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

June 19, 2018

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 range_check (unsigned m, unsigned n) {
  if (m > n) {
    unsigned t = m;
    m = n;
    n = t;
  }
  bool result = false;
  bool tmp = true;
  unsigned i = m;
  while ((i > 0) && (i < n)) {
    i = i + 1;
    if (i % m == 0)
        result = result || tmp;
  }
  return result;
}
```

Inputs		Outputs
`m`	`n`	result
3	7	true
1	0	false
2	5	true

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

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (important: assume that the term "decision" refers to all Boolean expressions that contain variables, including `result || tmp`).

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

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

(6 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, the `return` statement is a c-use, and `result = (result || tmp)` is a c-use but not a p-use):

	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:

(9 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

Inputs		Outputs
`m`	`n`	result

all-c-uses

Inputs		Outputs
`m`	`n`	result

(2 points)

(d) Consider the expression $((a \wedge b) \vee c)$, where a, b, and c are Boolean variables. Provide a *minimal* number of test cases such that modified condition/decision coverage is achieved for the expression. Clarify <u>for each test case</u> *which* condition(s) independently affect(s) the outcome.

MC/DC

Inputs			Outcome
`a`	`b`	`c`	`(a && b) || c`

(3 points)

2.) Hoare Logic

Prove the Hoare Triple below (assume that the domain of all variables in the program are the unsigned integers including zero, i.e., $\mathtt{x}, \mathtt{y}, \mathtt{n}, \mathtt{m} \in \mathbb{N} \cup \{0\}$). You need to find a sufficiently strong loop invariant.
Hint: you will need an expression that represents how often the loop has been executed.

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 := 0;

if (m != 0) {

  while (x != 0) {

    x = x - 1;

    y = y + m;

  }

} else {

  skip;

}
```

$\{y = n \cdot m\}$

(10 points)

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

```
x = 0; y = 50;
while (x != y) {
  x = x + 1;
  if (x % 2 == 0) {
    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

$$\{P \wedge B\} \quad \texttt{x} = \texttt{x} + 1;\ \texttt{if}\ (\texttt{x}\%2 == 0)\ \texttt{y} = \texttt{y} + 1;\ \texttt{else skip};\quad \{P\}$$

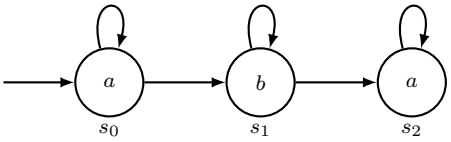
  (where B is `(x != y)`) does not hold.
- Otherwise, provide values of `x` and `y` that correspond to a reachable state showing that the formula is *not* an invariant.

$(x \leq 100) \wedge (y \leq 100)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(y - x) \leq 50$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(x \neq y)$	☐ 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. $\mathbf{AX}\, a$

ii. $\mathbf{EX}\, a$

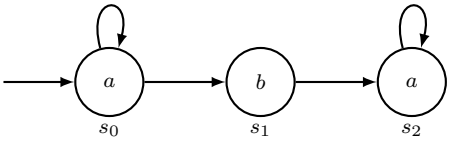
iii. $\mathbf{AF}\, b$

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

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

(5 points)

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



For each formula below, 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. The LTL formula $\mathbf{AF\,G}\,a$

ii. The CTL formula $\mathbf{AF\,AG}\,a$

Do the formulas (i) and (ii) above express the same property?

☐ yes ☐ no

If not, explain why.

(5 points)

5.) 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.

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

$$(x \vee y) \wedge (\neg x \vee \neg y) \wedge (y \vee z) \wedge (\neg y \vee \neg z) \wedge (z \vee x) \wedge (\neg z \vee \neg x) \tag{2}$$

$$(\neg x \vee y) \wedge (x \vee \neg y) \wedge (\neg y \vee z) \wedge (y \vee \neg z) \wedge (z \vee \neg z) \tag{3}$$

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

$$(x \vee y) \wedge (\neg x \vee \neg y) \wedge (y \vee \neg z) \wedge (z) \wedge (\neg x \vee z) \tag{5}$$

(5 points)

(b) **Tseitin transformation**
Let F be a formula in *conjunctive normal form* (CNF), i.e., a conjunction of clauses

$$\bigwedge_i \bigvee_j \ell_{i,j}, \quad \text{where } \ell_{i,j} \in \{P, \neg P \mid P \text{ is an atom}\}$$

Note that every clause C_i with m literals can be seen as a "nested" disjunction, i.e.,

$$C_i = (\ell_{i,1} \vee (\ell_{i,2} \vee (\ldots \vee (\ell_{i,m-1} \vee \ell_{i,m}))))$$

Explain how you can use Tseitin's transformation to convert any arbitrary formula in CNF into an equi-satisfiable formula in CNF whose clauses have at most 3 literals.

(5 points)