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

June 14, 2017

Kennzahl (study id)	Matrikelnummer (student id)	Familienname (family name)	Vorname (first name)	Gruppe (version)
				A

1.) Coverage

Consider the following program fragment and test suite:

```
bool prime (unsigned n) {
  bool result = true;
  unsigned i = 2;
  while ((i != n) && result) {
    if (n % i == 0)
        result = false;
    else
        i = i + 1;
  }
  return result;
}
```

Inputs	Outputs
`n`	result
0	false
3	true
42	false

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

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (assume that the term "decision" refers to all non-constant Boolean expressions in the program).

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

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

(7 points)

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

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (here, the parameters of the function do not constitute definitions, and the `return` statements are c-uses):

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

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

(7 points)

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

MC/DC

Inputs	Outputs
`n`	result

(1 point)

2.) Black/Gray-Box Testing

The function

```
float interest(float balance)
```

takes as parameter the balance of a bank account and returns the corresponding interest. The customer is charged 6,9% interest if the balance is negative, and the account has an overdraft limit of 1000 euro. The bank pays 0,01% interest for a positive balance up to 10 000 euro; 0,02% interest for a positive balance above 10 000 euro and up to 50 000 euro, and -0,4% penalty interest for balances above 50 000 euro. Customers with a balance of more than one million euros are required to upgrade to a premium bank account.

(a) **Equivalence Partitioning**

From the specification above, derive equivalence classes for the method `interest`. Use the table below to partition them into *valid equivalence classes* (valid inputs) and *invalid equivalence classes* (invalid inputs). Label each of the equivalence classes clearly with a number (in the according column). For each correct *equivalence class* you can score one point (up to 7 points).
(Do not provide test-cases here – that's task (b))

Condition	Valid	ID	Invalid	ID

(7 points)

(b) **Boundary Value Testing**

Use *Boundary Value Testing* to derive a test-suite for the method `interest`. Specify the input values of `balance`. Indicate clearly which equivalence classes each test-case covers by referring to the numbers from task (a). You can receive up to 8 points (1 per test case), where redundant test-cases and test-cases that do not represent boundary values do not count.

Input	**Output**	**Classes Covered**

(8 points)

3.) (a) **Invariants** Consider the following program, where `x` and `y` are non-negative natural numbers (possibly 0):

```
x = 0; y = 50;
while (x < 100) {
  x = x + 1;
  if (x > 50) {
    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 `x` and `y` before and after the loop body demonstrating that the Hoare triple

$$\{P \wedge B\} \quad \texttt{x} = \texttt{x} + 1;\ \texttt{if}\ (\texttt{x} > 50)\ \texttt{y} = \texttt{y} + 1;\ \texttt{else skip};\quad \{P\}$$

  (where B is `(x < 100)`) does not hold.
- Otherwise, provide values of `x` and `y` that correspond to a reachable state showing that the formula is *not* an invariant.

$(x \leq 50) \vee (y = x)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(x \leq 50 \wedge y = 50) \vee (50 \leq x \wedge x \leq 100 \wedge y = x)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(x \cdot y < 100^2)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			

(10 points)

(b) **Hoare Logic**

Prove the Hoare Triple below (assume that the domain of all variables in the program are the integers, i.e., $\mathtt{x}, \mathtt{y}, \mathtt{n}, \mathtt{m} \in \mathbb{Z}$. You need to find a sufficiently strong loop invariant. In the post-condition, $|n - m|$ denotes the absolute value of expression $n - m$.

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.

```
{true}

x := n;

y := m;

if (x > y) {

  t := x;

  x := y;

  y := t;

} else {

  skip;

}

while (x != 0) {

  x := x - 1;

  y := y - 1;

}

{(y = |n − m|)}
```

(10 points)

4.) Decision procedures

(a) Consider the following formulas in propositional logic; are they satisfiable? If yes, provide a satisfying assignment over booleans, if not, give the reasoning that leads to this conclusion.

$$\neg p \wedge (\neg q \vee q \vee \neg q) \wedge (p \vee \neg r) \wedge q \wedge (\neg t \vee r \vee \neg s) \wedge (s \vee s) \wedge t \tag{1}$$

$$(\neg q \vee t) \wedge r \wedge (p \vee \neg q \vee r) \wedge \neg q \wedge p \wedge (\neg r \vee q \vee \neg s) \wedge s \tag{2}$$

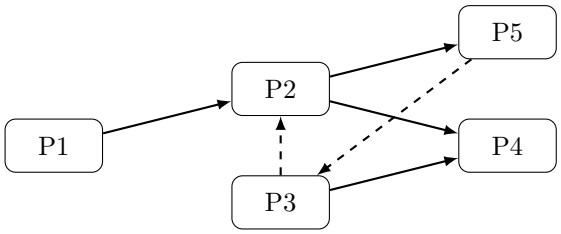
$$(\neg p \vee t) \wedge (\neg q \vee \neg p) \wedge (\neg s \vee s) \wedge p \wedge r \wedge (\neg t \vee r \vee t) \wedge (p \vee p) \tag{3}$$

$$\neg r \wedge q \wedge (\neg r \vee \neg q) \wedge t \wedge (\neg t \vee \neg t) \wedge (\neg t \vee r) \wedge p \tag{4}$$

$$\neg q \wedge (\neg r \vee t) \wedge \neg q \wedge (r \vee s \vee \neg r) \wedge \neg p \wedge s \wedge (q \vee \neg r \vee \neg t) \tag{5}$$

(5 points)

(b) Consider the following package dependency graph:



Nodes depict software packages; a solid arrow from package P_i to package P_j requires P_j to be installed if P_i is installed; a dashed arrow from package P_i to package P_j prohibits P_i to be installed if P_j is installed. We model information about installed packages using 4 boolean variables $x_1, \ldots, x_5$: for $i : 1 \leq i \leq 5$, if $x_i = \mathit{true}$, then package i is installed.

Encode the constraints of the graph as a propositional formula in CNF.
How can you check using a SAT solver whether $P3$ and $P5$ can be installed at the same time?

(8 points)

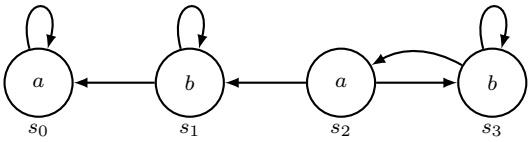
(c) Bring the following formula into conjunctive normal form (CNF) using *Tseitin transformation*.

$$(\neg a \vee (\neg\,(b \wedge c) \vee d))$$

(7 points)

5.) Temporal Logic

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 $\{s_0, s_1, s_2, s_3\}$, consider the computation trees starting at that state, and for each tree, check whether the given formula holds on it or not.

(a) $\mathbf{AG}\, a$

(b) $\mathbf{EG}\, \mathbf{F}\, a$

(c) $\mathbf{EF}\, \mathbf{G}\, \neg a$

(d) $\mathbf{A}\, \mathbf{X}\, b$

(e) $\mathbf{AF}\, \mathbf{AG}\, a \vee \mathbf{AF}\, \mathbf{AG}\, b$

(10 points)