POLITECNICO
MILANO 1863

Exercises on metamorphic testing

Exercise 1

- We have defined for following test case for a software that computes the sum of two matrices
  - A = [1 4 6 3; 6 7 4 2; 0 9 4 5]
  - B = [5 6 3 8; -3 -2 0 -5; 7 4 3 1]
- The result we have obtained from the software execution is
  - RIS = [6 10 9 11; 3 5 4 -3; 7 13 7 6]
- Define at least one additional test case using the metamorphic testing approach

Applying the metamorphic approach

- Step1: find one or more metamorphic relations
- Step2: use it to identify a test case
- Step3: execute the test and check if it is successful

Application to Exercise 1

- Possible metamorphic relations
  - Commutativity
    - A + B = B + A
  - Associativity
    - (A + B) + C = A + (B + C)
  - Sum and neutral element
    - A + O = A
- Given the available test case, we can use the commutativity
- New test case
  - A = [5 6 3 8; -3 -2 0 -5; 7 4 3 1]
  - B = [1 4 6 3; 6 7 4 2; 0 9 4 5]
- Expected result
  - RIS = [6 10 9 11; 3 5 4 -3; 7 13 7 6]

Exercise 2

- Consider a banking software handling money transfers between accounts.
- Each account can be accessed by multiple threads to account for scenarios like the following one
  - Every month, the salaries thread is transferring salaries from the employer account to the employees' accounts
  - While the transaction above is taking place, one of the employee issues a money transfer from his/her account to his/her son account
- We have successfully tested the software with the following test case
  - Initial state: account A = 3000, account B = 300, account C = -30
  - Input: operation from A to B, amount 200 and operation from B to C, amount 100
  - Result:
    - Final state: A = 2800, B = 400 , C = 70

Exercise 2

- Question 1: Do we have enough information to test concurrency?

- No, we don’t because we need to augment the test case with information about the series of read and write events concerning the variable shared between different threads/processes

Exercise 2

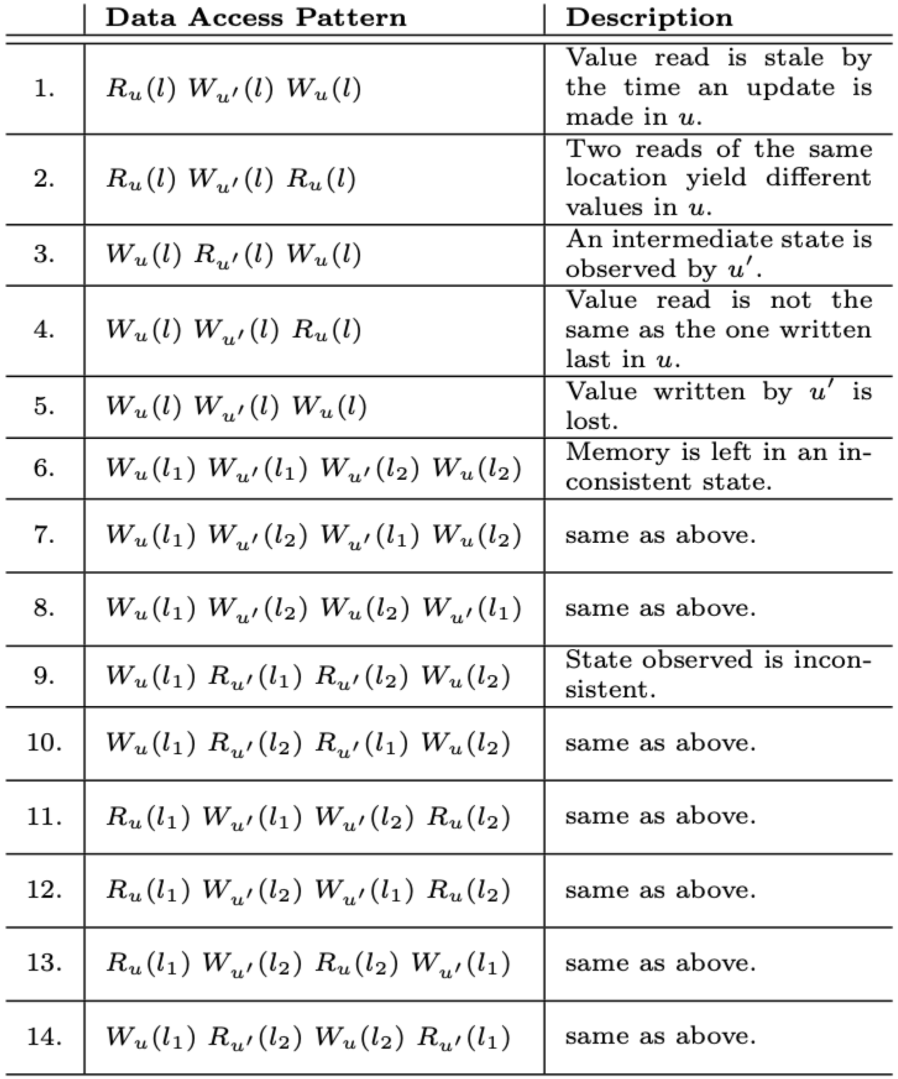
- Question 2: Assume the following series of read/write operations in the initial test case
  - *$<R_A^{t1}(3000), W_A^{t1}(2800), R_B^{t1}(300), W_B^{t1}(500), R_B^{t2}(500), R_C^{t2}(-30), W_B^{t2}(400), W_C^{t2}(70)>$*
  - How do you apply metamorphic test to find other test cases?

Exercise 2

- In this case we are focusing on testing concurrency
- So, our interest is in selecting different series of read/write and check that the software continues to behave as in the first test case

Exercise 2

- We can select the relevant situations from the ones in the table, for instance
  - *<R_A^{t1}(3000), W_A^{t1}(2800), R_B^{t1}(300), R_B^{t2}(300), W_B^{t1}(500), R_C^{t2}(-30), W_B^{t2}(200), W_C^{t2}(70)>*



	Data Access Pattern	Description
1.	$R_u(l)\ W_{u'}(l)\ W_u(l)$	Value read is stale by the time an update is made in u.
2.	$R_u(l)\ W_{u'}(l)\ R_u(l)$	Two reads of the same location yield different values in u.
3.	$W_u(l)\ R_{u'}(l)\ W_u(l)$	An intermediate state is observed by u'.
4.	$W_u(l)\ W_{u'}(l)\ R_u(l)$	Value read is not the same as the one written last in u.
5.	$W_u(l)\ W_{u'}(l)\ W_u(l)$	Value written by u' is lost.
6.	$W_u(l_1)\ W_{u'}(l_1)\ W_{u'}(l_2)\ W_u(l_2)$	Memory is left in an inconsistent state.
7.	$W_u(l_1)\ W_{u'}(l_2)\ W_{u'}(l_1)\ W_u(l_2)$	same as above.
8.	$W_u(l_1)\ W_{u'}(l_2)\ W_u(l_2)\ W_{u'}(l_1)$	same as above.
9.	$W_u(l_1)\ R_{u'}(l_1)\ R_{u'}(l_2)\ W_u(l_2)$	State observed is inconsistent.
10.	$W_u(l_1)\ R_{u'}(l_2)\ R_{u'}(l_1)\ W_u(l_2)$	same as above.
11.	$R_u(l_1)\ W_{u'}(l_1)\ W_{u'}(l_2)\ R_u(l_2)$	same as above.
12.	$R_u(l_1)\ W_{u'}(l_2)\ W_{u'}(l_1)\ R_u(l_2)$	same as above.
13.	$R_u(l_1)\ W_{u'}(l_2)\ R_u(l_2)\ W_{u'}(l_1)$	same as above.
14.	$W_u(l_1)\ R_{u'}(l_2)\ W_u(l_2)\ R_{u'}(l_1)$	same as above.