Q1

We know

$$p_j = \sum_{i=1}^{n} p_{ij} p_i, \quad j = 1, 2, \ldots, n. \tag{1}$$

From the figure, we have

$$p_{\mathrm{A}} = 0.8 p_{\mathrm{H}}, \tag{2}$$
$$p_{\mathrm{B}} = 0.7 p_{\mathrm{A}} + 0.5 p_{\mathrm{E}}, \tag{3}$$
$$p_{\mathrm{C}} = 1.0 p_{\mathrm{B}} + 0.7 p_{\mathrm{D}}, \tag{4}$$
$$p_{\mathrm{D}} = 0.5 p_{\mathrm{E}}, \tag{5}$$
$$p_{\mathrm{E}} = 0.3 p_{\mathrm{A}}, \tag{6}$$
$$p_{\mathrm{F}} = 0.6 p_{\mathrm{C}}, \tag{7}$$
$$p_{\mathrm{G}} = 0.4 p_{\mathrm{C}} + 1.0 p_{\mathrm{F}} + 0.2 p_{\mathrm{H}}, \text{ and} \tag{8}$$
$$p_{\mathrm{H}} = 0.3 p_{\mathrm{D}} + 1.0 p_{\mathrm{G}}. \tag{9}$$

We also have

$$p_{\mathrm{A}} + p_{\mathrm{B}} + p_{\mathrm{C}} + p_{\mathrm{D}} + p_{\mathrm{E}} + p_{\mathrm{F}} + p_{\mathrm{G}} + p_{\mathrm{H}} = 1. \tag{10}$$

With the aforementioned equations, we have a systems of equation that we can solve. Using software to solve these systems of equations, we get

$$p_{\mathrm{A}} \approx 0.1592, \tag{11}$$
$$p_{\mathrm{B}} \approx 0.1353, \tag{12}$$
$$p_{\mathrm{C}} \approx 0.1520, \tag{13}$$
$$p_{\mathrm{D}} \approx 0.0239, \tag{14}$$
$$p_{\mathrm{E}} \approx 0.0477, \tag{15}$$
$$p_{\mathrm{F}} \approx 0.0912, \tag{16}$$
$$p_{\mathrm{G}} \approx 0.1918, \text{ and} \tag{17}$$
$$p_{\mathrm{H}} \approx 0.0199. \tag{18}$$

Therefore we conclude the priority for testing should be

$$\mathrm{H} > \mathrm{G} > \mathrm{A} > \mathrm{C} > \mathrm{B} > \mathrm{F} > \mathrm{E} > \mathrm{D}. \tag{19}$$

Q2

i)

When the functional dependencies among the input and output variables is not known, we have

$$\begin{aligned} \#\text{Test cases} &= \#A \times \#B \times \#C \times \#D \times \#E \times \#F && (20)\\ &= 5 \times 5 \times 3 \times 5 \times 2 \times 3 && (21)\\ &= 2250. && (22) \end{aligned}$$

ii)

When the functional dependencies among the input and output variables is known, we have

$$\begin{aligned} \#\text{Test cases} &= \#X + \#Y + \#Z && (23)\\ &= (\#A \times \#D \times \#E) + (\#B) + (\#C \times \#F) && (24)\\ &= (5 \times 5 \times 2) + (5) + (3 \times 3) && (25)\\ &= 64. && (26) \end{aligned}$$

Example test cases are shown below. "N/A" represents an entry that can be anything.

Table 1: Test cases for output X

Test case	**A**	**B**	**C**	**D**	**E**	**F**
1	0	N/A	N/A	7	Y	N/A
2	0	N/A	N/A	7	N	N/A
3	0	N/A	N/A	8	Y	N/A
4	0	N/A	N/A	8	N	N/A
5	0	N/A	N/A	9	Y	N/A
6	0	N/A	N/A	9	N	N/A
7	0	N/A	N/A	10	Y	N/A
8	0	N/A	N/A	10	N	N/A
9	0	N/A	N/A	11	Y	N/A
10	0	N/A	N/A	11	N	N/A

11	1	N/A	N/A	7	Y	N/A
12	1	N/A	N/A	7	N	N/A
13	1	N/A	N/A	8	Y	N/A
14	1	N/A	N/A	8	N	N/A
15	1	N/A	N/A	9	Y	N/A
16	1	N/A	N/A	9	N	N/A
17	1	N/A	N/A	10	Y	N/A
18	1	N/A	N/A	10	N	N/A
19	1	N/A	N/A	11	Y	N/A
20	1	N/A	N/A	11	N	N/A
21	2	N/A	N/A	7	Y	N/A
22	2	N/A	N/A	7	N	N/A
23	2	N/A	N/A	8	Y	N/A
24	2	N/A	N/A	8	N	N/A
25	2	N/A	N/A	9	Y	N/A
26	2	N/A	N/A	9	N	N/A
27	2	N/A	N/A	10	Y	N/A
28	2	N/A	N/A	10	N	N/A
29	2	N/A	N/A	11	Y	N/A
30	2	N/A	N/A	11	N	N/A
31	3	N/A	N/A	7	Y	N/A
32	3	N/A	N/A	7	N	N/A
33	3	N/A	N/A	8	Y	N/A
34	3	N/A	N/A	8	N	N/A
35	3	N/A	N/A	9	Y	N/A
36	3	N/A	N/A	9	N	N/A
37	3	N/A	N/A	10	Y	N/A
38	3	N/A	N/A	10	N	N/A
39	3	N/A	N/A	11	Y	N/A
40	3	N/A	N/A	11	N	N/A
41	4	N/A	N/A	7	Y	N/A
42	4	N/A	N/A	7	N	N/A
43	4	N/A	N/A	8	Y	N/A
44	4	N/A	N/A	8	N	N/A

45	4	N/A	N/A	9	Y	N/A
46	4	N/A	N/A	9	N	N/A
47	4	N/A	N/A	10	Y	N/A
48	4	N/A	N/A	10	N	N/A
49	4	N/A	N/A	11	Y	N/A
50	4	N/A	N/A	11	N	N/A

Table 2: Test cases for output Y

Test case	**A**	**B**	**C**	**D**	**E**	**F**
51	N/A	A	N/A	N/A	N/A	N/A
52	N/A	B	N/A	N/A	N/A	N/A
53	N/A	C	N/A	N/A	N/A	N/A
54	N/A	D	N/A	N/A	N/A	N/A
55	N/A	E	N/A	N/A	N/A	N/A

Table 3: Test cases for output Z

Test case	**A**	**B**	**C**	**D**	**E**	**F**
56	N/A	N/A	100	N/A	N/A	α
57	N/A	N/A	100	N/A	N/A	β
58	N/A	N/A	100	N/A	N/A	γ
59	N/A	N/A	200	N/A	N/A	α
60	N/A	N/A	200	N/A	N/A	β
61	N/A	N/A	200	N/A	N/A	γ
62	N/A	N/A	300	N/A	N/A	α
63	N/A	N/A	300	N/A	N/A	β
64	N/A	N/A	300	N/A	N/A	γ

Q3

If we wanted to test all combinations using combinatorial testing, we would have

$$\#\text{Test cases} = 3 \times 3 \times 4 \times 3 \times 5 \times 4 \times 4 \times 5 \times 4 \tag{27}$$

$$= 172800. \tag{28}$$

Next, we wrote a Python script to automate generating the pairwise test cases. The result is shown below.

Table 4: Results from Python using pairwise testing

#	HKBH	KBH	KB	NVH	NV	OR	SLL	SLS	TS
1	NO	NO	12KEY	NO	DPAD	LAND	MASK	LRG	FNGR
2	NDEF	NDEF	NOKEYS	NDEF	NONAV	PORT	NO	MASK	FNGR
3	YES	YES	QWERTY	YES	TRKBALL	SQR	NDEF	NORM	FNGR
4	YES	NDEF	NDEF	NO	NDEF	NDEF	YES	SML	NTOUCH
5	NDEF	NO	NDEF	YES	WHEEL	NDEF	NDEF	NDEF	STYLUS
6	NO	YES	NOKEYS	NDEF	WHEEL	SQR	YES	NDEF	NDEF
7	NO	NDEF	QWERTY	YES	NDEF	LAND	NO	LRG	NDEF
8	NDEF	YES	12KEY	NO	NONAV	SQR	MASK	SML	STYLUS
9	YES	NO	NOKEYS	NDEF	TRKBALL	LAND	MASK	MASK	NTOUCH
10	NO	NO	QWERTY	NO	NONAV	PORT	YES	NORM	NTOUCH
11	YES	YES	12KEY	YES	DPAD	PORT	NO	NDEF	NTOUCH
12	NDEF	NDEF	12KEY	NDEF	DPAD	NDEF	NDEF	NORM	NDEF
13	NDEF	NDEF	QWERTY	NDEF	WHEEL	LAND	MASK	SML	STYLUS
14	NO	YES	NDEF	NDEF	NDEF	PORT	NDEF	LRG	STYLUS
15	NDEF	NO	NDEF	NO	TRKBALL	SQR	NO	SML	NDEF
16	YES	YES	NOKEYS	YES	NONAV	NDEF	NDEF	MASK	NDEF
17	NDEF	NDEF	NOKEYS	NO	NDEF	SQR	YES	LRG	NTOUCH
18	NO	NDEF	NDEF	NO	TRKBALL	NDEF	MASK	NDEF	FNGR
19	YES	YES	NDEF	NO	WHEEL	LAND	NDEF	NORM	NTOUCH
20	YES	NO	12KEY	YES	NDEF	PORT	MASK	NORM	NDEF
21	YES	YES	12KEY	YES	WHEEL	PORT	YES	MASK	STYLUS
22	NO	YES	NOKEYS	NO	DPAD	SQR	YES	MASK	STYLUS
23	YES	YES	QWERTY	YES	DPAD	NDEF	NO	SML	STYLUS
24	YES	YES	NDEF	NO	NONAV	LAND	YES	NDEF	FNGR
25	YES	YES	NDEF	NO	DPAD	NDEF	NDEF	LRG	STYLUS
26	NO	YES	NOKEYS	NO	TRKBALL	PORT	YES	SML	STYLUS
27	YES	YES	QWERTY	NO	WHEEL	PORT	NO	NORM	STYLUS
28	YES	YES	QWERTY	NO	NDEF	PORT	NDEF	NDEF	FNGR
29	YES	YES	NDEF	NO	WHEEL	PORT	NDEF	SML	FNGR
30	YES	YES	NDEF	NO	WHEEL	PORT	NDEF	MASK	STYLUS
31	YES	YES	QWERTY	NO	WHEEL	PORT	NDEF	MASK	STYLUS
32	YES	YES	NOKEYS	NO	WHEEL	PORT	NDEF	NORM	STYLUS
33	YES	YES	12KEY	NO	TRKBALL	PORT	NDEF	LRG	STYLUS
34	YES	YES	NDEF	NO	WHEEL	PORT	NDEF	LRG	STYLUS
35	YES	YES	NDEF	NO	NDEF	PORT	NDEF	MASK	STYLUS
36	YES	YES	NDEF	NO	NONAV	PORT	NDEF	LRG	STYLUS

As we can see from the above table, we have 36 test cases from pairwise testing. This is significantly less than the 172800 cases from combinatorial testing.

Q4

The grammar can be tested with a single test case. An example of such a test case is

$$abcdefghijklmnopqrstuvwxyz + A - 1 * B/CDEFGHIJKLMNOPQRSTUVWXYZ. \tag{29}$$

i)

We have 4 different operators, 52 different letters, and 10 different digits. This means we have $4 + 52 + 10 = 66$ possible symbols.

ii)

⟨expr⟩ has 3 permutations, ⟨id⟩ has 2 permutations, and ⟨num⟩ has 2 permutations. This means we have $3 + 2 + 2 + 66 = 73$ cases for production coverage.

iii)

We can't perform derivation coverage because ⟨id⟩::=⟨letter⟩|⟨letter⟩⟨id⟩ leads to infinite tests.