CMPT 473
Software Testing, Reliability and Security

Scale
& Combinatorial Testing

Nick Sumner

Recall from last time...

- Consider our triangle classifier
  - Takes 3 integers for sides 1, 2, and 3

Characteristic	b1	b2	b3
Side 1 <?> 0	Side 1 > 0	Side 1 = 0	Side 1 < 0
Side 2 <?> 0	Side 2 > 0	Side 2 = 0	Side 2 < 0
Side 3 <?>0	Side 3 > 0	Side 3 = 0	Side 3 < 0

3 guiding questions...

Recall from last time…

- Consider our triangle classifier
  - Takes 3 integers for sides 1, 2, and 3

Characteristic	b1	b2	b3
Side 1 <?> 0	Side 1 > 0	Side 1 = 0	Side 1 < 0
Side 2 <?> 0	Side 2 > 0	Side 2 = 0	Side 2 < 0
Side 3 <?>0	Side 3 > 0	Side 3 = 0	Side 3 < 0

How many tests does this create?

Recall from last time…

- Consider our triangle classifier
  - Takes 3 integers for sides 1, 2, and 3

Characteristic	b1	b2	b3
Side 1 <?> 0	Side 1 > 0	Side 1 = 0	Side 1 < 0
Side 2 <?> 0	Side 2 > 0	Side 2 = 0	Side 2 < 0
Side 3 <?>0	Side 3 > 0	Side 3 = 0	Side 3 < 0

How many tests does this create?

What will this test well?
What won't this test well?

Recall from last time...

- Consider our triangle classifier
  - Takes 3 integers for sides 1, 2, and 3

Characteristic	b1	b2	b3	b4
Value of side 1	Side 1 > 1	Side 1 = 1	Side 1 = 0	Side 1 < 0
Value of side 2	Side 2 > 1	Side 2 = 1	Side 2 = 0	Side 2 < 0
Value of side 3	Side 3 > 1	Side 3 = 1	Side 3 = 0	Side 3 < 0

How many tests now?

What is the scale?

Suppose inputs or characteristics I_1, I_2, I_3, …, I_n

- How does the number of tests change?

What is the scale?

Suppose inputs or characteristics $I_1, I_2, I_3, \ldots, I_n$

- How does the number of tests change?
- $|D_1| * |D_2| * |D_3| * \ldots * |D_n| = k^n$
- This is *combinatorial explosion*

What is the scale?

Suppose inputs or characteristics $I_1, I_2, I_3, \ldots, I_n$

- How does the number of tests change?
- $|D_1| * |D_2| * |D_3| * \ldots * |D_n| = k^n$
- This is *combinatorial explosion*

What does it mean in practice?

- Find command: 4x3x3x3x3x3x2 = 1944 tests

What is the scale?

Suppose inputs or characteristics $I_1, I_2, I_3, \ldots, I_n$

- How does the number of tests change?
- $|D_1| * |D_2| * |D_3| * \ldots * |D_n| = k^n$
- This is *combinatorial explosion*

What does it mean in practice?

- Find command: 4x3x3x3x3x3x2 = 1944 tests
- Website generator: > 30 → > 1 billion tests

What is the scale?

Suppose inputs or characteristics $I_1, I_2, I_3, \ldots, I_n$

- How does the number of tests change?
- $|D_1| * |D_2| * |D_3| * \ldots * |D_n| = k^n$
- This is *combinatorial explosion*

What does it mean in practice?

- Find command: 4x3x3x3x3x3x2 = 1944 tests
- Website generator: > 30 → > 1 billion tests

Too many to maintain!

Too many to reasonably even create!

How do we cope with scale?

- What did the input partitioning do?

How do we cope with scale?

- What did the input partitioning do?
  - Constraints

```
Pattern Size:
  Empty                                [Property Empty]
  Single character                     [Property NonEmpty]
  Many characters                      [Property NonEmpty]
  Longer than any line in the file     [Property NonEmpty]
```

```
Quoting:
  Pattern is quoted                    [Property Quoted]
  Pattern is not quoted                [If NonEmpty]
  Pattern is improperly quoted         [If NonEmpty]
```

How do we cope with scale?

- What did the input partitioning do?
  - Constraints
  - [property] to identify rules for useful tests
  - [error] to identify when 1 test for a block is sufficient

```
Pattern Size:
  Empty                                    [Property Empty]
  Single character                         [Property NonEmpty]
  Many characters                          [Property NonEmpty]
  Longer than any line in the file         [Property NonEmpty]
```

```
Quoting:
  Pattern is quoted                        [Property Quoted]
  Pattern is not quoted                    [If NonEmpty]
  Pattern is improperly quoted             [If NonEmpty]
```

How do we cope with scale?

- What did the input partitioning do?
  - Constraints
  - [property] to identify rules for useful tests
  - [error] to identify when 1 test for a block is sufficient
- What else might we do?

How do we cope with scale?

- What did the input partitioning do?
  - Constraints
  - [property] to identify rules for useful tests
  - [error] to identify when 1 test for a block is sufficient
- What else might we do?
  - Not test as thoroughly (sampling)

Why might this be okay?

How do we cope with scale?

- What did the input partitioning do?
  - Constraints
  - [property] to identify rules for useful tests
  - [error] to identify when 1 test for a block is sufficient
- What else might we do?
  - Not test as thoroughly (sampling)
  - Identify related variables/domains & test together

Why might this lead to fewer tests?

Choosing Combinations

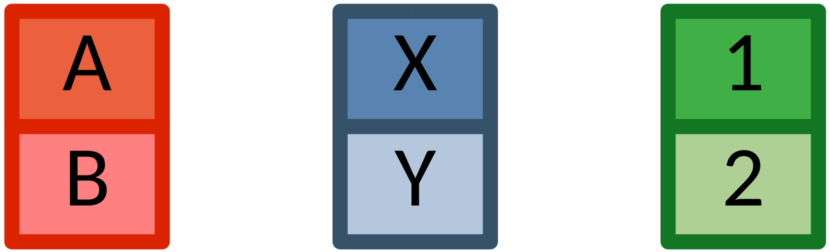
Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried

Choosing Combinations

Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried

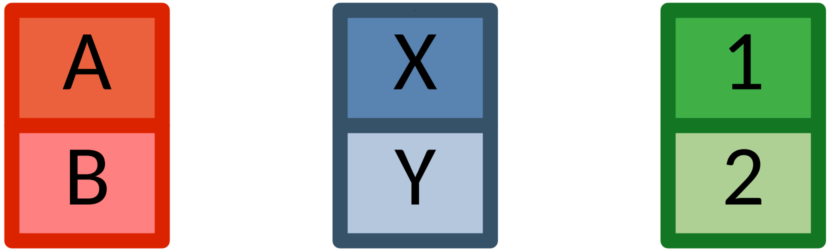


Choosing Combinations

Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried

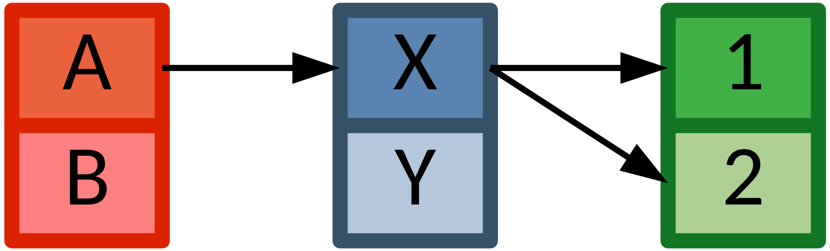
Adequate Tests:



Choosing Combinations

Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried

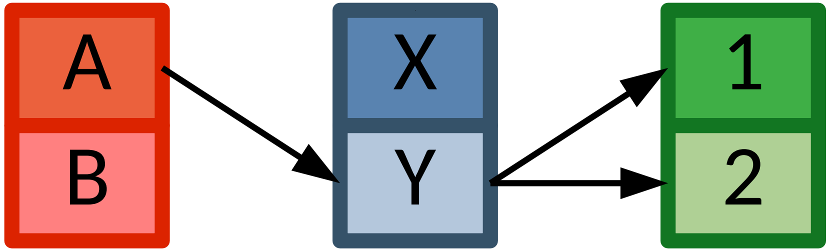


Adequate Tests:
AX1, AX2

Choosing Combinations

Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried



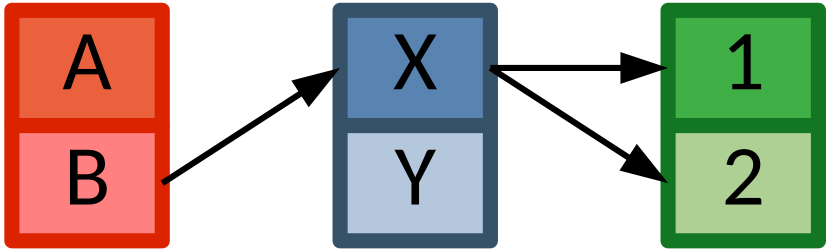
Adequate Tests:

AX1, AX2
AY1, AY2

Choosing Combinations

Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried



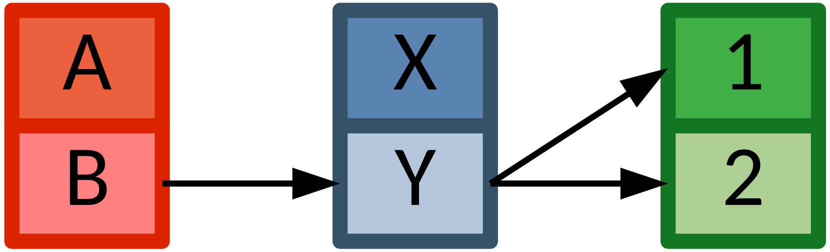
Adequate Tests:

AX1, AX2
AY1, AY2
BX1, BX2

Choosing Combinations

Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried



Adequate Tests:

AX1, AX2
AY1, AY2
BX1, BX2
BY1, BY2

Choosing Combinations

Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried
  - Leaps headfirst into combinatorial explosion: **k^n tests**

Choosing Combinations

Several possible strategies to consider:

- All Combinations
  - Every combination of every block is tried
  - Leaps headfirst into combinatorial explosion: k^n tests

But is it inherently bad?

Combinations – Each Choice

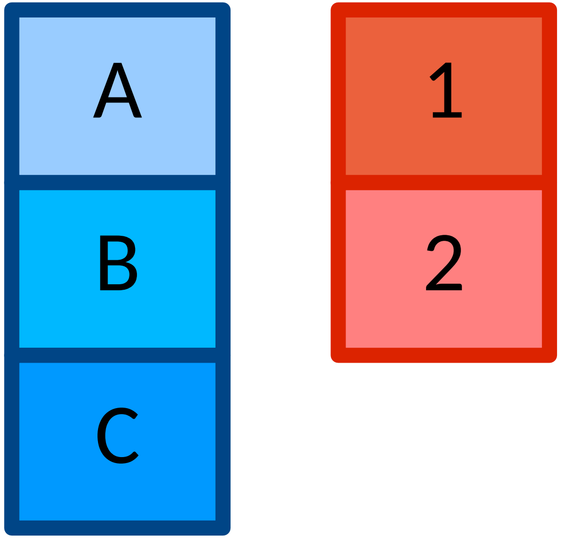
- How can we minimize the number of tests and still test each block?

Combinations – Each Choice

- How can we minimize the number of tests and still test each block?
- Each Choice
  - 1 value from each block is used in at least 1 test

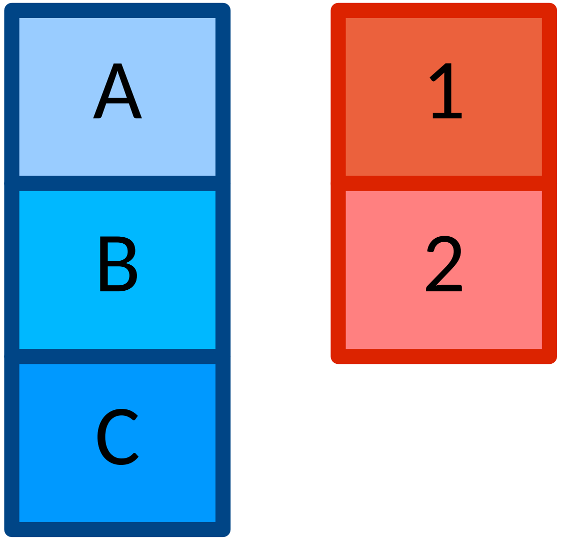
Combinations – Each Choice

- How can we minimize the number of tests and still test each block?
- Each Choice
  - 1 value from each block is used in at least 1 test



Combinations – Each Choice

- How can we minimize the number of tests and still test each block?
- Each Choice
  - 1 value from each block is used in at least 1 test



Adequate Tests:
(A,1), (B,2), (C,1)

Combinations – Each Choice

- How can we minimize the number of tests and still test each block?
- Each Choice
  - 1 value from each block is used in at least 1 test

What does this look like for the triangle classifier?

Combinations – Each Choice

- How can we minimize the number of tests and still test each block?
- Each Choice
  - 1 value from each block is used in at least 1 test

What does this look like for the triangle classifier?

Are these tests *good*? Why?

Combinations – Each Choice

- How can we minimize the number of tests and still test each block?
- Each Choice
  - 1 value from each block is used in at least 1 test

How many tests? Why?

Combinations – Each Choice

- How can we minimize the number of tests and still test each block?
- Each Choice
  - 1 value from each block is used in at least 1 test
  - # tests = maximum number of blocks

How many tests? Why?

Combinations – ???

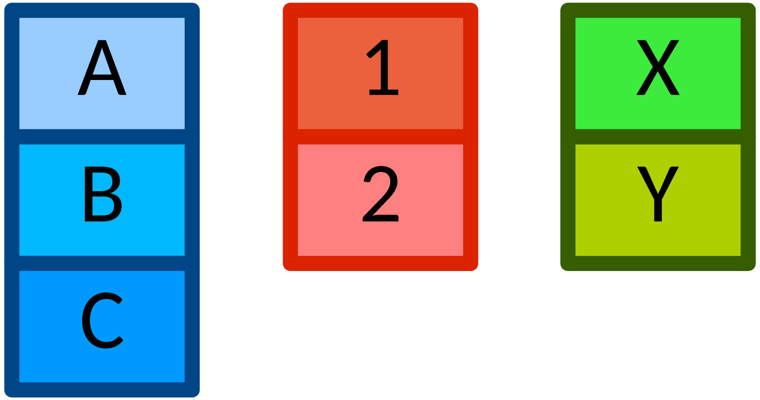
- Can we come up with a compromise?

Combinations – ???

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block

Combinations – ???

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block

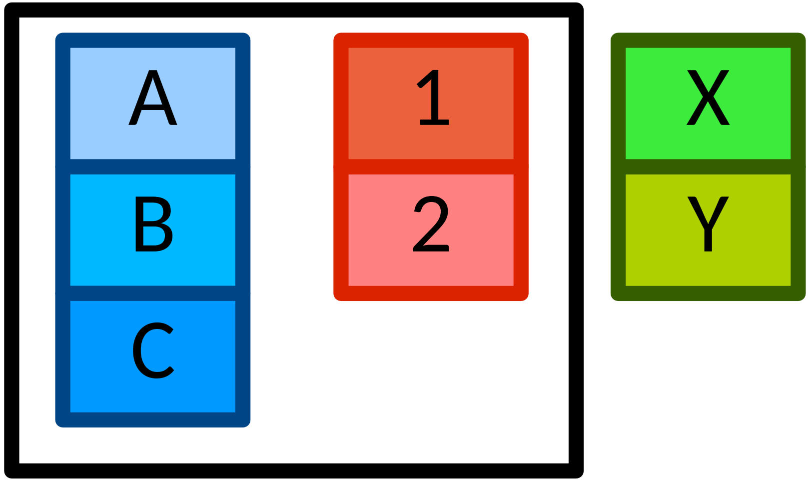


Adequate Tests:
(A,1,*), (A,2,*)
(B,1,*), (B,2,*)
(C,1,*), (C,2,*)

Combinations – ???

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block

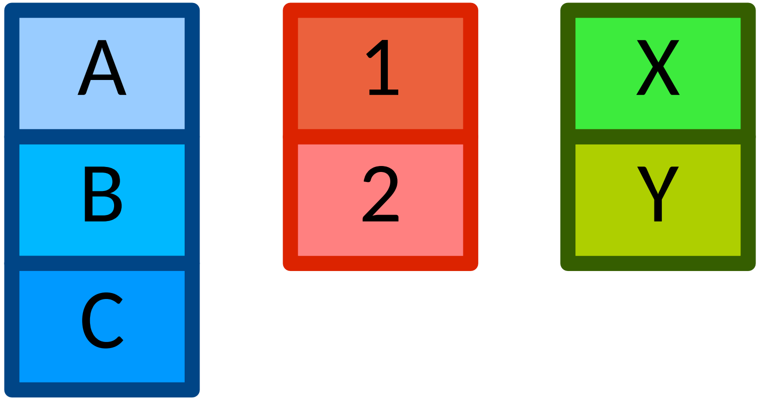
all combinations of two



Adequate Tests:
(A,1,*), (A,2,*)
(B,1,*), (B,2,*)
(C,1,*), (C,2,*)

Combinations – ???

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block

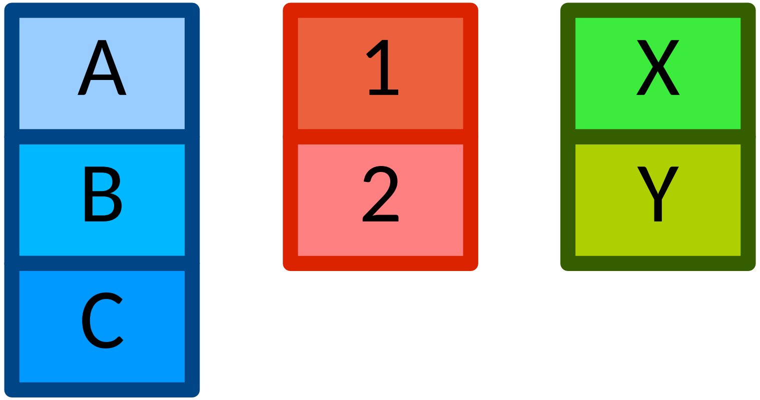


Adequate Tests:
(A,1,**X**), (A,2,**Y**)
(B,1,**Y**), (B,2,**X**)
(C,1,*), (C,2,*)

Fill in X and Y to make sure all pairwise combos are tested!

Combinations – ???

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block



Adequate Tests:
(A,1,X), (A,2,Y)
(B,1,Y), (B,2,X)
(C,1,*), (C,2,*)

What should the last two be?

Combinations – ???

- Can we come up with a compromise?

Combinations – Pairwise

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block

What does this look like for the triangle classifier?

Combinations – Pairwise

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block

What does this look like for the triangle classifier?

Are these tests *good*? Why?

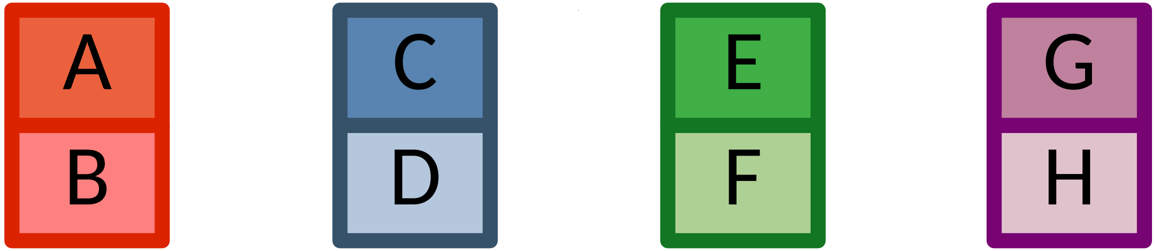
Combinations – Pairwise

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block

How many tests?

Combinations – Pairwise

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block



How many tests?

Combinations – Pairwise

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block
  - #tests ≥ product of 2 largest domain partitionings

How many tests?

Combinations – Pairwise

- Can we come up with a compromise?
- Pairwise
  - 1 value for each block combined with 1 value for each other block
  - #tests ≥ product of 2 largest domain partitionings

How many tests?

Expected on the order of $|D_1| * |D_2| * \log(n)$

Combinations – ???

- Can we extend this further?

Combinations – T-wise

- Can we extend this further?
- T-wise
  - 1 value from each block for each group of T characteristics

Combinations – T-wise

- Can we extend this further?
- T-wise
  - 1 value from each block for each group of T characteristics

How many tests?

Combinations – T-wise

- Can we extend this further?
- T-wise
  - 1 value from each block for each group of T characteristics
  - #tests ≥ product of T largest domain partitionings

How many tests?

Combinations – T-wise

- Can we extend this further?
- T-wise
  - 1 value from each block for each group of T characteristics
  - #tests ≥ product of T largest domain partitionings

What happens as T increases?

Combinations – T-wise

- Can we extend this further?
- T-wise
  - 1 value from each block for each group of T characteristics
  - #tests ≥ product of T largest domain partitionings
  - Bounded by $(\text{max number of blocks})^T$
  - More expensive than pairs & uncertain gains

T is often called the *test strength*

Combinations – Base Choice

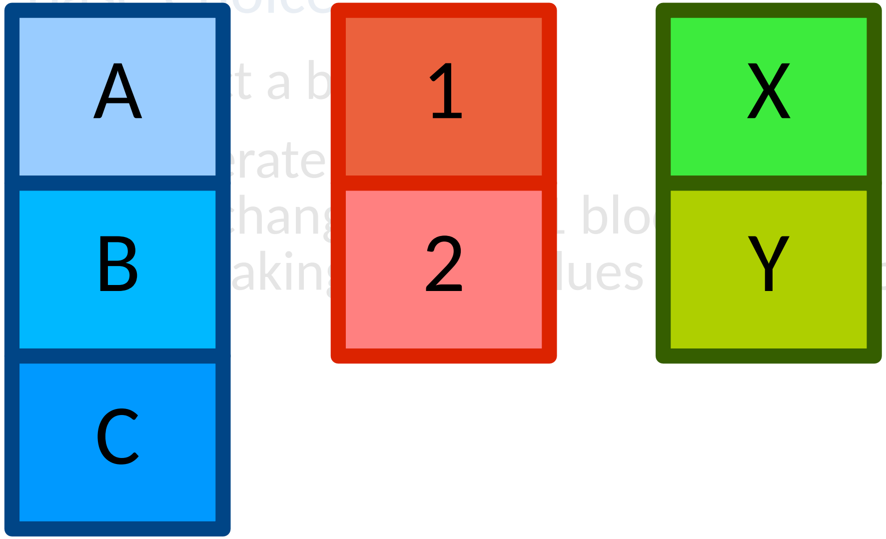
- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?
- Base Choice
  - Select a base test
  - Generate tests by
    changing only 1 block and
    taking other values from the base

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?
- Base Choice

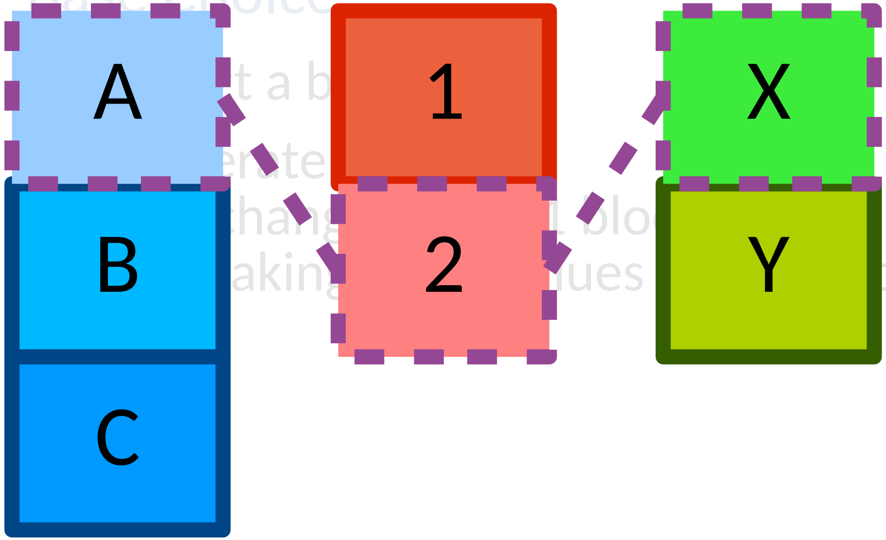


Base Test:

Adequate Tests:

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?
- Base Choice

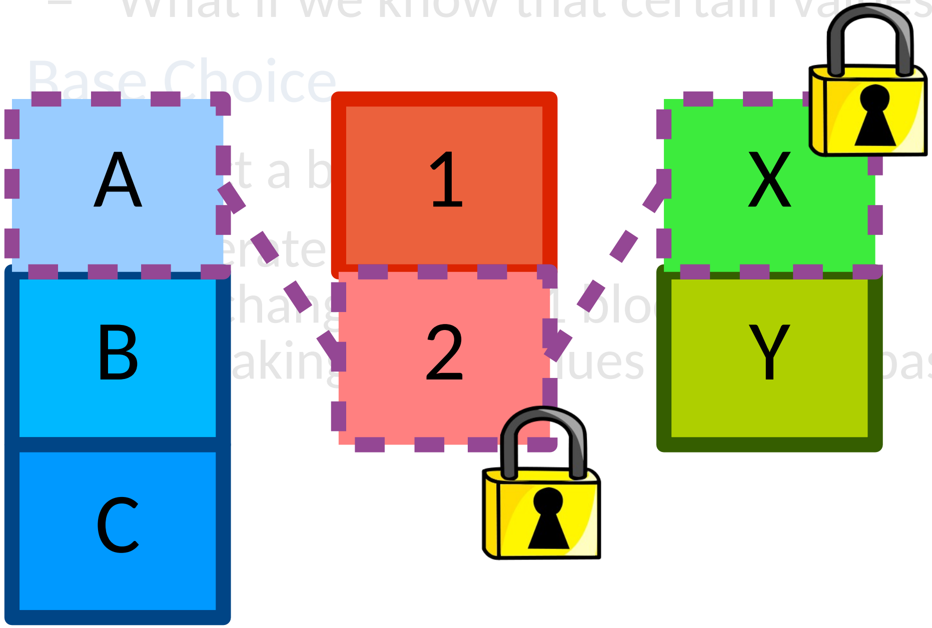


Base Test:
(A,2,X)

Adequate Tests:

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?
- Base Choice

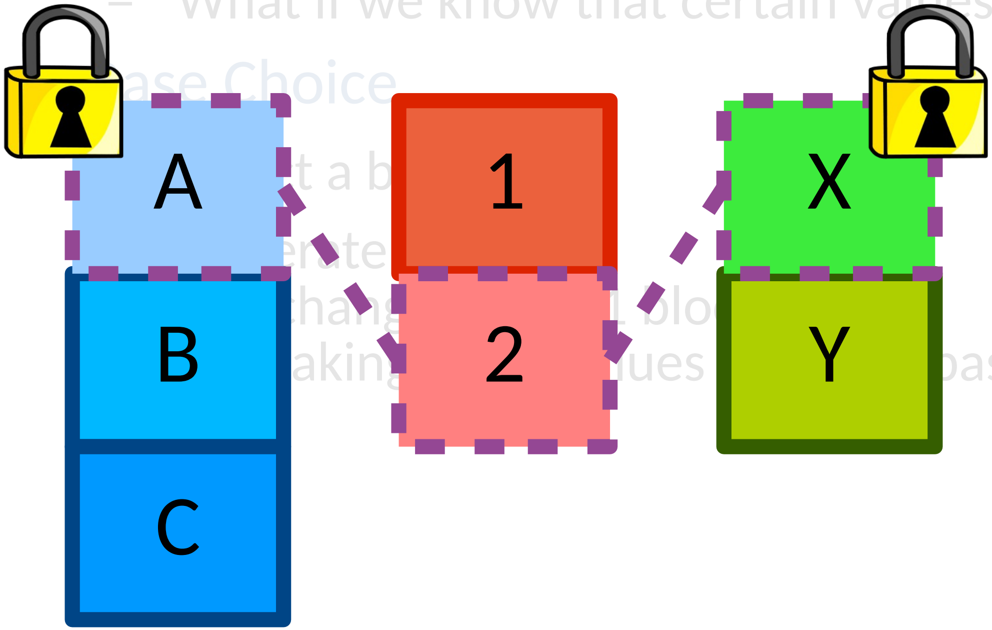


Base Test:
(A,2,X)

Adequate Tests:
(B,2,X), (C,2,X)

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?

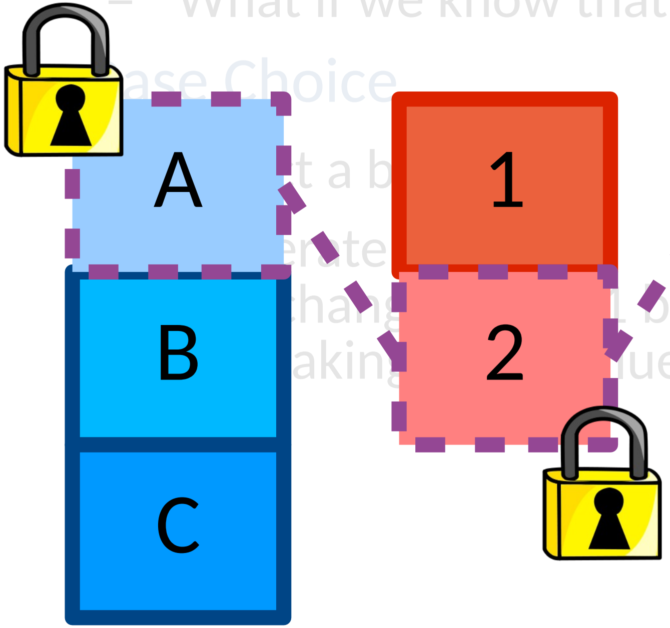


Base Test:
(A,2,X)

Adequate Tests:
(B,2,X), (C,2,X)
(A,1,X)

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?

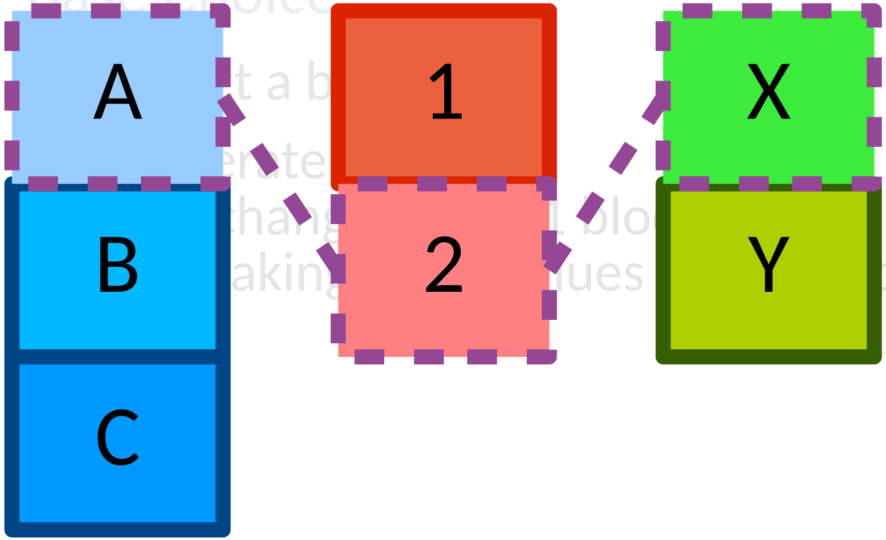


Base Test:
(A,2,X)

Adequate Tests:
(B,2,X), (C,2,X)
(A,1,X)
(A,2,Y)

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?
- Base Choice



Base Test:
(A,2,X)

Adequate Tests:
(B,2,X), (C,2,X)
(A,1,X)
(A,2,Y)

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?
- Base Choice
  - Select a base test
  - Generate tests by
    changing only 1 block and
    taking other values from the base

What does this look like for the triangle classifier?

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?
- Base Choice
  - Select a base test
  - Generate tests by
    changing only 1 block and
    taking other values from the base

How many tests?

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - What if we know that certain values are important?
- Base Choice
  - Select a base test
  - Generate tests by
    changing only 1 block and
    taking other values from the base
  - # tests = 1 base + 1 per each other block

How many tests?

$$1 + \sum |D_i - 1|$$

Base Choices

Which test to use as a base is crucial

Why? What if we choose poorly?

Base Choices

Which test to use as a base is crucial

- Must at least be *feasible*
  - Do the combined values create a valid run?

Base Choices

Which test to use as a base is crucial

- Must at least be *feasible*
  - Do the combined values create a valid run?

How might we select a base test?

Base Choices

Which test to use as a base is crucial

- Must at least be *feasible*
  - Do the combined values create a valid run?
- Guided by:
  - Most likely?
  - Simplest?
  - Smallest?
  - Etc.

Base Choices

Which test to use as a base is crucial

- Must at least be *feasible*
  - Do the combined values create a valid run?
- Guided by:
  - Most likely?
  - Simplest?
  - Smallest?
  - Etc.
- Decision must be well understood & well maintained

Combinations – ???

- Notice the pattern.
  - Can Base Choice be extended?

Combinations – ???

- Notice the pattern.
  - Can Base Choice be extended?
- *Multiple Base Choice*
  - Select 1 or more base characteristics
  - Generate base tests by using each at least once
  - Change 1 block at a time to an unselected one just as before

Combinations – ???

- Notice the pattern.
  - Can Base Choice be extended?
- *Multiple Base Choice*
  - Select 1 or more base characteristics
  - Generate base tests by using each at least once
  - Change 1 block at a time to an unselected one just as before

M base tests:

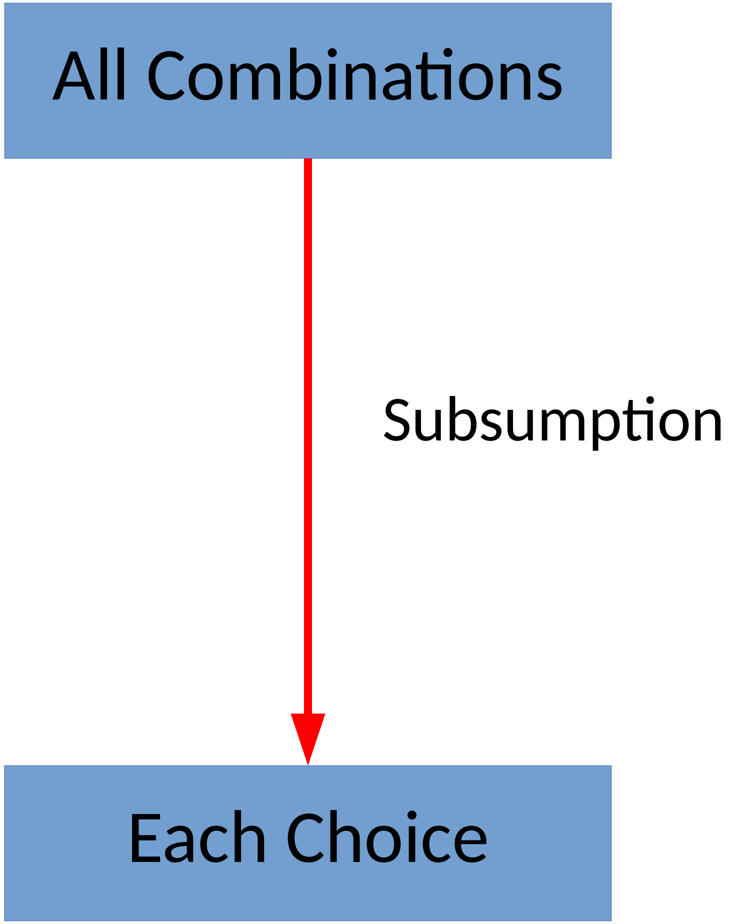
$$M * (1 + \sum |D_i - 1|)$$

How are they related?

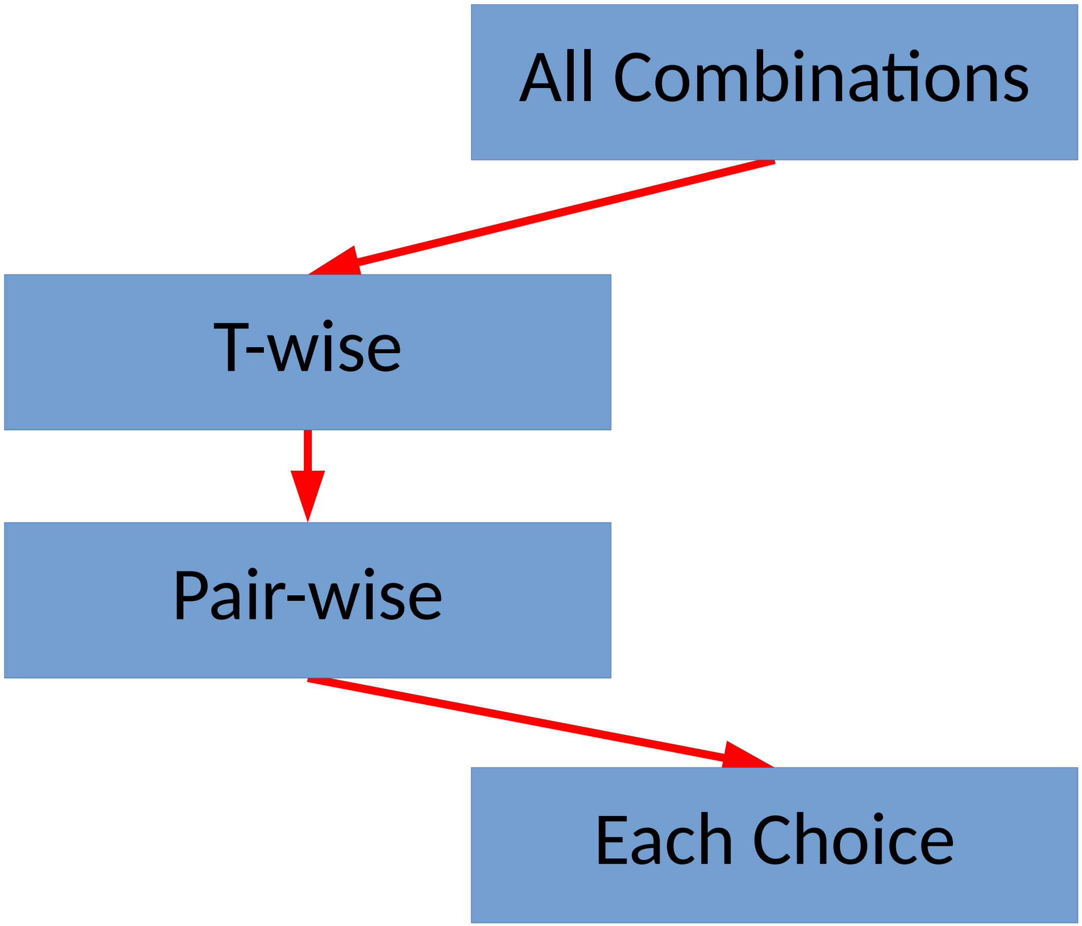
All Combinations

Each Choice

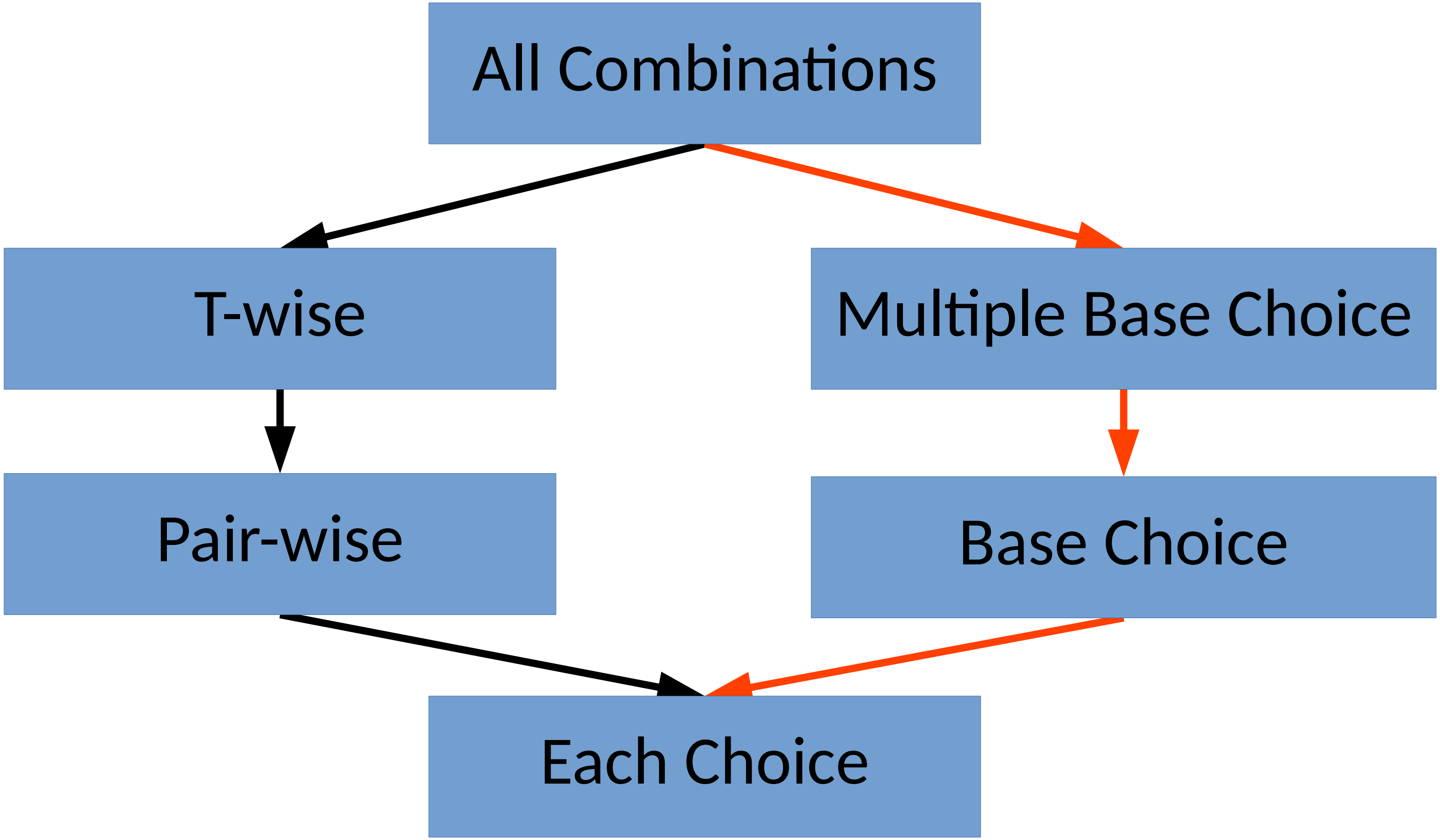
How are they related?



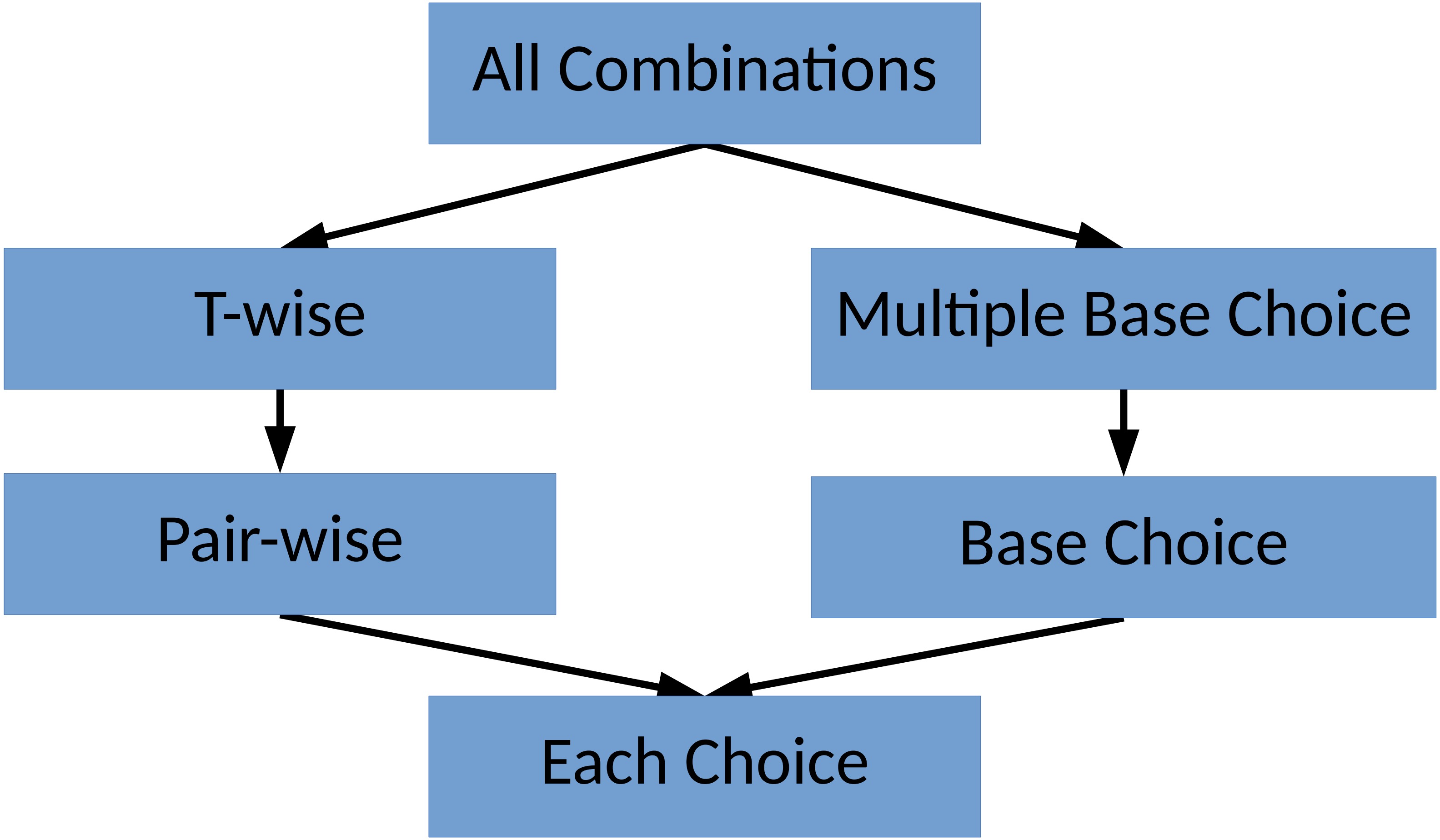
How are they related?



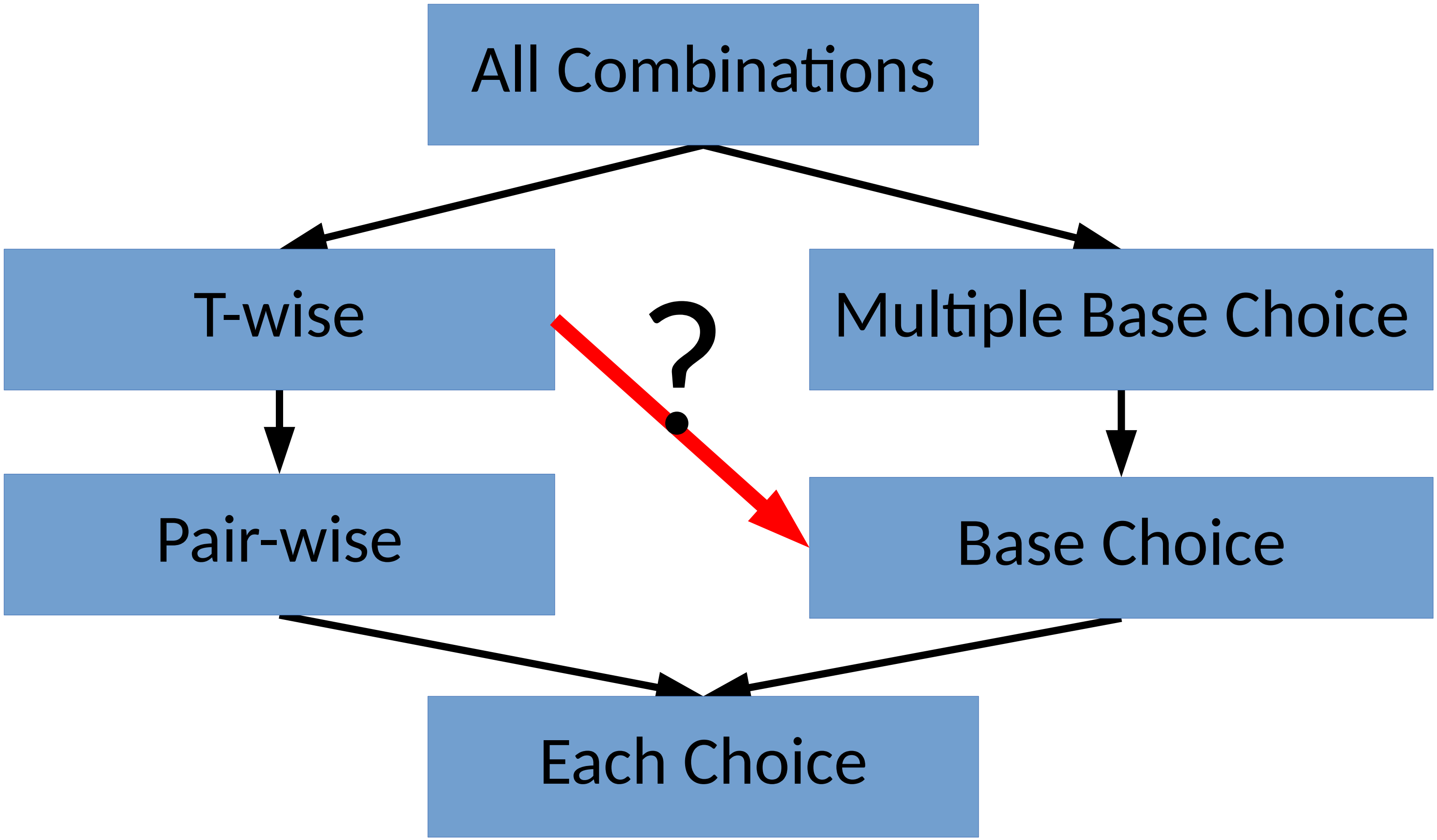
How are they related?



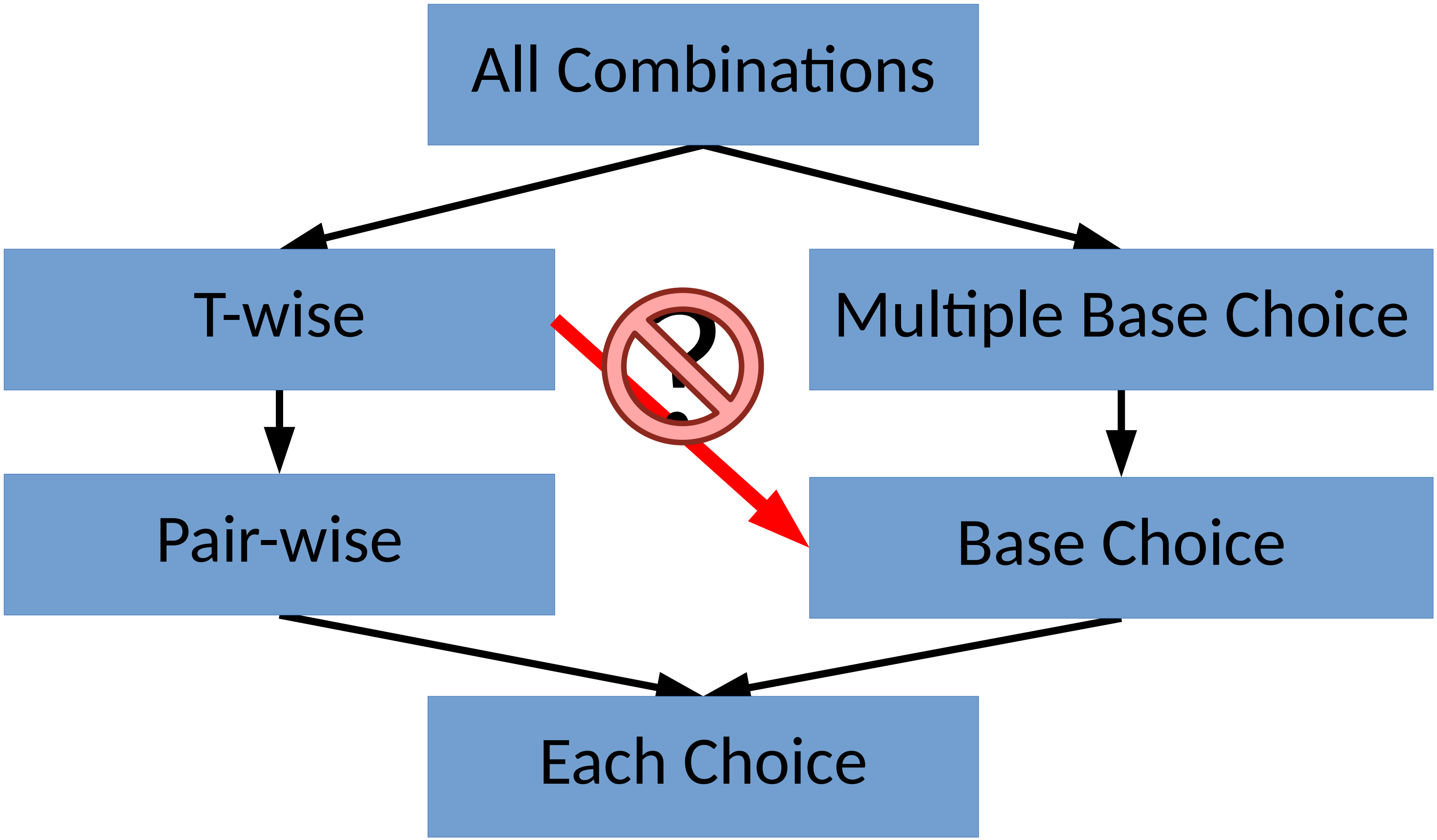
How are they related?



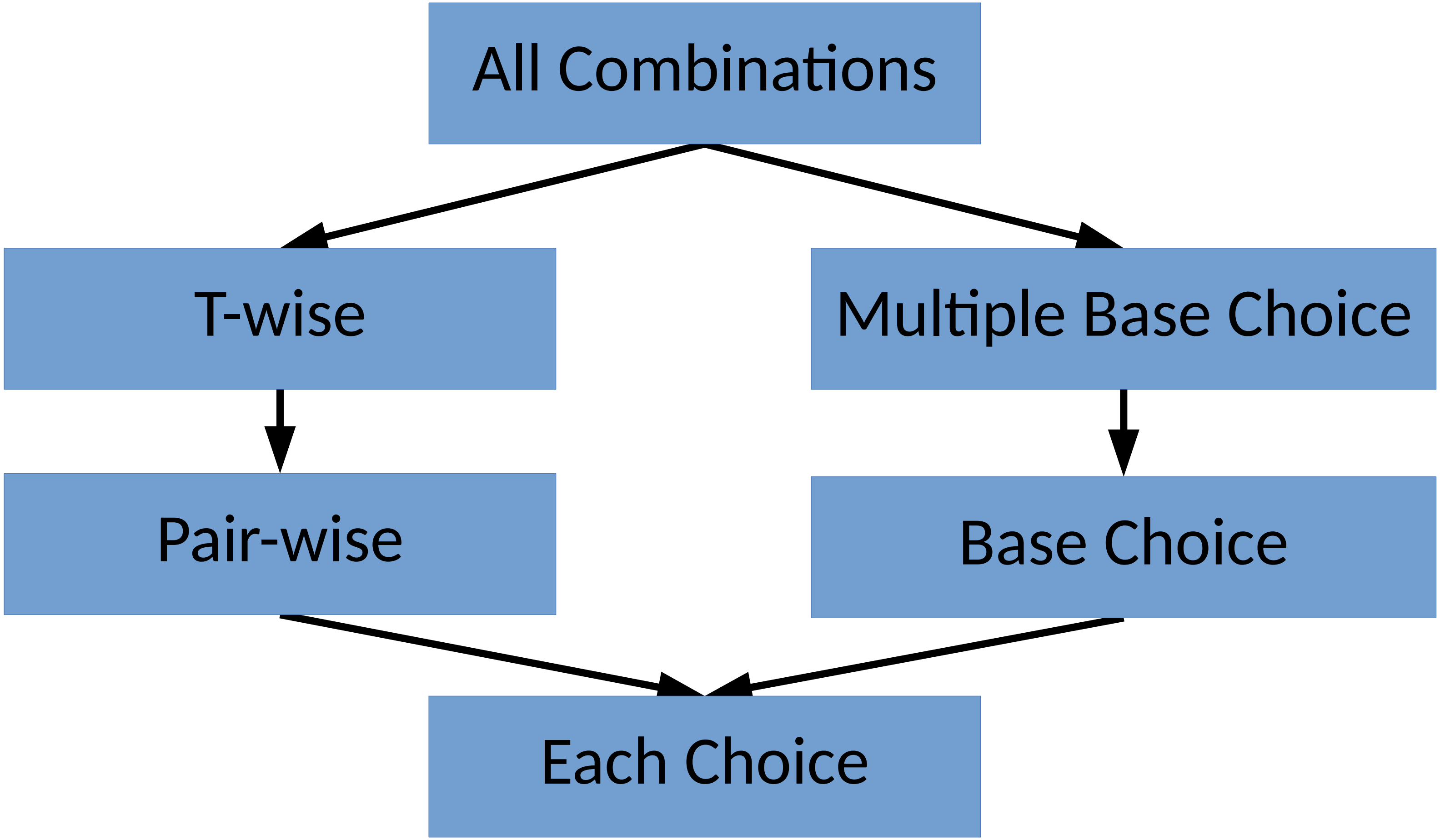
How are they related?



How are they related?



How are they related?



Remembering the constraints

- Constraints, and [error]s can reduce the # of tests further

Remembering the constraints

- Constraints, and [error]s can reduce the # of tests further
  - No need to test invalid constraints
  - No need to test more than one [error]

Concerns with pairwise testing

- We can reduce the number of tests. Great. What is the cost-benefit?

Concerns with pairwise testing

- We can reduce the number of tests. Great. What is the cost-benefit?
- Problems
  - Pairwise interactions are only truly tested when independent of others
  - The selected representative problem persists
  - Simple random testing seems to be as effective

Concerns with pairwise testing

- We can reduce the number of tests. Great. What is the cost-benefit?
- Problems
  - Pairwise interactions are only truly tested when independent of others
  - The selected representative problem persists
  - Simple random testing seems to be as effective
- Care must be taken, while there is tooling & some industry adoption, it cannot be adopted blindly

Summary

- Combinatorial testing strategies can reduce the cost of input space partitioning

Summary

- Combinatorial testing strategies can reduce the cost of input space partitioning
- Care must be taken to control the loss of testing power in the process