ECE 322

SOFTWARE TESTING AND MAINTENANCE

Fall 2025

Assignment #4

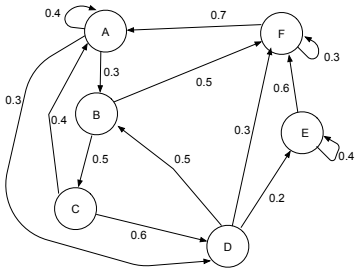
SOLUTIONS

Due date: Monday, October 27, 2025 by 3:00 PM

Total: 40 points

10 points

1. The behavior of a control software system can be provided in the form of the finite state machine shown below. The figure shows the corresponding transition probabilities between the states *A, B,...,F*.



Determine an order in which individual states have to be tested.

Solution
To determine stationary probabilities p_A, p_B... we have to solve a system of linear equations

$p_A = 0.4p_A+0.3p_B + 0.7p_F$
$p_B=0.3p_A+0.5p_D$
$p_C=0.5p_B$
$p_D=0.6p_C$
$p_E=0.4p_E+0.2p_D$
$p_F=0.5p_B+0.3p_D+0.6p_E+0.3p_F$

To solve the system of equations (the equations are linearly dependent), any of the equations from the above system (say, the last one) has to be replaced by the equation

$$p_A +p_B + p_C +p_D+ p_E +p_F =1$$

As a result (you may use a python package numpy.linalg.solve), we have the following probabilities:

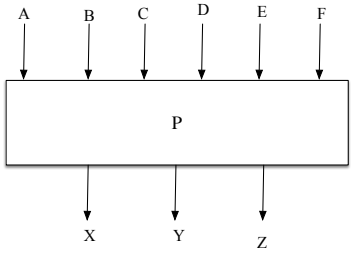
p =[0.421, 0.149, 0.074, 0.045, 0.015, 0.297]

The states are tested by starting with the one with the highest probability, namely

A (p_A=0.421), *F* (p_F=0.297), *B* (p_B=0.149), *C* (p_C=0.074), *D* (p_D=0.045), *E* (p_E=0.015)

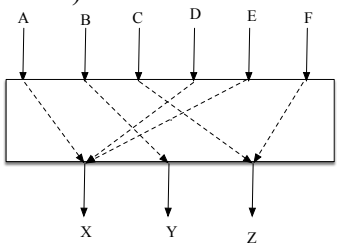
10 points

2. Consider the program P with 6 inputs A, B, C, D, E, F and 3 outputs X, Y, and Z. The input variables assume some values shown in the table below.



input	values
A	0, 1, 2, 3,4, 5
B	A, B, C, D, E
C	100, 200,300
D	7, 8, 9, 10, 11
E	U, W, Z
F	α, β, γ, δ, ε

Consider the input-output testing strategy. Consider the situation when (i) you do not know functional dependencies among the input and output variables, and (ii) where these dependencies are known (as shown in the figure below).



What is the number of test cases required when using these two strategies. For (ii) propose a collection of test cases. In this strategy, discuss how you could minimize the number of test cases; show an example.

Solution
(i)if functional dependencies are not known, the number of tests is $6*5^3*3^2$ =6,750.
(ii) for X one has 6*5*5 =150 test cases. For Y one has 5 test cases. For Z one has 3*5=15 test cases. In total, the number of test cases does not exceed 50+5+15 = 170 test cases.

The minimization of test cases can be achieved by selecting suitable values of not relevant inputs for the given output. For instance, for Z one can choose test cases

A	**B**	**C**	**D**	**E**	**F**
0	A	100	11	N	α
0	A	100	11	N	β
0	A	100	11	N	γ
0	A	100	11	N	δ
0	A	100	11	N	ε
0	A	200	11	N	α
0	A	200	11	N	β
0	A	200	11	N	γ
0	A	200	11	N	δ
0	A	200	11	N	ε
0	A	300	11	N	α
0	A	300	11	N	β
0	A	300	11	N	γ
0	A	300	11	N	δ
0	A	300	11	N	ε

Some other option could be

A	**B**	**C**	**D**	**E**	**F**
0	B	100	10	N	α
0	B	100	10	N	β
0	B	100	10	N	γ
0	B	100	10	N	δ
0	B	100	10	N	ε
0	B	200	10	N	α
0	B	200	10	N	β
0	B	200	10	N	γ
0	B	200	10	N	δ
0	B	200	10	N	ε
0	B	300	10	N	α
0	B	300	10	N	β
0	B	300	10	N	γ
0	B	300	10	N	δ

0	B	300	11	N	ε

10 points

3. For the configuration problem shown below, develop test cases using the strategy of pairwise testing. Compare the number of test cases obtained here with the number of tests required to consider all combinations of input values.
Hint: you can use the python package discussed in the lecture.

KEYBOARD	12KEY, NOKEYS, QWERTY, UNDEFINED
NAVIGATION	DPAD, NONAV, TRACKBALL, UNDEFINED, WHEEL
MEMORY	A, B, C, D, E, F, G, H, I, J, K
ORIENTATION	LANDSCAPE, PORTAIT, SQUARE, UNDEFINED
SCREEN LAYOUT	MASK, NO, UNDEFINED, YES

Solution
Using the software, the generated list of 55 test cases is shown below:

0:['12KEY', 'DPAD', 'A', 'LANDSCAPE', 'MASK']
1:['NOKEYS', 'NONAV', 'B', 'PORTAIT', 'MASK']
2:['QWERTY', 'TRACKBALL', 'C', 'SQUARE', 'MASK']
3:['UNDEFINED', 'UNDEFINED', 'D', 'UNDEFINED', 'MASK']
4:['UNDEFINED', 'WHEEL', 'E', 'SQUARE', 'NO']
5:['QWERTY', 'WHEEL', 'F', 'PORTAIT', 'UNDEFINED']
6:['NOKEYS', 'DPAD', 'G', 'UNDEFINED', 'UNDEFINED']
7:['12KEY', 'UNDEFINED', 'H', 'SQUARE', 'UNDEFINED']
8:['12KEY', 'TRACKBALL', 'I', 'PORTAIT', 'NO']
9:['NOKEYS', 'TRACKBALL', 'J', 'LANDSCAPE', 'YES']
10:['QWERTY', 'NONAV', 'K', 'LANDSCAPE', 'NO']
11:['UNDEFINED', 'NONAV', 'J', 'UNDEFINED', 'UNDEFINED']
12:['UNDEFINED', 'DPAD', 'H', 'PORTAIT', 'YES']
13:['QWERTY', 'UNDEFINED', 'I', 'UNDEFINED', 'YES']
14:['NOKEYS', 'WHEEL', 'A', 'UNDEFINED', 'NO']
15:['12KEY', 'WHEEL', 'C', 'LANDSCAPE', 'YES']
16:['12KEY', 'NONAV', 'D', 'SQUARE', 'YES']
17:['NOKEYS', 'UNDEFINED', 'E', 'LANDSCAPE', 'UNDEFINED']
18:['UNDEFINED', 'TRACKBALL', 'F', 'LANDSCAPE', 'YES']
19:['QWERTY', 'DPAD', 'B', 'SQUARE', 'NO']
20:['NOKEYS', 'TRACKBALL', 'K', 'SQUARE', 'UNDEFINED']
21:['12KEY', 'UNDEFINED', 'G', 'PORTAIT', 'NO']
22:['12KEY', 'WHEEL', 'K', 'UNDEFINED', 'MASK']
23:['QWERTY', 'TRACKBALL', 'G', 'SQUARE', 'YES']
24:['UNDEFINED', 'DPAD', 'I', 'SQUARE', 'UNDEFINED']
25:['UNDEFINED', 'NONAV', 'C', 'UNDEFINED', 'UNDEFINED']
26:['QWERTY', 'TRACKBALL', 'A', 'SQUARE', 'UNDEFINED']
27:['NOKEYS', 'NONAV', 'H', 'UNDEFINED', 'NO']
28:['NOKEYS', 'DPAD', 'F', 'SQUARE', 'NO']
29:['QWERTY', 'WHEEL', 'D', 'LANDSCAPE', 'UNDEFINED']
30:['UNDEFINED', 'UNDEFINED', 'B', 'UNDEFINED', 'UNDEFINED']

31:['12KEY', 'TRACKBALL', 'E', 'UNDEFINED', 'YES']
32:['12KEY', 'UNDEFINED', 'J', 'SQUARE', 'NO']
33:['UNDEFINED', 'WHEEL', 'G', 'LANDSCAPE', 'MASK']
34:['QWERTY', 'DPAD', 'J', 'PORTAIT', 'MASK']
35:['NOKEYS', 'NONAV', 'I', 'LANDSCAPE', 'MASK']
36:['NOKEYS', 'NONAV', 'E', 'PORTAIT', 'MASK']
37:['NOKEYS', 'DPAD', 'C', 'PORTAIT', 'NO']
38:['QWERTY', 'WHEEL', 'H', 'LANDSCAPE', 'MASK']
39:['UNDEFINED', 'UNDEFINED', 'K', 'PORTAIT', 'YES']
40:['12KEY', 'UNDEFINED', 'F', 'UNDEFINED', 'MASK']
41:['NOKEYS', 'WHEEL', 'B', 'LANDSCAPE', 'YES']
42:['NOKEYS', 'DPAD', 'D', 'PORTAIT', 'NO']
43:['12KEY', 'NONAV', 'A', 'PORTAIT', 'YES']
44:['12KEY', 'TRACKBALL', 'B', 'PORTAIT', 'YES']
45:['UNDEFINED', 'TRACKBALL', 'D', 'PORTAIT', 'YES']
46:['UNDEFINED', 'NONAV', 'F', 'PORTAIT', 'YES']
47:['UNDEFINED', 'DPAD', 'K', 'PORTAIT', 'YES']
48:['UNDEFINED', 'WHEEL', 'I', 'PORTAIT', 'YES']
49:['UNDEFINED', 'UNDEFINED', 'A', 'PORTAIT', 'YES']
50:['QWERTY', 'UNDEFINED', 'C', 'PORTAIT', 'YES']
51:['QWERTY', 'WHEEL', 'J', 'PORTAIT', 'YES']
52:['QWERTY', 'DPAD', 'E', 'PORTAIT', 'YES']
53:['UNDEFINED', 'NONAV', 'G', 'PORTAIT', 'YES']
54:['UNDEFINED', 'TRACKBALL', 'H', 'PORTAIT', 'YES']

When testing all combinations, we require 4*5*11*4*4 =3,520 test cases.

10 points

4. Propose a suite of test cases to complete syntax-based testing realizing production coverage for the following production rule describing syntax of basic algebraic expressions

<expr> ::= <term> | <expr> + <term>
<term> ::= <factor> | <term> * <factor>
<factor> ::= <variable>
<variable>::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

so that the following coverage criteria are satisfied:

(i) terminal symbol coverage,
(ii) production coverage, and
(iii) derivation coverage

Solution

A suite of tests cases could be different but make sure that

Terminal symbol coverage: select such test cases that all terminal symbols are covered.

Production coverage: select such test cases that all production rules are covered.