;

  x = x * 2;

  y = y / 2;

}
```

$\{(\mathsf{res} = \mathsf{a} \cdot \mathsf{b})\}$

(10 points)

3.) Invariants Consider the following program, where `i`, `j`, `x`, and `y` are unsigned integers (i.e, $\mathbb{N} \cup \{0\}$). Note that `i/2` denotes integer division (always rounds down).

```
i = x;
j = y;
while (i > 1) {
  i = i / 2;
  j = j * 2;
}
```

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 `i`, `j`, `x`, and `y` before and after the loop body demonstrating that the Hoare triple

$$\{P \wedge B\} \quad \texttt{i} = \texttt{i}/2; \ \texttt{j} = \texttt{j} * 2; \quad \{P\}$$

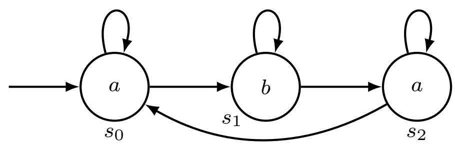
  (where B is `(i > 1)`) does not hold.
- Otherwise, provide values of `i`, `j`, `x`, and `y` that correspond to a reachable state showing that the formula is *not* an invariant.

$(i \cdot j \leq x \cdot y)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(i \cdot j = x \cdot y)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(i \cdot j > 0)$	☐ 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. $\mathbf{EG}\,\mathbf{F}\, a$

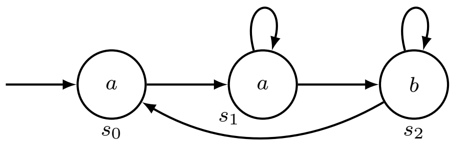
iii. $\mathbf{A}(a \wedge \mathbf{X}\, b)$

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

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

(5 points)

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



Use the **tableaux algorithm** for CTL 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!

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

(4 points)

5.) Decision procedures

Consider the following formula in propositional logic; is it satisfiable?

- If yes, **provide all satisfying assignments** and **explain how you arrived at that number**
- 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.

$$
\begin{aligned}
&(\neg x_1 \lor x_2) \land (x_1 \lor \neg x_2) \land (\neg x_2 \lor x_3) \land (x_2 \lor \neg x_3) \land \\
&\quad (\neg x_3 \lor x_4) \land (x_3 \lor \neg x_4) \land (\neg x_4 \lor x_5) \land (x_4 \lor \neg x_5) \land \\
&\quad (\neg x_5 \lor x_6) \land (x_5 \lor \neg x_6) \land (\neg x_6 \lor x_7) \land (x_6 \lor \neg x_7) \land \\
&\quad (\neg x_1 \lor \neg x_6 \lor \neg x_7) \land (\neg x_1 \lor \neg x_7 \lor x_6) \land (\neg x_6 \lor x_7 \lor x_1) \land (x_6 \lor \neg x_7 \lor x_1)
\end{aligned}
$$

(6 points)

6.) General Questions

Indicate whether the following statements are true or false!

Statement	**True**	**False**
Any assertion implied by an inductive invariant of a program is also an inductive invariant.	◯	◯
<u>No</u> CTL formula that contains at two least temporal operators (with preceding path quantifiers) can be reformulated as an eqivalent LTL property.	◯	◯
If a program terminates on all inputs, path coverage can always be achieved.	◯	◯
Any formula in Conjunctive Normal Form can be converted into a Binary Decision Diagram.	◯	◯
Any formula in Conjunctive Normal Form can be converted into a Binary Decision Diagram whose size is polynomial in the number of variables.	◯	◯
If `all-c-uses/some-p-uses` *and* `all-p-uses/some-c-uses` is achieved, then `all-uses` is also achieved.	◯	◯
The Hoare triple {true} `while(true) skip;` {false} is valid.	◯	◯

(7 points)