Advanced Topics in Software Engineering
(A.Y. 2019/2020)
Written exam - 1h30

July 13^{th}, 2020

Exercise 1

You are building a software systems that has to control a physical system that can assume four different states σ_i with $i \in \{1, 2, 3, 4\}$, represented by the values assumed by two different boolean variables x and y, and according to specific conditions of the environment that can be represented by six boolean variables b_j with $j \in \{1, 2, 3, 4, 5, 6\}$. In particular the rules driving the behavious are the following ones:

$$\begin{array}{ll} b_1 \Rightarrow b_3 & b_1, b_2, b_4 \text{ cannot hold at the same time} \\ b_1 \vee b_2 \rightarrow x & (b_5 \wedge b_4) \vee ((b_2 \wedge b_6) \vee \neg b_1) \rightarrow y \\ b_4 \rightarrow y & \end{array}$$

Derive a test set after having derived the condition-effect graph

Exercise 2

Consider the specification of the following I/O Finite State Machine.

- $\Sigma_I = \{a, b\}$ is the input alphabet
- $\Sigma_O = \{0, 1\}$ is the output alphabet
- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$ is the set of states
- q_1 is the initial state
- δ, ω are the state transition function and the output function, respectively, and are represented by the directed graph in Figure 1.

Apply the $\mathscr{W} - method$ strategy to derive a test suite assuming an implementation with one additional state.

Exercise 3

Shortly discuss, possibly providing a simplified schema of code, why applying the statement coverage adequacy criteria does not guarantee the satisfaction of the decision coverage criteria.

After having answered to the previous request provide a test set only satifying the statement coverage criteria, and another one satisfying the decision coverage criteria:

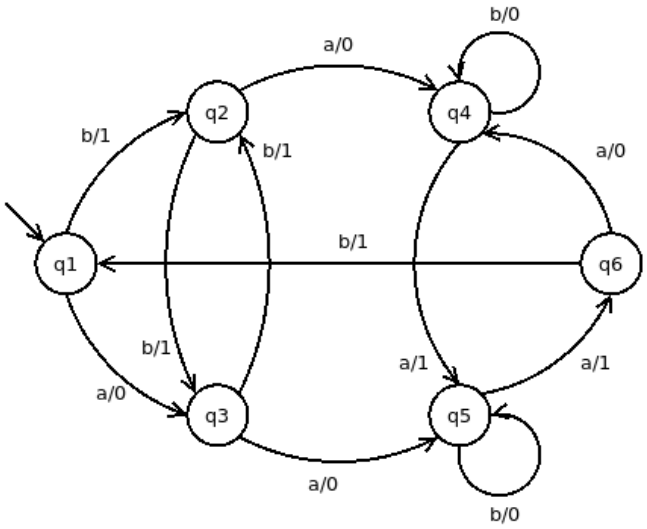

Figure 1: I/O FSM for the exercise 2

```
begin
  int x,y,z;
  input (x,y);
  z=0;
  if (x>0) then {
    print('x positive number - OK!');
  } else {
    print('x negative number - transformed!');
    x = -x;
  }
  if (y>0) then {
    print('y positive number - OK!');
  } else {
    print('y negative number - transformed!');
    y = -y;
  }
  while (y>0) {
    z = z + x;
    y = y - 1;
  }
end
```