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

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Side 1 &lt;0</td>
<td>Side 1 &gt;0</td>
<td>Side 1 =0</td>
<td>Side 1 &lt;0</td>
</tr>
<tr>
<td>Side 2 &lt;0</td>
<td>Side 2 &gt;0</td>
<td>Side 2 =0</td>
<td>Side 2 &lt;0</td>
</tr>
<tr>
<td>Side 3 &lt;0</td>
<td>Side 3 &gt;0</td>
<td>Side 3 =0</td>
<td>Side 3 &lt;0</td>
</tr>
</tbody>
</table>

3 guiding questions...
Recall from last time...

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

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Side 1 &lt; 0</td>
<td>Side 1 &gt; 0</td>
<td>Side 1 = 0</td>
<td>Side 1 &lt; 0</td>
</tr>
<tr>
<td>Side 2 &lt; 0</td>
<td>Side 2 &gt; 0</td>
<td>Side 2 = 0</td>
<td>Side 2 &lt; 0</td>
</tr>
<tr>
<td>Side 3 &lt; 0</td>
<td>Side 3 &gt; 0</td>
<td>Side 3 = 0</td>
<td>Side 3 &lt; 0</td>
</tr>
</tbody>
</table>

How many tests does this create?
Recall from last time...

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

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Side 1 &lt;0</td>
<td>Side 1 &gt; 0</td>
<td>Side 1 = 0</td>
<td>Side 1 &lt; 0</td>
</tr>
<tr>
<td>Side 2 &lt;0</td>
<td>Side 2 &gt; 0</td>
<td>Side 2 = 0</td>
<td>Side 2 &lt; 0</td>
</tr>
<tr>
<td>Side 3 &lt;0</td>
<td>Side 3 &gt; 0</td>
<td>Side 3 = 0</td>
<td>Side 3 &lt; 0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of side 1</td>
<td>Side 1 &gt; 1</td>
<td>Side 1 = 1</td>
<td>Side 1 = 0</td>
<td>Side 1 &lt; 0</td>
</tr>
<tr>
<td>Value of side 2</td>
<td>Side 2 &gt; 1</td>
<td>Side 2 = 1</td>
<td>Side 2 = 0</td>
<td>Side 2 &lt; 0</td>
</tr>
<tr>
<td>Value of side 3</td>
<td>Side 3 &gt; 1</td>
<td>Side 3 = 1</td>
<td>Side 3 = 0</td>
<td>Side 3 &lt; 0</td>
</tr>
</tbody>
</table>

How many tests now?
What is the scale?

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

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

What does it mean in practice?
- Find command: $4 \times 3 \times 3 \times 3 \times 3 \times 3 \times 2 = 1944$ tests
What is the scale?

Suppose inputs or characteristics $I_1, I_2, I_3, ..., I_n$

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

What does it mean in practice?

- Find command: $4 \times 3 \times 3 \times 3 \times 3 \times 3 \times 2 = 1944$ tests
- Website generator: $> 30 \rightarrow > 1$ 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?
- $|D_1| \times |D_2| \times |D_3| \times \ldots \times |D_n| = k^n$
- This is *combinatorial explosion*

What does it mean in practice?

- Find command: $4 \times 3 \times 3 \times 3 \times 3 \times 3 \times 2 = 1944$ tests
- Website generator: $> 30 \rightarrow > 1$ 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

<table>
<thead>
<tr>
<th>Pattern Size:</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>[Property Empty]</td>
</tr>
<tr>
<td>Single character</td>
<td>[Property NonEmpty]</td>
</tr>
<tr>
<td>Many characters</td>
<td>[Property NonEmpty]</td>
</tr>
<tr>
<td>Longer than any line in the file</td>
<td>[Property NonEmpty]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quoting:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern is quoted</td>
<td>[Property Quoted]</td>
</tr>
<tr>
<td>Pattern is not quoted</td>
<td>[If NonEmpty]</td>
</tr>
<tr>
<td>Pattern is improperly quoted</td>
<td>[If NonEmpty]</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Pattern Size:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>[Property Empty]</td>
</tr>
<tr>
<td>Single character</td>
<td>[Property NonEmpty]</td>
</tr>
<tr>
<td>Many characters</td>
<td>[Property NonEmpty]</td>
</tr>
<tr>
<td>Longer than any line in the file</td>
<td>[Property NonEmpty]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quoting:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern is quoted</td>
<td>[Property Quoted]</td>
</tr>
<tr>
<td>Pattern is not quoted</td>
<td>[If NonEmpty]</td>
</tr>
<tr>
<td>Pattern is improperly quoted</td>
<td>[If NonEmpty]</td>
</tr>
</tbody>
</table>
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
  - \([\text{property}]\) to identify rules for useful tests
  - \([\text{error}]\) 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

A B C 1 2

X Y

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?
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 \geq \text{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 \geq \text{product of 2 largest domain partitionings}

How many tests?

Expected on the order of \(|D_1| \times |D_2| \times \log(n)\)
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 \geq \text{product of } T \text{ 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
  - \( \# \text{tests} \geq \text{product of T largest domain partitionings} \)
  - 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
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

**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
  - Select a base test
  - Generate tests by changing only 1 block and taking other values from the base

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:
- Select a base test.
- Generate tests by changing only 1 block and taking other values from the base.

**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 Choice**
  - Select a base test
  - Generate tests by changing only 1 block and taking other values from the base

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

**Base Test**: (A,2,X)
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.

**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

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?

\[ 1 + \sum \mid D_i - 1 \mid \]
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

\[
M \text{ base tests:} \\
M \times \left(1 + \sum |D_i - 1| \right)
\]
How are they related?

All Combinations

Each Choice
How are they related?

All Combinations

Subsumption

Each Choice
How are they related?

- All Combinations
  - T-wise
    - Pair-wise
      - Each Choice
How are they related?

- All Combinations
- T-wise
- Pair-wise
- Multiple Base Choice
- Base Choice
- Each Choice
How are they related?

All Combinations

T-wise

Pair-wise

Multiple Base Choice

Base Choice

Each Choice
How are they related?

All Combinations

T-wise

Pair-wise

Multiple Base Choice

Base Choice

Each Choice
How are they related?

- All Combinations
- T-wise
- Pair-wise
- Multiple Base Choice
- Base Choice
- Each Choice
How are they related?

All Combinations

T-wise

Pair-wise

Multiple Base Choice

Base Choice

Each Choice
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