cause -effect จะตัดกรณีที่ไม่มีทางเกิดขึ้นทิ้งไป

MULTI-VARIABLE TESTING

* ตัวแปรอิสระต่อกัน

Combination Chart

- In a combination test, we test several variables together. Each test explicitly sets values for each of the variables under test.

	Var 1	Var 2	Var 3	Var 4	Var 5
Test 1	Value 11	Vlue 12	Value 13	Value 14	Value 15
Test 2	Value 21	Value 22	Value 23	Value 24	Value 25
Test 3	Value 31	Value 32	Value 33	Value 34	Value 35
Test 4	Value 41	Value 42	Value 43	Value 44	Value 45
Test 5	Value 51	Value 52	Value 53	Value 54	Value 55
Test 6	Value 61	Value 62	Value 63	Value 64	Value 65

* ตัวแปรไหนที่เอามา Combine กัน

Challenges of multivariable testing

1. The space of possible tests is enormous
2. How to decide which variables to combine?
3. What IS the fault model?
4. How to figure out what the relationships among the variables actually are, in detail?

Combination Testing

- There are several approaches to combination testing:
  - ***Mechanical (or procedural).*** *The tester uses a routine procedure to* determine a good set of tests
  - ***Risk-based.*** *The tester combines test values (the values of each* variable) based on perceived risks associated with noteworthy combinations
  - ***Scenario-based.*** *The tester combines test values on the basis of* interesting stories created for the combinations

Domain testing

- In 1-dimensional testing, we run two tests for every boundary:
  - Test the boundary-valid value
  - Test the boundary-invalid value
- For example if X < 24 defines the boundary, we use X=24 (invalid) and X=24-delta (valid). We choose the smallest workable delta to minimize the possibility of an error hiding within the gap between 24 and 24-delta.
- In multi-dimensional testing, we start by testing each dimension on its own, reasonably thoroughly.
- Then we reduce the set of values to test per dimension, probably to the boundaries:
  - Too-low (TL), valid lowest (VL), valid biggest (VB), too big (TB)
  - where

    TL = VL-delta and TB=VB+delta

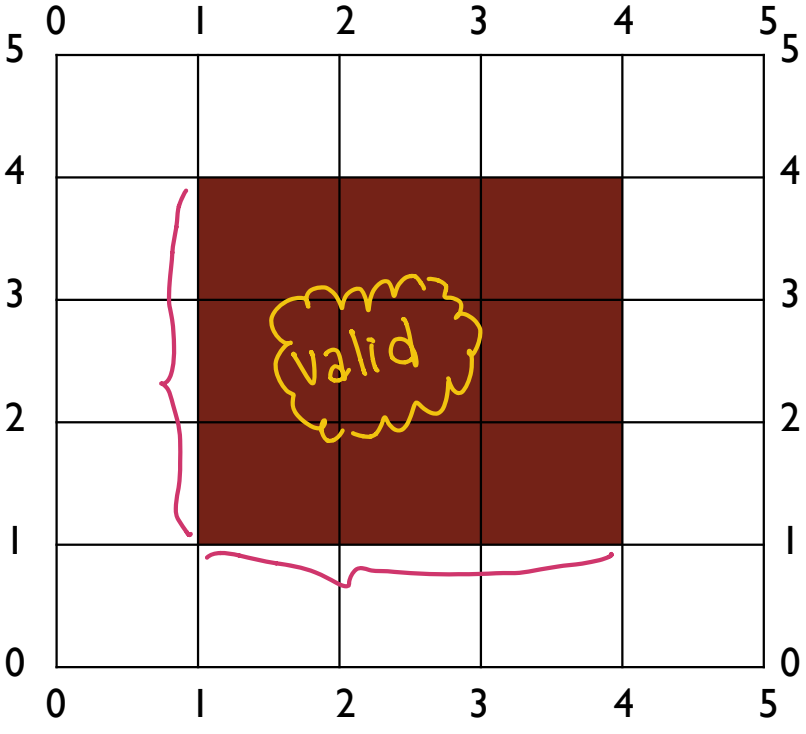
Defining the domain: 2 variables

2 ตัวแปร

- Suppose we have two numeric variables, V1 and V2.
- We analyze each variable in terms of its subdomains and boundaries. Thus we have for each variable:
  - V1: Too-low (TL), valid lowest (VL), valid biggest (VB), too big (TB)
  - V2: Too-low (TL), valid lowest (VL), valid biggest (VB), too big (TB)
  - Where we set
    - TL = VL-delta (smallest available difference between two numbers)
    - TH = VB+delta

* เลือก test case น้อยสุด แต่ค่าทั้งหมด ถูก test ครบด้วย *

Example: 2 variables



Consider the following domain definition:

$1 <= V1 < 4$

$1 <= V2 < 4$

Store data to 3 digits precision:

TL = 0.999

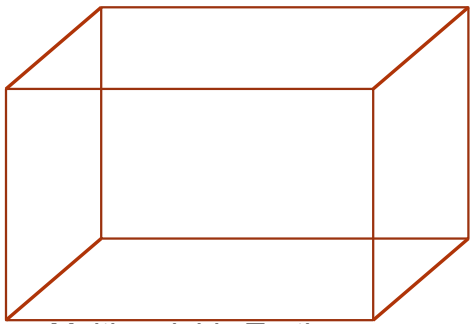
VL = 1.000

VB = 3.999

TB = 4.000

Defining the domain: 3 independent variables

- Suppose we have 3 numeric variables, V1, V2, V3.
- We analyze each variable in terms of its subdomains and boundaries.
- Thus we might have for each variable:
  - V1: Too-low (TL), valid lowest (VL), valid biggest (VB), too big (TB)
  - V2: Too-low (TL), valid lowest (VL), valid biggest (VB), too big (TB)
  - V3: Too-low (TL), valid lowest (VL), valid biggest (VB), too big (TB)



In this simple model, anything inside the box is a valid value, and anything outside the box is not.
(When we restrict ourselves to valid values, we are thinking inside the box.)

Mechanical approach #1

"Weak testing" version 1

	V1	V2	V3
Test 1	VL	VL	VL
Test 2	VB	VB	VB
Test 3	TL	TL	TL
Test 4	TB	TB	TB

Too-low (TL),
Lowest valid (VL),
Biggest valid (VB),
Too big (TB)

เอา error ไม่ดู่ว่า error อันในนอก

- We create enough tests to cover every value of every variable, once. If the largest number of values is N, we need only N tests
- Note the collisions of error cases. If Test 3 fails, is it because of the bad value of V1, V2, V3, or some combination of them?
- What bug do we expect to find in Test 3 that we would not find in a test of single dimension, with a bad value? Why do we need a combination?

"Weak testing" version 2

	V1	V2	V3
Test 1	VL	VL	VL
Test 2	VB	VB	VB
Test 3	TL	VL	VB
Test 4	VB	TL	VL
Test 5	VL	VL	TL
Test 6	TB	VB	VL
Test 7	VL	TB	VB
Test 8	VB	VL	TB

Too-low (TL),
Lowest valid (VL),
Biggest valid (VB),
Too big (TB)

- In this second version, we treat error cases specially:
  - Generate a core set of tests for "valid" (non-error) inputs
  - Generate additional tests in which one error case is allowed per test case. (Jorgensen calls this "weak robust equivalence class testing."
  - We might also add a few market-critical combinations

"Weak testing" version 3

- *All Singles* -

- Drop the error cases
  - test them in single-variable tests.
- Create tests only for valid values
  - Jorgensen calls this "weak normal equivalence class testing"
- Note the coverage that we do and do not achieve:
  - We have a test for every valid value of interest of every variable
  - We are not set up to detect interactions among variables.
- Here, for example, we check all minima together and all maxima.
  - Should we worry about Low-High combinations?

	V1	V2	V3
Test 1	VL	VL	VL
Test 2	VB	VB	VB

Mechanical approach #2

"Strong testing" version 1

- Test every combination of values of interest:
- Jorgensen calls this "strong robust equivalence class testing"

	V1	V2	V3
Test 1	TL	TL	TL
Test 2	TL	TL	VL
Test 3	TL	TL	VB
Test 4	TL	TL	TB
Test 5	TL	VL	TL
Test 6	TL	VL	VL
Test 7	TL	VL	VB
Test 8	TL	VL	TB
Test 9	TL	VB	TL
Test 10	TL	VB	VL

- This is part of the table. The complete table has 4 * 4 * 4 tests.
- In general, if N is the number of variables we test together and they have k_1, k_2 … k_N values, strong testing requires

$$k_1 \times k_2 \times \ldots \times k_N \text{ tests}$$

"Strong testing" version 2
All n-tuples

- Start with strong testing
  - But restrict the values of interest to valid values.
  - Jorgensen calls this "strong normal equivalence class analysis"
- Cover error cases in the one variable tests.
- If there are N independent dimensions, and we test only VL and VB for each, there are 2^N **tests**

More “strong testing"

- Another variation includes all valid-value combinations plus a separate set of combination tests in which one, some, or all variables have an error value.

- Tests that include several errors are of interest only if we think that multiple errors might have some type of cumulative effect.

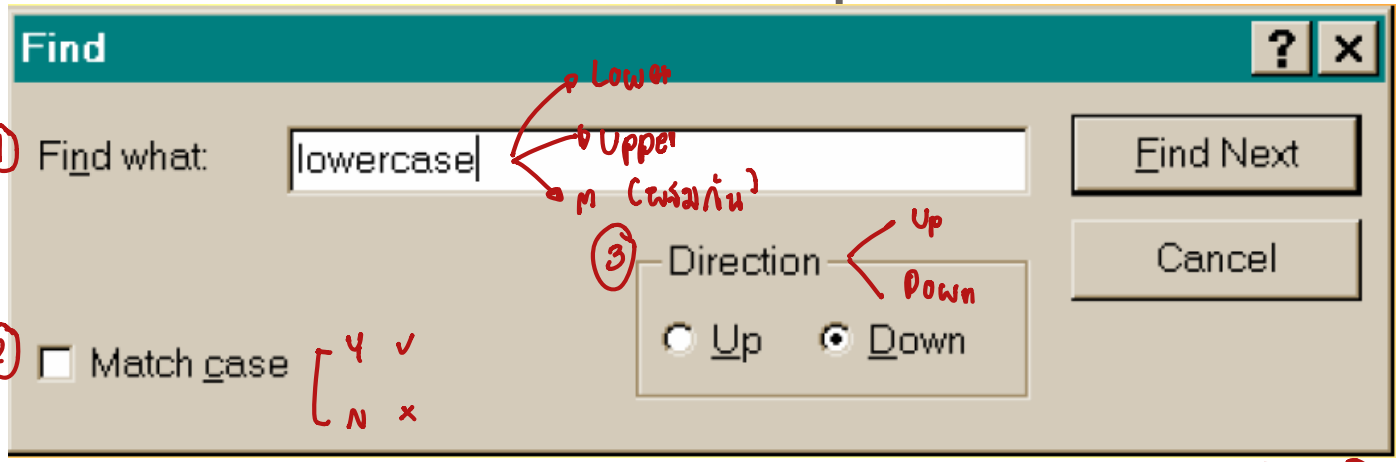
Mechanical approach #3
Combinatorial testing

- We have N variables
- We assume the variables are independent
  - A value of one variable does not change the effects or validity of values of other variables
- We consider only valid values of interest
  - An invalid value stops the test.
    (V1, V2, Bad, V4, V5) → what do we learn about V1, V2, V4 or V5?
    Anything in this test of interest other than "Bad" will be masked
- Our goal is to sample from the space of possible N-tuples in way that assures a minimum level of combination coverage:
  - All N-tuples → all combinations of valid values
  - All singles → all individual valid values
  - All pairs → all pairs of valid values
  - – All triples → all triplets of valid values

pairwise

* ตัวแปรต่าง อิสระต่อกัน *

Combinatorial Example (pair wise)

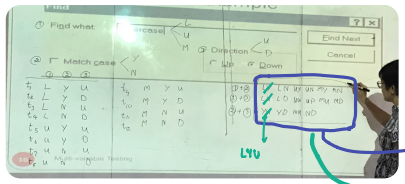


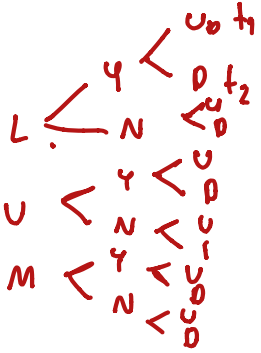
↓

ตรงๆ partition

* ทั้ง 3 อันนี้ ตัวแปร เป็นอิสระต่อกัน *

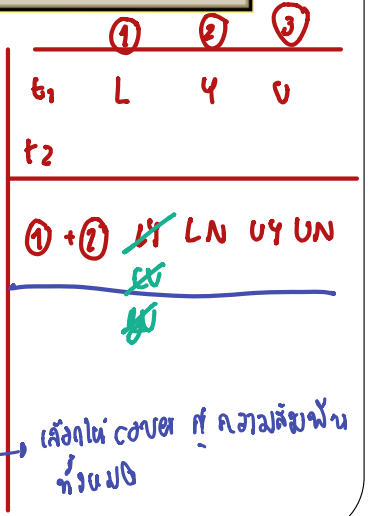
ดังนั้น => 3 × 2 × 2 => 12 testCase





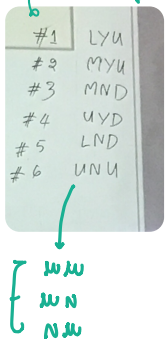
(12 คู่)

test Requirement



เลือกให้ cover ที่ความสัมพันธ์ทั้งหมด

Combinations Example



1. How many combinations of these three variables are possible?
2. List ALL the combinations of these three variables.
3. Now create combination tests that cover all possible pairs of values, but don't try to cover all possible triplets. List one such set.
4. How many test cases are in this set?

Combinations Example

1. How many combinations of these three variables are possible?
   - **Find what has 3 values (lowercase, mixed, caps) (L M C)**
   - **Match case has 2 values (Yes / No) (Y N)**
   - **Direction has 2 values (Up / Down) (U D)**

   So there will by 3 x 2 x 2 = 12 tests
2. List ALL the combinations of these three variables.

LY U	MY U	CY U
LY D	MY D	CY D
L N U	M N U	C N U
L N D	M N D	C N D

3. By the way, a more complete analysis will also consider whether the string is in the document or not. We'll add a 4th binary variable to the analysis soon.

สร้าง all-pairs โดยเลือกมาเป็นคู่ เพื่อลดค่า testcase → ทำ test requirement

Building an all-pairs table

- Label the columns with the variable names.
- List variables in descending order (of number of possible values)
- Each column will have repetition.
  - To determine how many times (rows in which) to repeat the first value before creating a row for the second multiply the number of variable values in column 1 x the number that will be in column 2
- In our example,
  - Find What has 3 values
  - Match Case has 2 values
  - So there will be at least 6 rows

Combination Testing

- Building an all-pairs combination table:
  - In the second column, list all the values of the second variable, skip the line, list the values again, etc. In our example, variable 2's possible values are U,D so the table looks like this so far

3 ค่า 2 ค่า

Find (L M C)	Match (Y N)
L	Y
L	N
M	Y
M	N
C	Y
C	N

① สร้าง เอาตัวมากสุด มาเป็น column หน้า

cover test requirement

✓ ① + ② แล้ว : LY LN MY MN CY CN

Combination Testing

- Building an all-pairs combination table:
  - Each section of the third column (think of LL as defining a section, MM as defining another) will have to contain every value of variable 3. Order the values such that the variables also make all pairs with variable 2.
  - Our variable 3 has two values, U and D
  - The third section can be filled in either way, and you might highlight it so that you can reverse it later. The decision (say D, U) is arbitrary.

Find (L M C)	Match (Y N)	Direct (U D)
L	Y	U
L	N	D
M	Y	D
M	N	U
C	Y	D
C	N	U

Combination Testing

- *Now that we've solved the 3-column exercise, let's try adding more variables. Each will have two values.*
- **Let's start by making this look a little more general**

ตัวแปร 3 ตัว → V1 V2 V3

A	D	F	① × ②
A	E	G	
B	D	G	① × ③
B	E	F	
C	D	G	② × ③
C	E	F	

The 4th column goes in easily:

- We start by making sure we hit all pairs of values of column 4 and column 2
- then all pairs of column 4 and column 3.

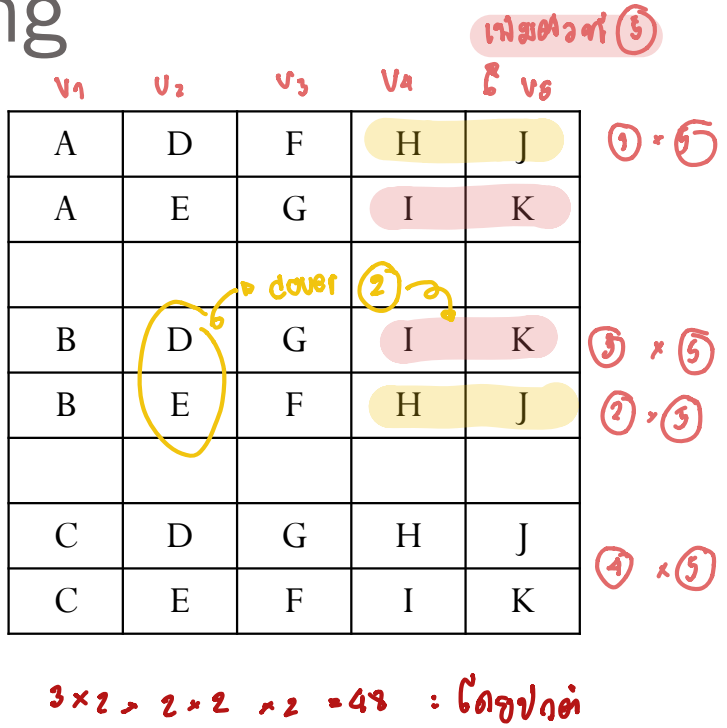
6 เพิ่มตัวแปรที่ 4

① × ④	A	D	F	H
	A	E	G	I
		↓ ยังไม่สลับ		
	B	D	G	I
② × ④	B	E	F	H
				เลยสลับอีกรอบ
	C	D	G	H
③ × ④	C	E	F	I

สลับกลับมาเพื่อให้ Combine

Combination Testing

- Watch this first attempt on column 5.
- We achieve all pairs of JK with columns 1, 2, and 3, but miss it for column 4.
- The most recent arbitrary choice was KJ in the 2nd section. (Once that was determined, we had to pick JK for the third in order to pair K with an F in the 3rd column.)
- So we will erase the last choice and try again:



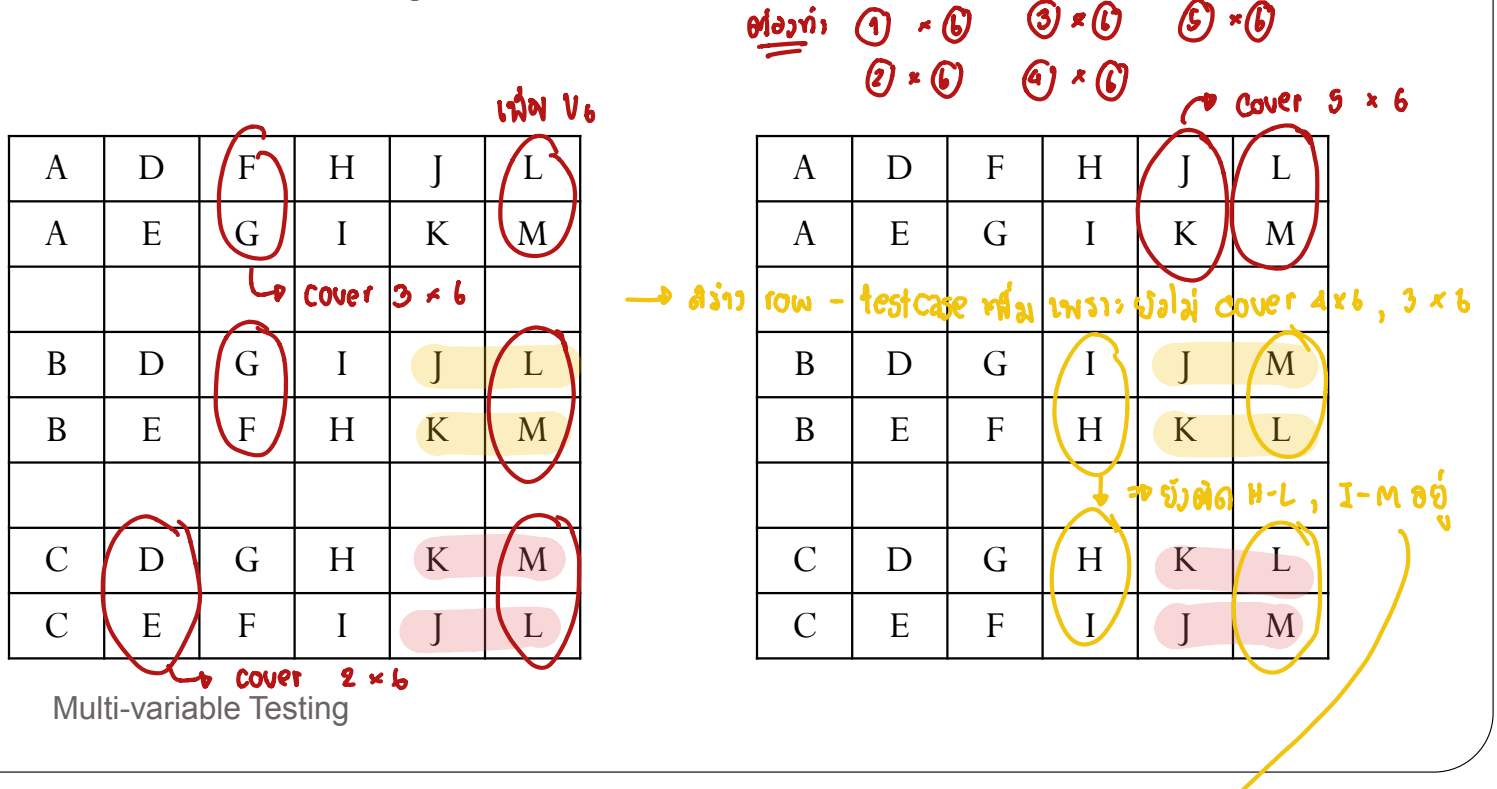
Combination Testing

- We flipped the last arbitrary choice (column 5, section 2, to JK from KJ) and erased the JK in section 3.
- We then fill in section 3 by checking for missing pairs.
- JK, JK, JK gives us three DJ, DJ, DJ pairs (2nd and 5th columns) so we have to flip to KJ for the third section.
- Now everything works

A	D	F	H	J
A	E	G	I	K
B	D	G	I	**J**
B	E	F	H	**K**
C	D	G	H	**K**
C	E	F	I	**J**

Combination Testing

- But when we add the next column, we see that we just can't achieve all pairs with 6 values. The first one works up to column 4 but then fails to get pair KL or JM. The next fails on HM and IL



A	D	F	H	J	L
A	E	G	I	K	M
B	D	G	I	J	L
B	E	F	H	K	M
C	D	G	H	K	M
C	E	F	I	J	L

A	D	F	H	J	L
A	E	G	I	K	M
B	D	G	I	J	M
B	E	F	H	K	L
C	D	G	H	K	L
C	E	F	I	J	M

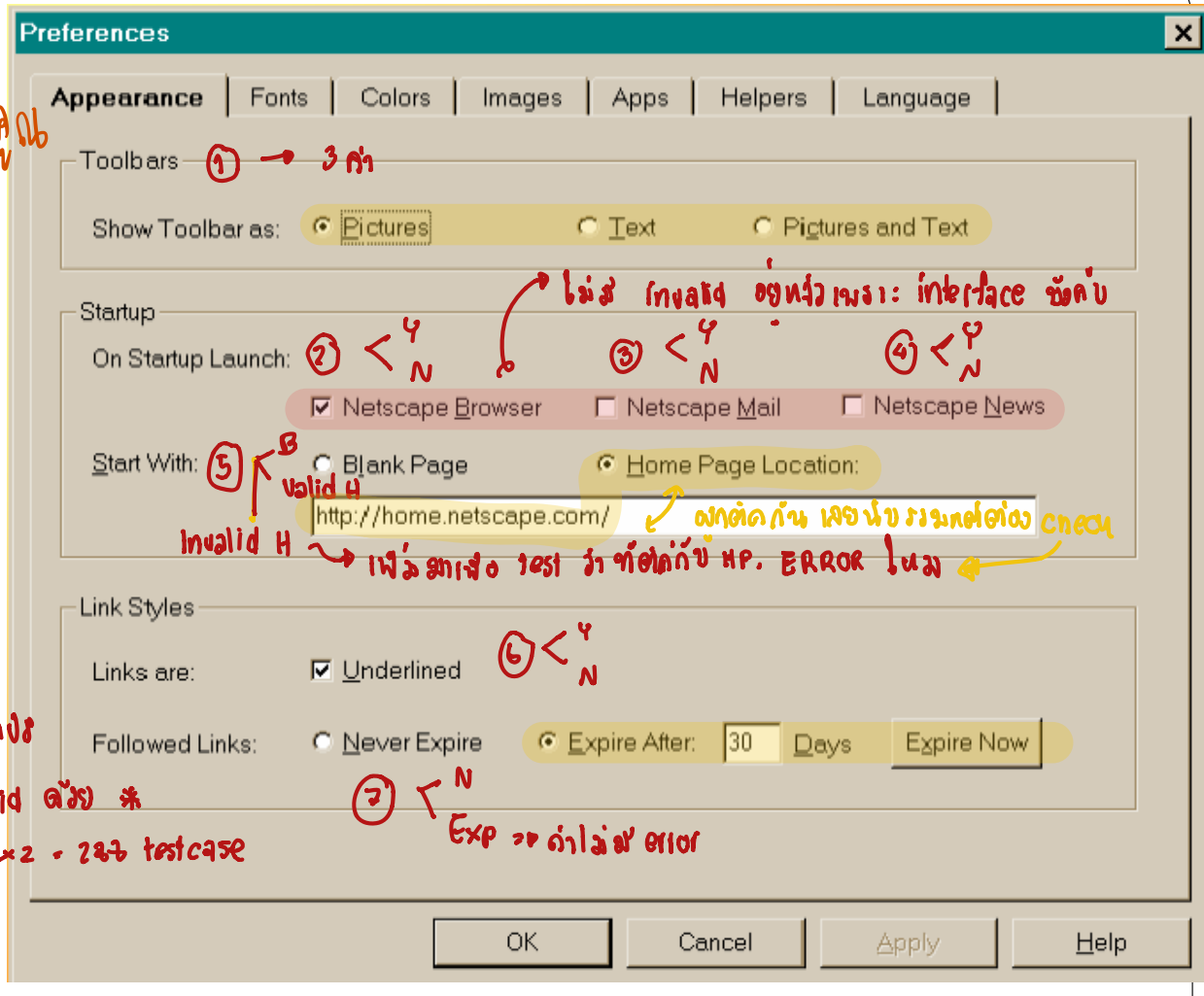
Combination Testing

- When all else fails, add rows. We need one for HM and one for IL, so add two rows. In general, we would need as many rows as the last column has values.
- The other values in the two rows are arbitrary, leave them blank and fill them in as needed when you add new columns. At the very end, fill the remaining blank ones with arbitrary values
- We have 8 tests instead of 3x2x2x2x2x2=96

A	D	F	H	J	L
A	E	G	I	K	M
A	E	G	H		M
B	D	G	I	J	M
B	E	F	H	K	L
			I		L
C	D	G	H	K	L
C	E	F	I	J	M

ตัวที่เป็นไปได้ทั้งหมด เอาทั้งหมดมาคูณกัน

Let's try this again on an old Netscape preference dialog



steps 1 → ต้องระบุทั้งหมด

* ปลายเปิด → test invalid ด้วย *

ได้ 3×2×2×2×3×2×2 = 288 testcase

2. → all si

* all single = 3 tests

The Netscape example

- If we just look at the Appearance tab of the Netscape Preferences dialog, we see the following variables:
  - **Toolbars** -- 3 choices (P,T, B) (***pictures, text or both)***
  - **On Startup Launch** --(***browser, mail, news). Each is an independent binary.***
    - Browser (Y, N)
    - Mail (Y, N)
    - News (Y, N)
  - **Start With** -- 3 choices (B,V,E) (***blank page, valid existing file, error (syntax) in the URL)*** (Many more cases are possible)
  - **Links** -- 2 choices (D,U) (***don't underline, underlined)***
  - **Followed Links** -- 2 choices (N,E) (***never expire, expire after 30 days) (Many more cases are possible)***

The Netscape example

- I simplified the combinations by simplifying the choices for two fields.
- In the **Start With** field, I used either a valid home page name or a blank. Some other tests for this field are:
  - Link to a different type of file, such as pdf
  - Link to a nonexistent file
  - Abbreviated URL, such as name.htm instead of http://
  - File on the local drive, the local network drive, or the remote drive
  - maximum length file names, maximum length paths
  - Note that a bad URL won't stop Netscape from starting, so we should be able to use an error case here without blocking testing of the other variables
- For combination testing, select a few of these that look like they might interact with other variables. Test the rest independently.
- Similarly for the **Expire After** field. This lets you enter the number of days to store links. If you use more than one value, use boundary cases, not all the numbers in the range.
- *In multi-variable testing, use partition analysis or other special values instead of testing all values in combination with all other variables' all values.*

All N-tuples

- We can create 3 x 2 x 2 x 2 x 3 x 2 x 2 = 288 different test cases by testing these variables in combination. Here are some examples, from the combination table.
- This is what Jorgensen would call "strong normal" testing.
- Strong because we test for faults triggered by a combination of conditions.
- Normal because we omit error cases.

	Toolbars PTB	On Startup, Browser Y/N	On Startup, Mail Y/N	On Startup, News Y/N	Start With BVE	Links DU	Followed NE
Test #1	P	Y	Y	Y	B	D	N
2	P	Y	Y	N	B	D	E
3	P	Y	N	Y	B	U	N
4	P	Y	N	N	B	U	E
5	P	Y	Y	Y	V	D	N
6	P	Y	Y	N	V	D	E
7	P	N	N	Y	V	U	N
8	P	N	N	N	V	U	E
9	P	N	Y	Y	E	D	N
10	P	N	Y	N	E	D	E
11	P	N	N	Y	E	U	N
12	P	N	N	N	E	U	E

Here are the 288 test cases. Every value of every variable is combined with every combination of the other variables.

all combination

• มี 7 ตัวแปร

1	PYYYBDN	PNYYBDN	TYYYBDN	TNYYBDN	BYYYBDN	BNYYBDN
2	PYYYVDE	PNYYVDE	TYYYVDE	TNYYVDE	BYYYVDE	BNYYVDE
3	PYYYEDN	PNYYEDN	TYYYEDN	TNYYEDN	BYYYEDN	BNYYEDN
4	PYYYBUE	PNYYBUE	TYYYBUE	TNYYBUE	BYYYBUE	BNYYBUE
5	PYYYVUN	PNYYVUN	TYYYVUN	TNYYVUN	BYYYVUN	BNYYVUN
6	PYYYEUE	PNYYEUE	TYYYEUE	TNYYEUE	BYYYEUE	BNYYEUE
7	PYYNBDN	PNYNBDN	TYYNBDN	TNYNBDN	BYYNBDN	BNYNBDN
8	PYYNVDE	PNYNVDE	TYYNVDE	TNYNVDE	BYYNVDE	BNYNVDE
9	PYYNEDN	PNYNEDN	TYYNEDN	TNYNEDN	BYYNEDN	BNYNEDN
10	PYYNBUE	PNYNBUE	TYYNBUE	TNYNBUE	BYYNBUE	BNYNBUE
11	PYYNVUN	PNYNVUN	TYYNVUN	TNYNVUN	BYYNVUN	BNYNVUN
12	PYYNEUE	PNYNEUE	TYYNEUE	TNYNEUE	BYYNEUE	BNYNEUE
13	PYYYBDE	PNYYBDE	TYYYBDE	TNYYBDE	BYYYBDE	BNYYBDE
14	PYYYVDN	PNYYVDN	TYYYVDN	TNYYVDN	BYYYVDN	BNYYVDN
15	PYYYEDE	PNYYEDE	TYYYEDE	TNYYEDE	BYYYEDE	BNYYEDE
16	PYYYBUN	PNYYBUN	TYYYBUN	TNYYBUN	BYYYBUN	BNYYBUN
17	PYYYVUE	PNYYVUE	TYYYVUE	TNYYVUE	BYYYVUE	BNYYVUE
18	PYYYEUN	PNYYEUN	TYYYEUN	TNYYEUN	BYYYEUN	BNYYEUN
19	PYYNBDE	PNYNBDE	TYYNBDE	TNYNBDE	BYYNBDE	BNYNBDE
20	PYYNVDN	PNYNVDN	TYYNVDN	TNYNVDN	BYYNVDN	BNYNVDN
21	PYYNEDE	PNYNEDE	TYYNEDE	TNYNEDE	BYYNEDE	BNYNEDE
22	PYYNBUN	PNYNBUN	TYYNBUN	TNYNBUN	BYYNBUN	BNYNBUN
23	PYYNVUE	PNYNVUE	TYYNVUE	TNYNVUE	BYYNVUE	BNYNVUE
24	PYYNEUN	PNYNEUN	TYYNEUN	TNYNEUN	BYYNEUN	BNYNEUN
25	PYNYBDE	PNNYBDE	TYNYBDE	TNNYBDE	BYNYBDE	BNNYBDE
26	PYNYVDN	PNNYVDN	TYNYVDN	TNNYVDN	BYNYVDN	BNNYVDN
27	PYNYEDE	PNNYEDE	TYNYEDE	TNNYEDE	BYNYEDE	BNNYEDE
28	PYNYBUN	PNNYBUN	TYNYBUN	TNNYBUN	BYNYBUN	BNNYBUN
29	PYNYVUE	PNNYVUE	TYNYVUE	TNNYVUE	BYNYVUE	BNNYVUE
30	PYNYEUN	PNNYEUN	TYNYEUN	TNNYEUN	BYNYEUN	BNNYEUN
31	PYNNBDE	PNNNBDE	TYNNBDE	TNNNBDE	BYNNBDE	BNNNBDE
32	PYNNVDN	PNNNVDN	TYNNVDN	TNNNVDN	BYNNVDN	BNNNVDN
33	PYNNEDE	PNNNEDE	TYNNEDE	TNNNEDE	BYNNEDE	BNNNEDE
34	PYNNBUN	PNNNBUN	TYNNBUN	TNNNBUN	BYNNBUN	BNNNBUN
35	PYNNVUE	PNNNVUE	TYNNVUE	TNNNVUE	BYNNVUE	BNNNVUE
36	PYNNEUN	PNNNEUN	TYNNEUN	TNNNEUN	BYNNEUN	BNNNEUN
37	PYNYBDN	PNNYBDN	TYNYBDN	TNNYBDN	BYNYBDN	BNNYBDN
38	PYNYVDE	PNNYVDE	TYNYVDE	TNNYVDE	BYNYVDE	BNNYVDE
39	PYNYEDN	PNNYEDN	TYNYEDN	TNNYEDN	BYNYEDN	BNNYEDN
40	PYNYBUE	PNNYBUE	TYNYBUE	TNNYBUE	BYNYBUE	BNNYBUE
41	PYNYVUN	PNNYVUN	TYNYVUN	TNNYVUN	BYNYVUN	BNNYVUN
42	PYNYEUE	PNNYEUE	TYNYEUE	TNNYEUE	BYNYEUE	BNNYEUE
43	PYNNBDN	PNNNBDN	TYNNBDN	TNNNBDN	BYNNBDN	BNNNBDN
44	PYNNVDE	PNNNVDE	TYNNVDE	TNNNVDE	BYNNVDE	BNNNVDE
45	PYNNEDN	PNNNEDN	TYNNEDN	TNNNEDN	BYNNEDN	BNNNEDN
46	PYNNBUE	PNNNBUE	TYNNBUE	TNNNBUE	BYNNBUE	BNNNBUE
47	PYNNVUN	PNNNVUN	TYNNVUN	TNNNVUN	BYNNVUN	BNNNVUN
48	PYNNEUE	PNNNEUE	TYNNEUE	TNNNEUE	BYNNEUE	BNNNEUE

Mu

All N-tuples

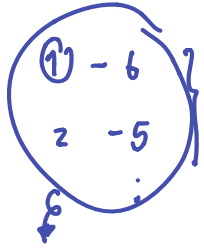
When creating a combination table, I strongly recommend that you order the columns from the variable with the most values to the variable with the least.

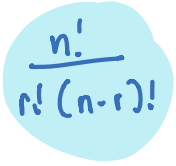
	Toolbars PTB	Start With BVE	On Startup, Browser Y/N	On Startup, Mail Y/N	On Startup, News Y/N	Links DU	Followed NE
Test #1	P	B	Y	Y	Y	D	N
2	P	B	Y	Y	N	D	E
3	P	B	Y	N	Y	U	N
4	P	B	Y	N	N	U	E
5	P	V	Y	Y	Y	D	N
6	P	V	Y	Y	N	D	E
7	P	V	N	N	Y	U	N
8	P	V	N	N	N	U	E
9	P	E	N	Y	Y	D	N
10	P	E	N	Y	N	D	E
11	P	E	N	N	Y	U	N
12	P	E	N	N	N	U	E

All Singles

- There are
  $3+3+2+2+2+2+2=16$
  different individual (single) values of interest.
- We can cover them in 3 tests

	Toolbars PTB	Start With BVE	On Startup, Browser Y/N	On Startup, Mail Y/N	On Startup, News Y/N	Links DU	Followed NE
Test #1	P	B	Y	Y	Y	D	N
2	T	V	N	N	N	U	E
3	B	E	Y	Y	Y	D	N





What about pairs?

- To simplify this, many testers would test variables in pairs, each test involving only 2 values.
- There are 109 pairs in our example.
- Testing only 2 variables at once is an inefficient form of combination testing.
- One test that combines 7 variables incorporates 21 tests of pairs of variables.

	Combo	Toolbars PTB	Start With B/E	On Startup, Browser Y/N	On Startup, Mail Y/N	On Startup, News Y/N	Links D/U	Followed N/E
9	Pair 1	P	B					
6	2	P		Y				
6	3	P			Y			
6	4	P				Y		
6	5	P					D	
6	6	P						N
6	7		B	Y				
6	8		B		Y			
6	9		B			Y		
6	10		B				D	
6	11		B					N
4	12			Y	Y			
4	13			Y		Y		
4	14			Y			D	
4	15			Y				N
4	16				Y	Y		
4	17				Y		D	
4	18				Y			N
4	19					Y	D	
4	20					Y		N
4	21						D	N

9 + 6×10 + 4×10 → 109

Combinatorics

- "Combinatorics is, loosely, the science of counting. This is the area in mathematic in which we study families of sets (usually) finite with certain characteristic arrangements of their elements or subsets, and ask what combinations are possible, and how many there are. This includes numerous quite elementary topics, such as enumerating all possible permutations or combinations of a finite set."
- In combinatorial testing, we test many variables together as an efficient way of testing many of the combinations of those variables (e.g. testing 7 variables together in one test captures 7C2 = 21 tests of the pairs).
- So how many tests would we have to run to cover ***all*** *the pairs?*

All pairs for Netscape

- We can cover all 109 pairs inside 9 tests

	Toolbars PTB	Start With BVE	On Startup, Browser Y/N	On Startup, Mail Y/N	On Startup, News Y/N	Links DU	Followed NE
Test #1	P	B	Y	Y	Y	D	N
2	P	V	Y	N	N	D	E
3	P	E	N	Y	Y	U	E
4	T	E	Y	N	Y	U	N
5	T	B	N	Y	N	D	E
6	T	V	N	N	Y	D	N
7	B	B	Y	N	N	U	E
8	B	V	N	Y	Y	U	E
9	B	E	N	N	N	D	N

All pairs for Netscape

- Let's work it through.
- We start with the first two variables (biggest and second biggest number of values of interest.)
- Here are all the pairs of those two variables.
- There are 3x3 = 9 of them

	Toolbars PTB	Start with VBE
Test#1	P	B
2	P	V
3	P	E
4	T	E
5	T	B
6	T	V
7	B	B
8	B	V
9	B	E

All pairs for Netscap

Add the next variable

- We need all the pairs of the
  - 1st and 2nd variables
  - 1st and 3rd variables
  - 2nd and 3rd variables
- We already have the pairs for 1st & 2nd variables
- For the 1st and 3rd, we need
  - a Y with a P, an N with a P,
  - a Y with a T, an N with a T,
  - a Y with a B and an N with a B.
- The values of the 3rd variable for the other cases don't matter for 1st & 3rd, but they might matter for 2nd & 3rd.

	Toolbars PTB	Start with VBE	On Startup, Brower Y/N
Test#1	P	B	Y
2	P	V	N
3	P	E	
4	T	E	N
5	T	B	Y
6	T	V	
7	B	B	
8	B	V	
9	B	E	

All pairs for Netscap

Add the 4th variable. We have the pairs for the 1st 3, we just have to work in the 4th.

	Toolbars PTB	Start with VBE	On Startup, Brower Y/N	On Startup, Mail Y/N
Test#1	P	B	Y	Y
2	P	V	N	N
3	P	E	Y	Y
4	T	E	N	N
5	T	B	Y	Y
6	T	V	N	N
7	B	B	Y	N
8	B	V	N	Y
9	B	E	Y	N

All pairs for Netscape

Keep going, through the 7th variable.

	Toolbars PTB	Start with VBE	On Startup, Brower Y/N	On Startup, Mail Y/N	On Startup, News Y/N	Links DU	Followed NE
Test#1	P	B	Y	Y			
2	P	V	N	N			
3	P	E	Y	Y			
4	T	E	N	N			
5	T	B	Y	Y			
6	T	V	N	N			
7	B	B	Y	N			
8	B	V	N	Y			
9	B	E	Y	N			

All pairs

Reminder of a common misconception.
The lower bound on the number of rows is the number of values of column 1 times column 2 but we often need more than that

#	01	01	01
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

A	01	01	01	01
1	0	0	0	
2	0	1	1	
3	1	0	1	
4	1	1	0	

B	01	01	01	01
1	0	0	0	
2	0	1	1	
3	1	0	1	
4	1	1	0	

C	01	01	01	01
1	0	0	0	
2	0	1	1	
3	1	0	1	
4	1	1	0	

	01	01	01	01
1	0	0	0	
2	0	1	1	
3	1	0	1	
4	1	1	0	

	01	01	01	01
1	0	0	0	
2	0	1	1	
3	1	0	1	
4	1	1	0	
5				
6				

1 equivalent partil

2 boundary

3. clause effect