CMPT 473
Software Quality Assurance

Scale & Combinatorial Testing

Nick Sumner

Recall from Last Time

- Consider our triangle classifier
  - Takes 3 integers for sides 1, 2, & 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, & 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, & 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, & 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 (part 2)

- We can subdivide partitions to cover more behavior

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, …, I_n

- How does the number of tests change?
- $|D_1|$ * $|D_2|$ * $|D_3|$ * … * $|D_n|$ = 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!

What Is The Scale?

Suppose inputs or characteristics I_1, I_2, I_3, ..., 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 would this lead to fewer tests?

Choosing Combinations

Several possible strategies:

- All Combinations

Choosing Combinations

Several possible strategies:

- All Combinations
  - Every combination of every block is tried
  - Leaps headfirst into combinatorial explosion

Choosing Combinations

Several possible strategies:

- All Combinations
  - Every combination of every block is tried
  - Leaps headfirst into combinatorial explosion

But is it inherently bad?

Combinations – Each Choice

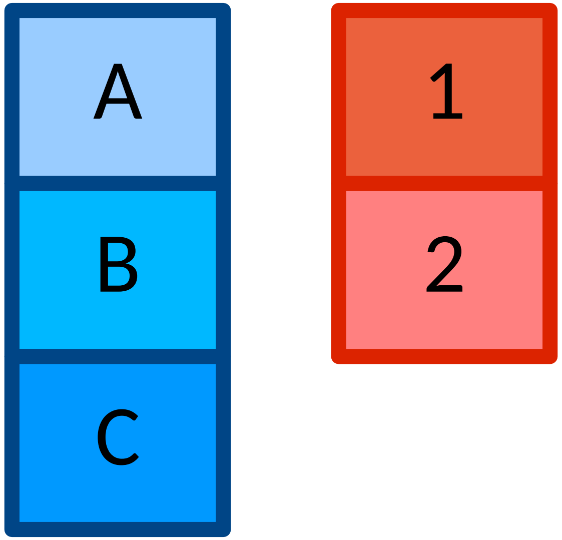
- How can we minimize #tests and still test each block?

Combinations – Each Choice

- How can we minimize #tests and still test each block?
- Each Choice
  - 1 value from each block used in at least one test

Combinations – Each Choice

- How can we minimize #tests and still test each block?
- Each Choice
  - 1 value from each block used in at least one test



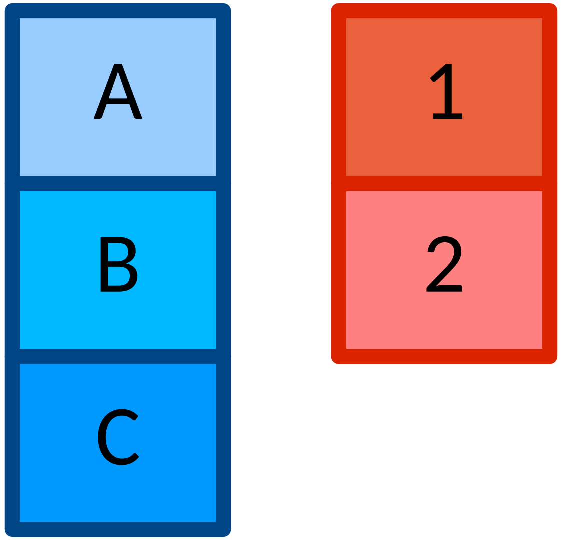
A
B
C

1
2

Adequate Tests:

Combinations – Each Choice

- How can we minimize #tests and still test each block?
- Each Choice
  - 1 value from each block used in at least one test



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

Combinations – Each Choice

- How can we minimize #tests and still test each block?
- Each Choice
  - 1 value from each block used in at least one test

What does this look like for the triangle classifier?

Combinations – Each Choice

- How can we minimize #tests and still test each block?
- Each Choice
  - 1 value from each block used in at least one test

What does this look like for the triangle classifier?

Are these tests *good*? Why?

Combinations – Each Choice

- How can we minimize #tests and still test each block?
- Each Choice
  - 1 value from each block used in at least one test

How many tests?

Combinations – Each Choice

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

How many tests?

Why?

Combinations – ???

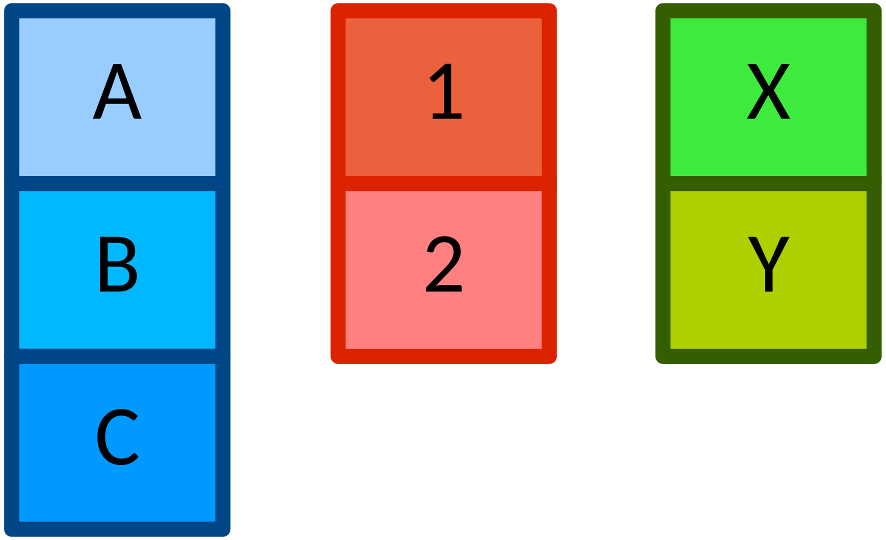
- Can we come up with a compromise?

Combinations – Pair Wise/All Pairs

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

Combinations – Pair Wise/All Pairs

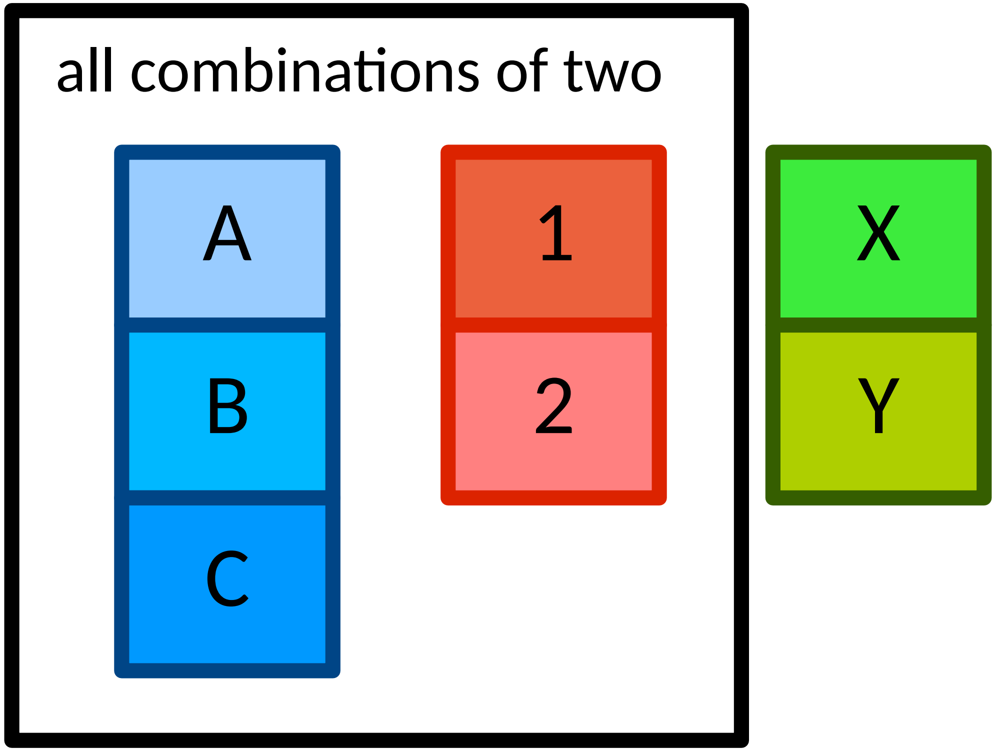
- Can we come up with a compromise?
- Pair Wise
  - 1 value for each block combined with 1 value for each other block


A
B
C
1
2
X
Y

Adequate Tests:

Combinations – Pair Wise/All Pairs

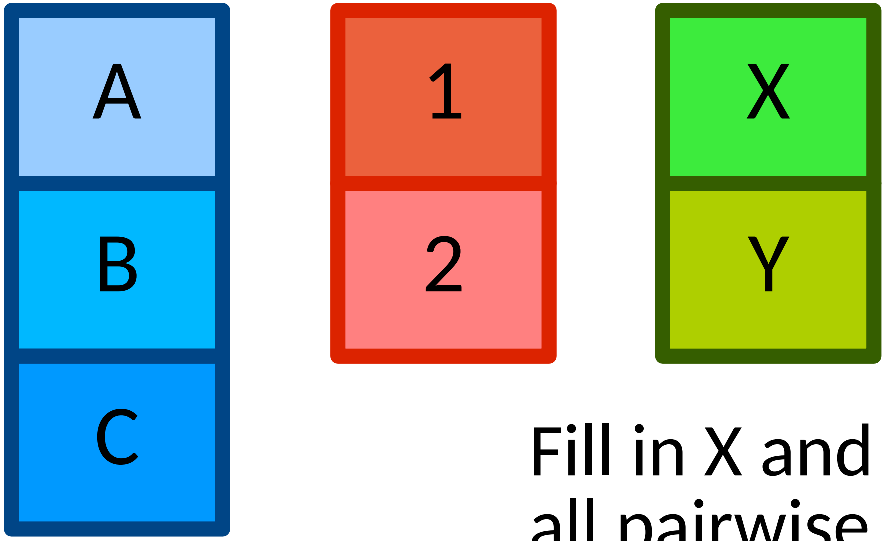
- Can we come up with a compromise?
- Pair Wise
  - 1 value for each block combined with 1 value for each other block



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

Combinations – Pair Wise/All Pairs

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

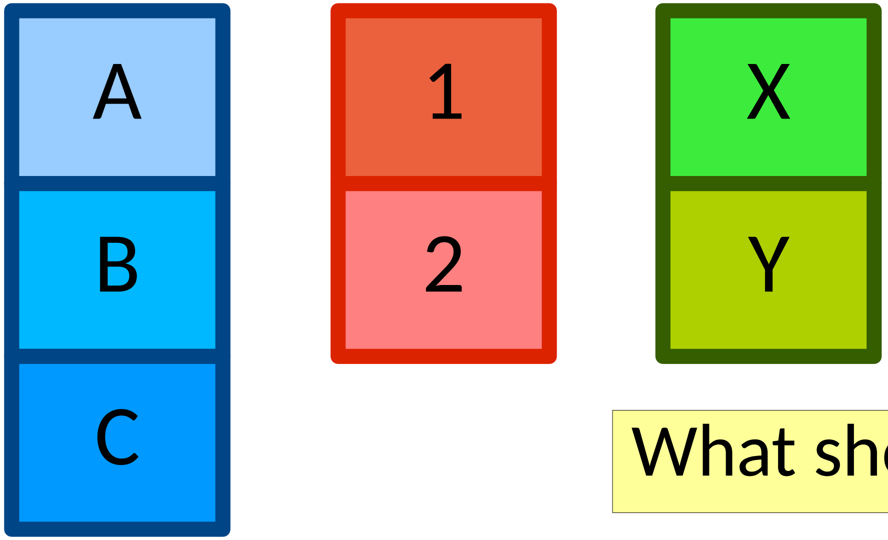


Adequate Tests:
(A,1,**X**), (A,2,**Y**)
(B,1,**Y**), (B,2,**X**)
(C,1,*), (C,2,*)

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

Combinations – Pair Wise/All Pairs

- Can we come up with a compromise?
- Pair Wise
  - 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 – Pair Wise/All Pairs

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

What does this look like for the triangle classifier?

Combinations – Pair Wise/All Pairs

- Can we come up with a compromise?
- Pair Wise
  - 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 – Pair Wise/All Pairs

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

How many tests?

Combinations – Pair Wise/All Pairs

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

How many tests?

Combinations – Pair Wise/All Pairs

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

How many tests?

Combinations – Pair Wise/All Pairs

- Can we come up with a compromise?
- Pair Wise
  - 1 value for each block combined with 1 value for each other block
  - #tests $\boxed{\geq}$ 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

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

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 (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 one block and taking other values from the base

Combinations – Base Choice

- So far, all of our approaches are domain agnostic

at i ow t tain v important?

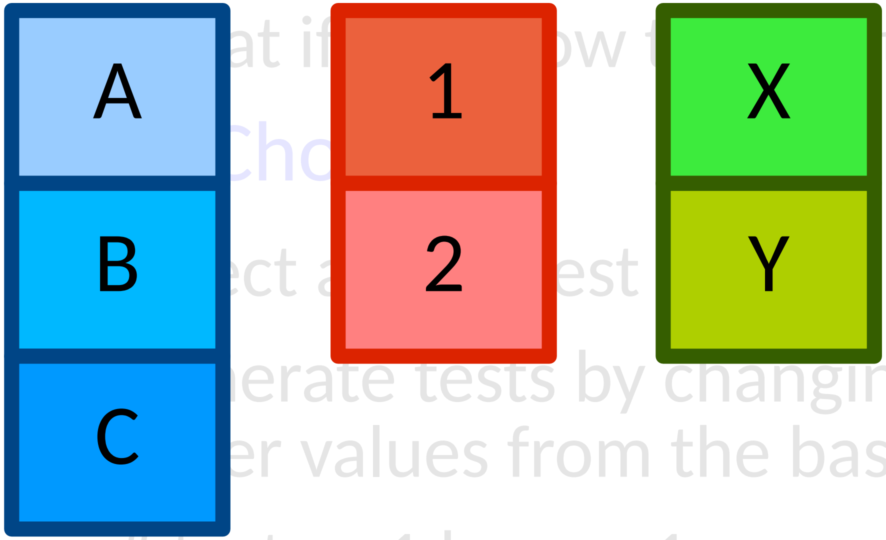
- Cho

ect a est

erate tests by changing only one block and taking

er values from the base

– # tests = 1 base + 1 per each other block

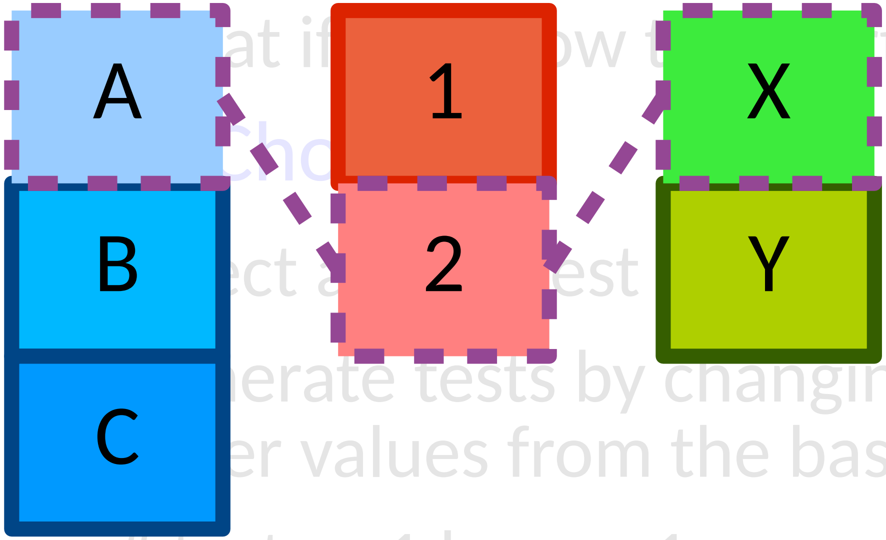


Base Test:

Adequate Tests:

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
  - at if ow t tain v important?
- Cho
  - ect a est
  - nerate tests by changing only one block and taking er values from the base
    - # tests = 1 base + 1 per each other block

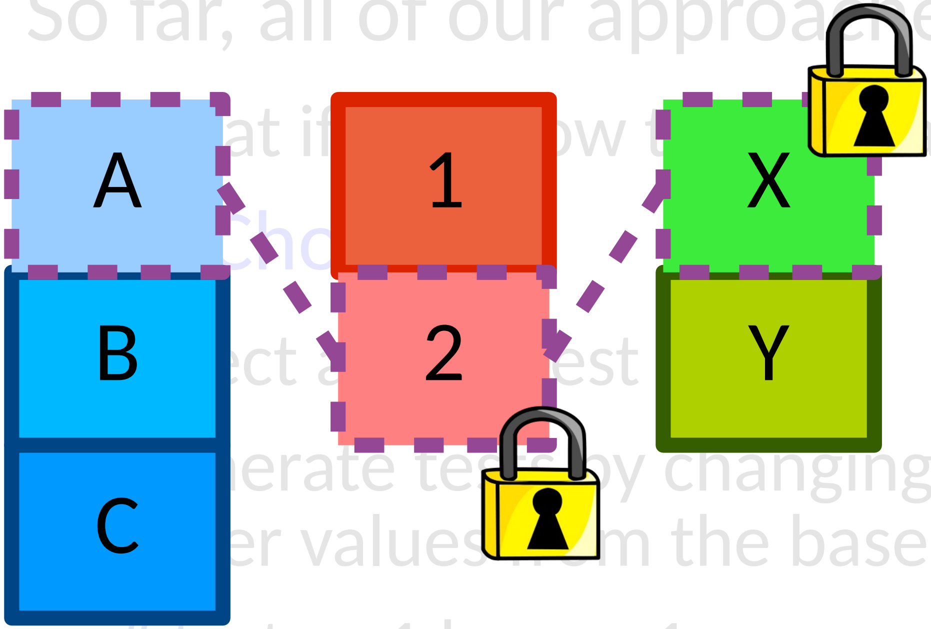


Base Test:
(A,2,X)

Adequate Tests:

Combinations – Base Choice

- So far, all of our approaches are domain agnostic
- at if ow t in v important?
- Cho
  - ect a est
  - erate te by changing only one block and taking er values m the base
    - # tests = 1 base + 1 per each other block

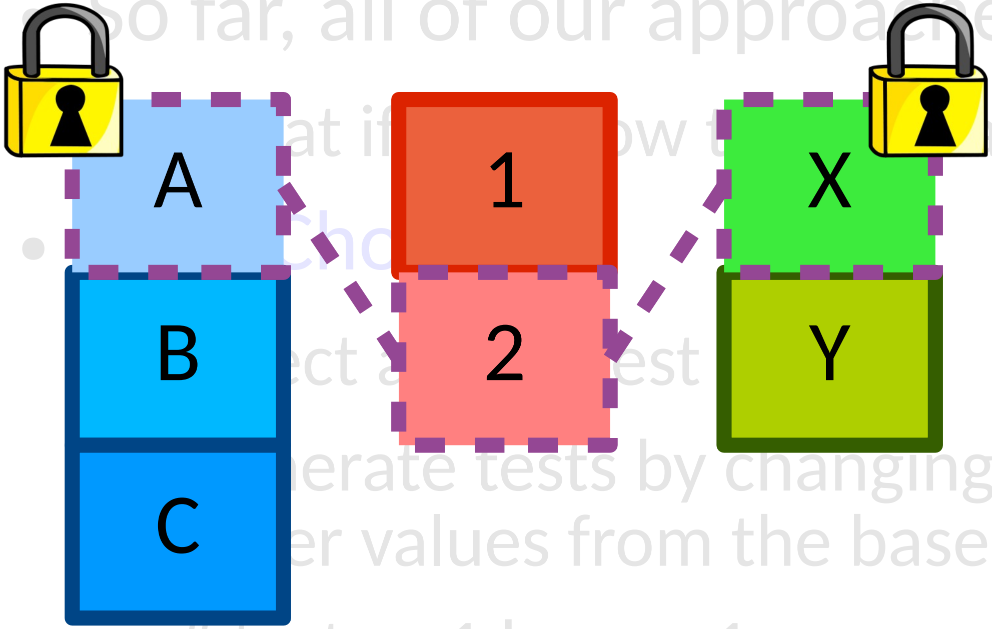


Base Test:
(A,2,X)

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

Combinations – Base Choice

- So far, all of our approa hes are domain agnostic
- at if ow t in v important?
- Cho
- ect a est
- erate tests by changing only one block and taking er values from the base
  - # tests = 1 base + 1 per each other block



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

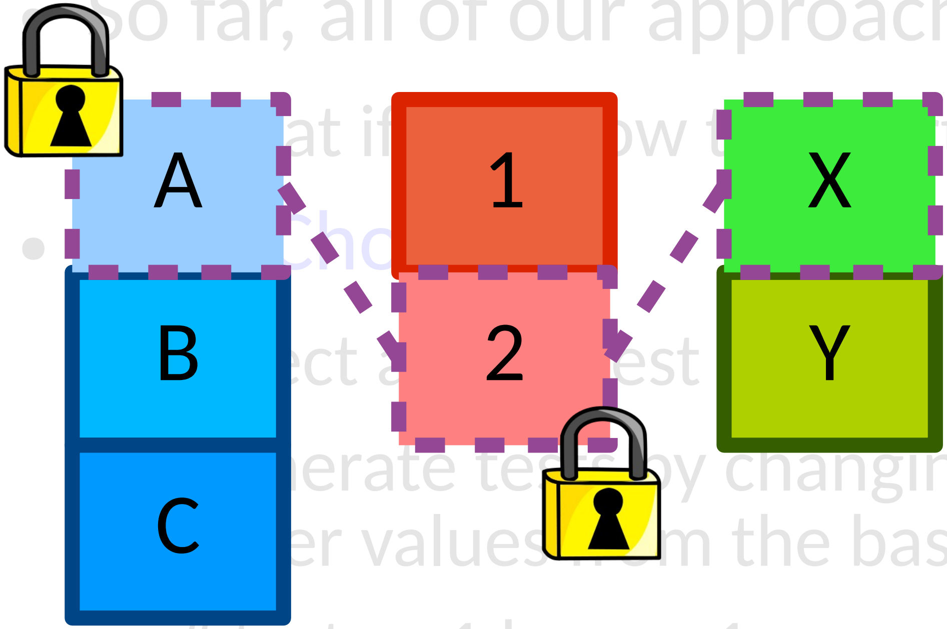
at if ow t tain v important?

- Cho

ect a est

erate te y changing only taking er values m the base

– # tests = 1 base + 1 per each oth ck

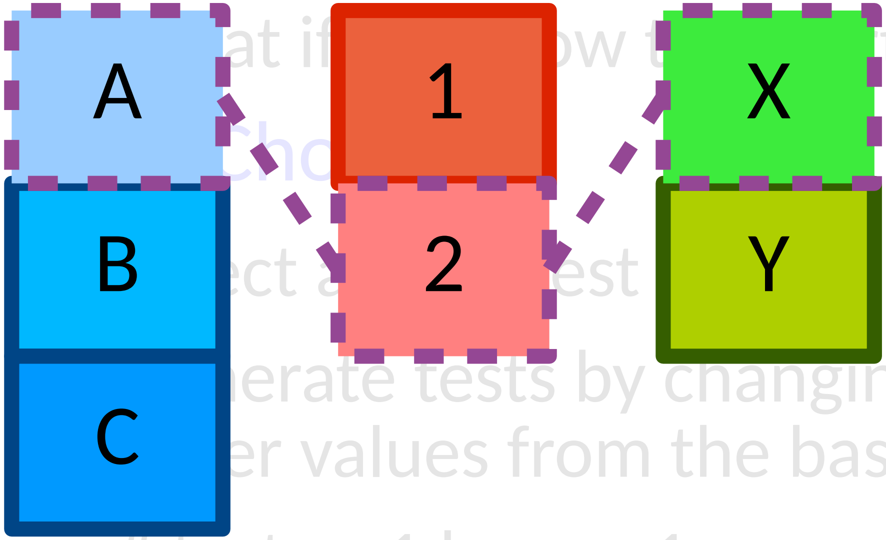


Base Test:
(A,2,X)

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
- at if ow t tain v important?
- Cho
  - ect a est
  - erate tests by changing only one block and taking er values from the base
    - # tests = 1 base + 1 per each other block



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 one block and taking other values from the base
  - # tests = 1 base + 1 per each other block

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 one block and taking other values from the base
  - # tests = 1 base + 1 per each other block

What does this look like for the triangle classifier?

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 one block and taking other values from the base
  - # tests = 1 base + 1 per each other block

What does this look like for the triangle classifier?

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 choices be extended?

Combinations – Multiple Base Choice

- *Notice the pattern.*
  - Can base choices 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:

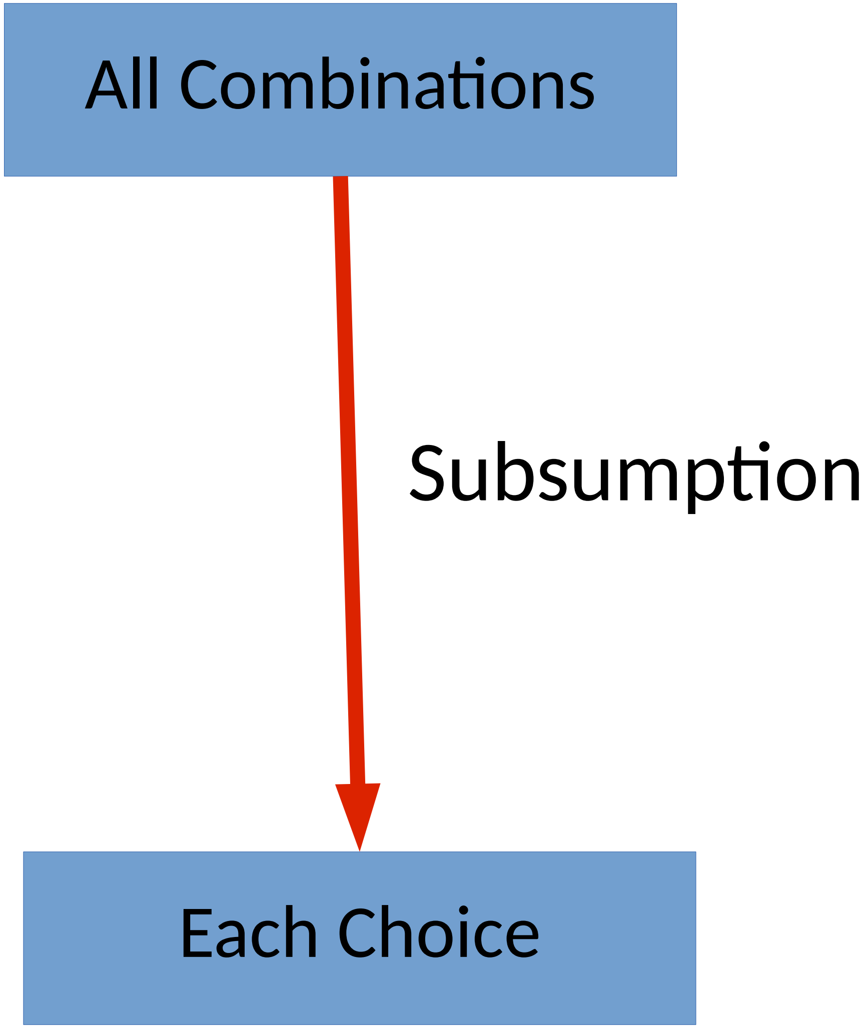
$$M * (1 + \sum |D_i\text{-}1|)$$

How Are They Related?

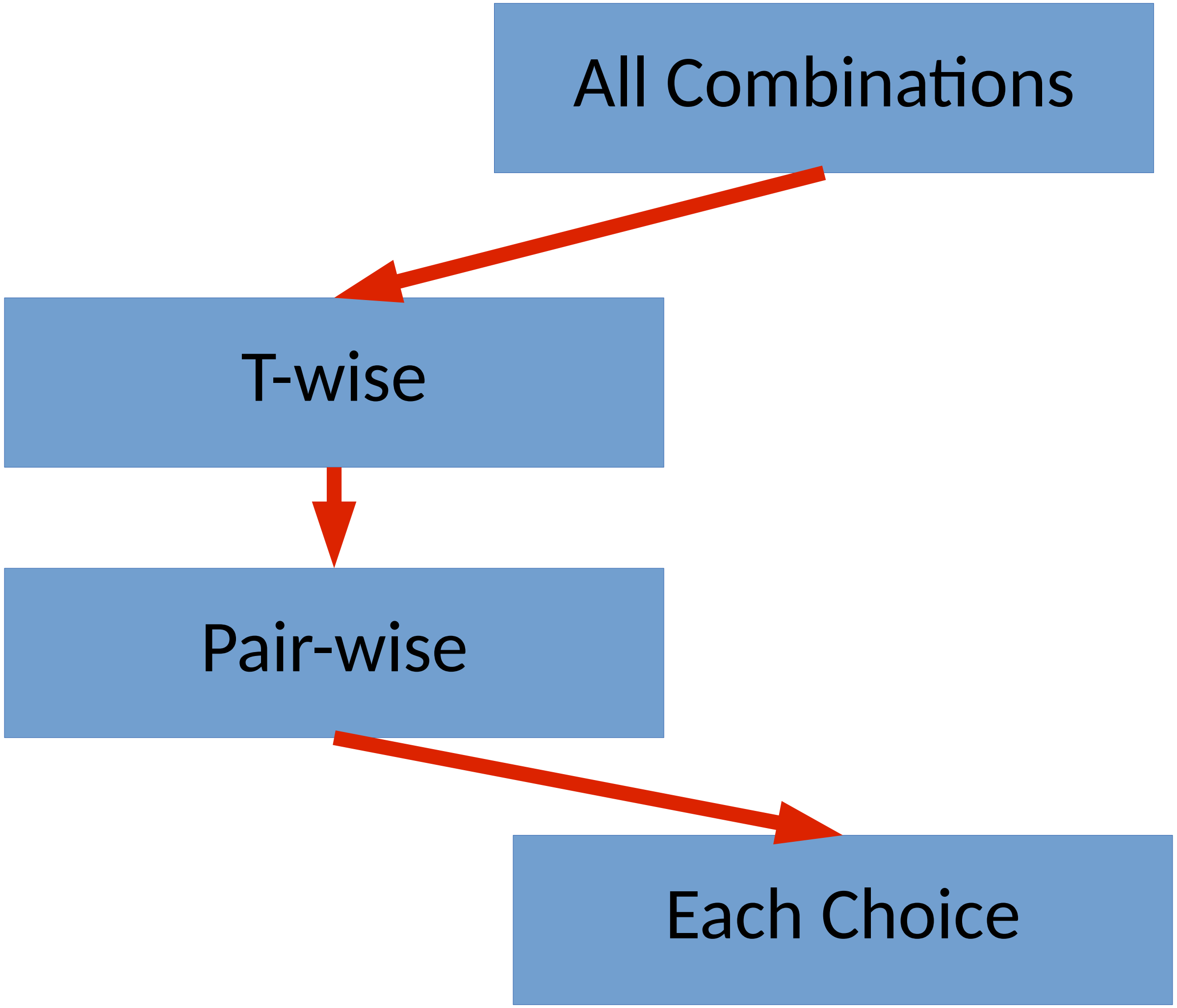
All Combinations

Each Choice

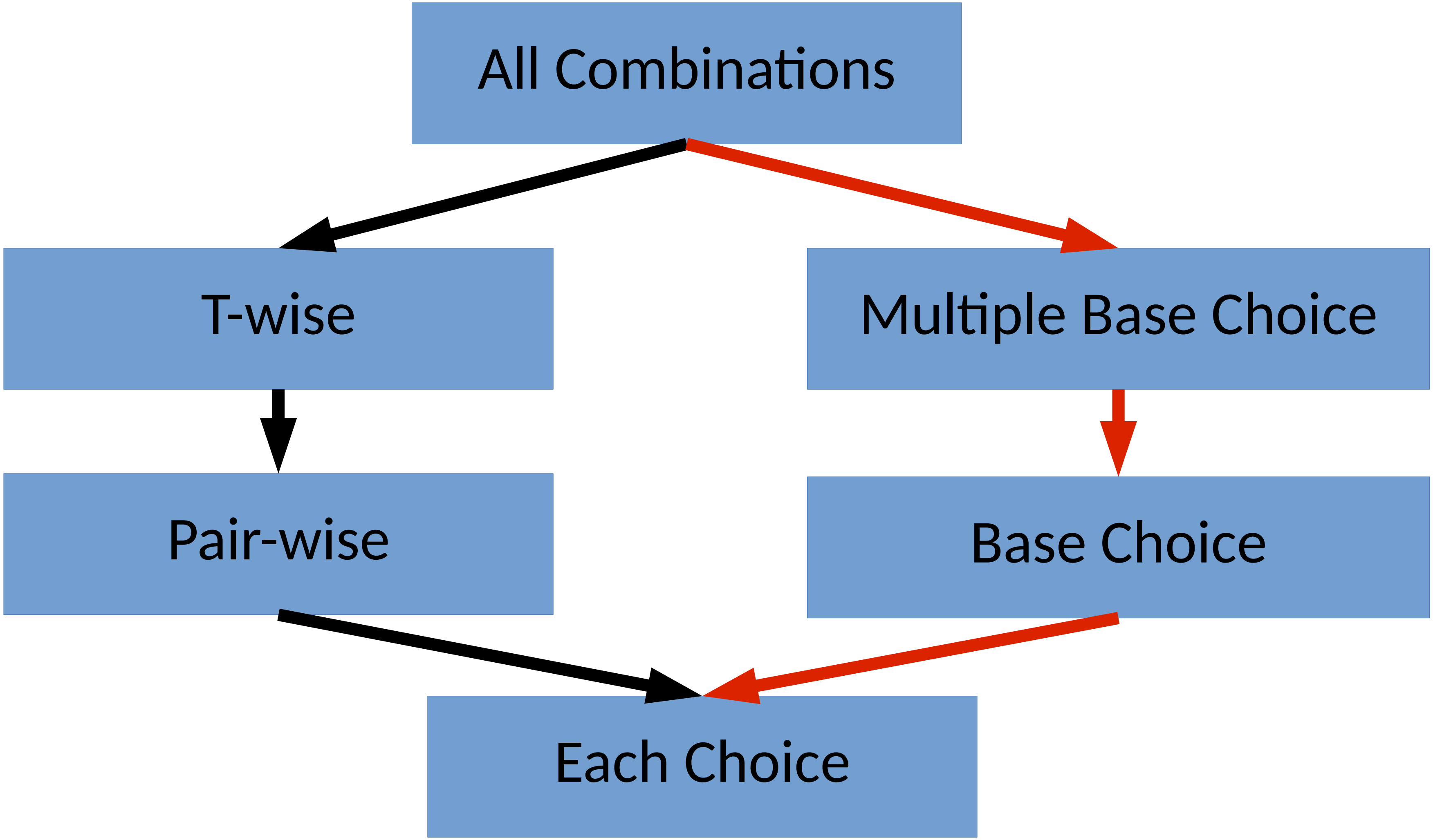
How Are They Related?



How Are They Related?



How Are They Related?



All Combinations

T-wise

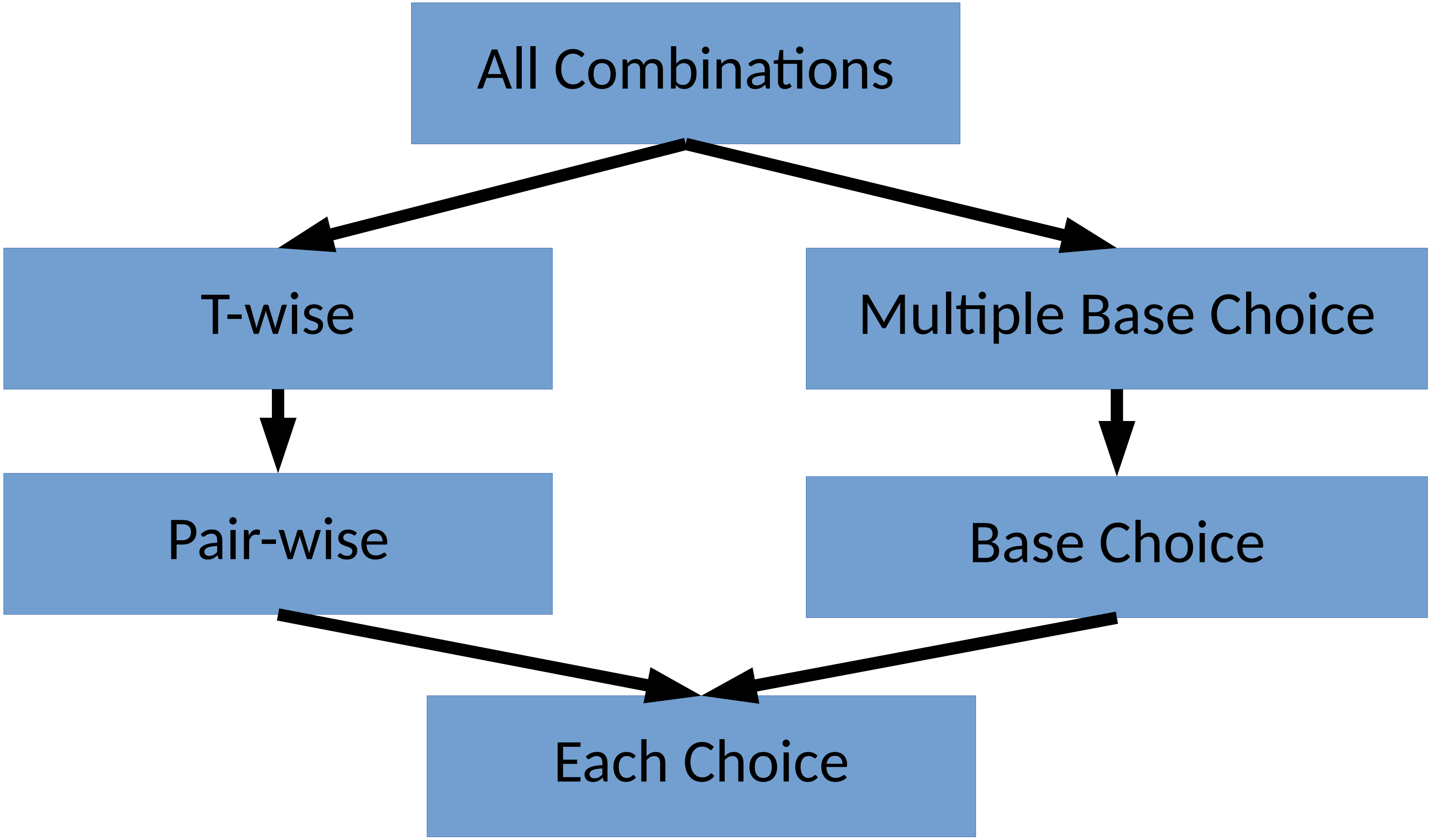
Multiple Base Choice

Pair-wise

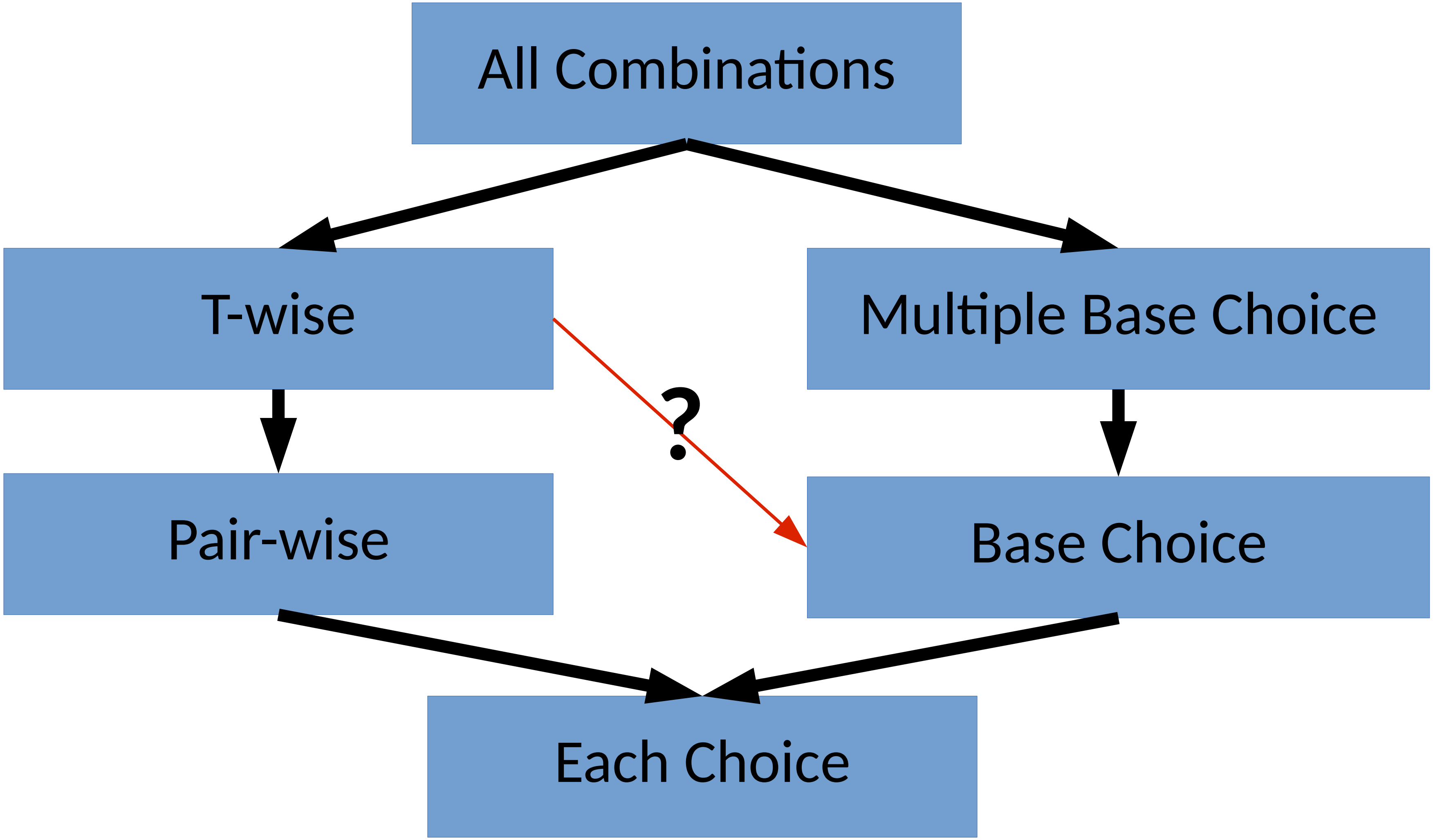
Base Choice

Each Choice

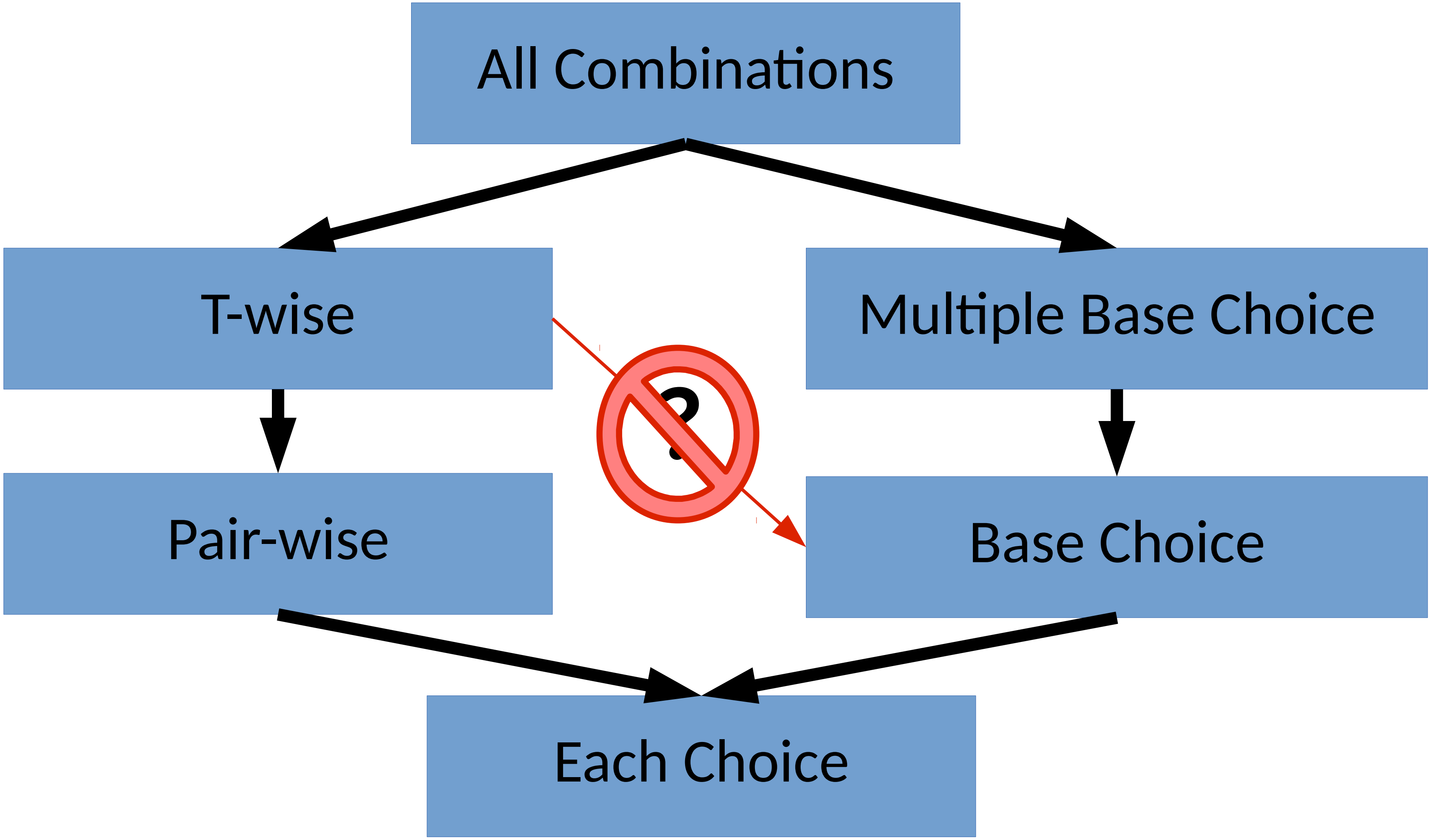
How Are They Related?



How Are They Related?



How Are They Related?



Remember 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]