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

June 22, 2016

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:

```
int maxsum (int max, int val) {
  int result = 0;
  int i = 0;
  if (val < 0)
    val = -val;
  while ((i < val) && (result ≤ max)) {
    i = i+1;
    result = result + i;
  }
  if (result ≤ max)
    return result;
  else
    return max;
}
```

Inputs		Outputs
`max`	`val`	result
0	0	0
0	-1	0
10	1	1

(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 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 full coverage cannot be achieved, explain why.

MC/DC

Inputs		Outputs
`max`	`val`	result

(1 point)

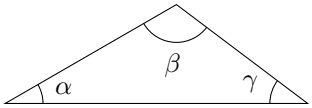
2.) Black/Gray-Box Testing

The function

`enum {ACUTE, RIGHT, OBTUSE} triangle( float` α`, float` β`, float` γ`)`

takes as parameters 3 angles α, β, and γ (in degrees) of a valid triangle and determines its type:

- acute: all angles are less than 90°,
- right: has a right angle (90°),
- obtuse: has an angle that is larger than 90°.



A valid triangle satisfies the following conditions:

- All angles are positive values (in degrees), and
- the sum of all angles is the straight angle (180°).

(a) **Equivalence Partitioning**

From the specification above, derive equivalence classes for the method `triangle`. 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 `triangle`. Specify the inputs points α, β, and γ. Indicate clearly which equivalence classes each test-case covers by referring to the numbers from task (a). You can receive up to 8 points, where redundant 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 `r` and `i` are integers and `n` is a non-negative natural number (possibly 0):

```
r = -1;
i = 0;
while ((i <= n) && (r == -1)) {
  if (i*i == n)
    r = i;
  else
    i = i + 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

$$\{P \wedge B\} \quad \texttt{if}\ (\texttt{i} * \texttt{i} == \texttt{n})\ \texttt{r} = \texttt{i};\ \texttt{else}\ \texttt{i} = \texttt{i} + \texttt{1}; \quad \{P\}$$

  does not hold.
- Otherwise, provide values of `r`, `i`, and `n` that correspond to a reachable state showing that the formula is *not* an invariant.

$(r^2 = n) \vee (r < 0)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(i^2 \leq n)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(r \leq 1) \vee (r \neq n)$	☐ 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 natural numbers including 0, i.e., $\texttt{x}, \texttt{y} \in \mathbb{N}_0$ or, equivalently, both `x` and `y` are of type `unsigned`). 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.

{true}

```
if (x > y) {

  t := x;

  x := y;

  y := t;

} else {

  skip;

}

while (x < y) {

  x := x + 1;

  y := y - 1;

}
```

$\{(x - y \leq 1)\}$

(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.

$$(\neg a \vee a) \wedge (\neg b \vee e \vee b) \wedge (v \vee c) \wedge c \wedge e \wedge (\neg c \vee b) \wedge (\neg d \vee \neg c) \tag{1}$$

$$b \wedge (\neg b \vee \neg b) \wedge \neg e \wedge (\neg b \vee e) \wedge (\neg e \vee \neg d) \wedge d \wedge c \tag{2}$$

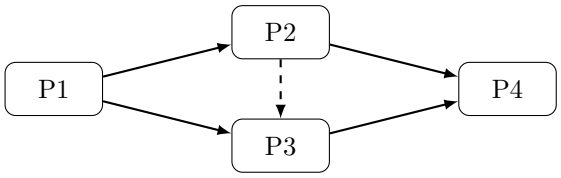
$$(e \vee \neg e \vee \neg b) \wedge (u \vee a \vee \neg e) \wedge \neg d \wedge \neg c \wedge a \wedge (\neg e \vee b) \wedge \neg d \tag{3}$$

$$(v \vee \neg e) \wedge \neg c \wedge (a \vee a) \wedge b \wedge (\neg d \vee d \vee \neg d) \wedge d \wedge (\neg b \vee e \vee \neg a) \tag{4}$$

$$\neg d \wedge c \wedge (\neg d \vee b) \wedge (\neg e \vee d \vee \neg a) \wedge (v \vee \neg d \vee e) \wedge a \wedge e \tag{5}$$

(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 $x_1, \ldots, x_4$: for $i : 1 \leq i \leq 4$, 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 $P1$ can be installed?

(6 points)

(c) Check the satisfiability of the following SMT formulas. Assume that $x, y, z, a, b, c \in \mathbb{Z}$ are integer constants, and $f\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ and $g\colon \mathbb{Z} \to \mathbb{Z}$ are binary and unary uninterpreted functions over integers respectively. Whenever a formula is satisfiable, give a satisfying assignment for it, i.e., integer values for all variables and function interpretations over integers that make the formula true under the assignment. Whenever a formula is not satisfiable, give a reason why it is unsatisfiable.

$$f(3, y) = 6 \wedge f(y, x) = f(x, y) \wedge f(y, 4) = 8 \wedge f(y, y) = 4 \tag{6}$$

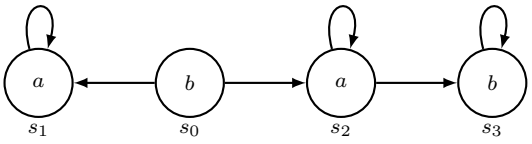
$$\begin{aligned} &\wedge f(1, x) = 3 \wedge f(1, x) = f(x, 1) \wedge g(x) = f(1, x) \\ &\qquad g(g(g(1))) = 1 \wedge g(g(1)) \neq f(x, 1) \wedge x = g(g(1)) \end{aligned} \tag{7}$$

$$\begin{aligned} &f(x, x) = x \wedge f(y, y) = y \wedge a \neq b \wedge f(x, y) = f(y, x) \wedge f(0, 1) = a \wedge f(1, 0) = b \\ &\qquad \wedge (f(x, x) = 0 \vee f(x, x) = 1) \wedge (f(y, y) = 0 \vee f(y, y) = 1) \end{aligned} \tag{8}$$

(9 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, consider the computation trees starting with one of the states from the set $\{s_0, s_1, s_2, s_3\}$, and for each tree, check whether the given formula holds on it or not.

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

(b) $a \vee \mathbf{A}\,\mathbf{X}\,b$

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

(d) $\mathbf{EF}\,\mathbf{G}\,\neg b$

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

(10 points)