More ISP

Isaac Griffith

CS 4422 and CS 5522
Department of Computer Science
Idaho State University

Outcomes

At the end of Today's Lecture you will be able to:

- Understand the basics of ISP
- To partition an input domain
- Model an input domain
- Understand parameters and their characteristics
- Understand multiple ISP based criteria

Inspiration

"A bug in the hand is worth two in the box." – Anonymous

triang() IDM based on Syntax

- `triang()` has one testable function and three integer inputs

First Characterization of TriType's Inputs

Characteristic	b_1	b_2	b_3
q_1 = "Relation of Side 1 to 0"	> 0	= 0	< 0
q_2 = "Relation of Side 2 to 0"	> 0	= 0	< 0
q_3 = "Relation of Side 3 to 0"	> 0	= 0	< 0

- A maximum of 3 * 3 * 3 = **27** tests
- Some triangles are **valid**, some are **invalid**
- **Refining** the characterization can lead to more tests...

Refining triang()'s IDM

Second Characterization of TriType's Inputs

Characteristic	b_1	b_2	b_3	b_4
q_1 = "Relation of Side 1 to 0"	> 1	$= 1$	$= 0$	< 0
q_2 = "Relation of Side 2 to 0"	> 1	$= 1$	$= 0$	< 0
q_3 = "Relation of Side 3 to 0"	> 1	$= 1$	$= 0$	< 0

- A maximum of 4 * 4 * 4 = **64** tests
- **Complete** because the inputs are integers (0..1)

Refining triang()'s IDM

Possible values for partition q_1

Characteristic	b_1	b_2	b_3	b_4
side1	5	1	0	-5
boundaries	2	1	0	-1

ROAR

triang() IDM Based on Behavior

- First two characterizations are based on **syntax**-parameters and their type
- A **semantic** level characterization could use the fact that the three integers represent a triangle

Geometric Characterization of triang()'s Inputs

Characteristic	b_1	b_2	b_3	b_4
q_1 = "Geometric Classification"	scalene	isosceles	equilateral	invalid

- **Problem**: Equilateral is also isosceles!

triang() IDM Based on Behavior

- We need to **refine** the example to make characteristics valid

Correct Characterization of triang()'s Inputs

Characteristic	b_1	b_2	b_3	b_4
q_1 = "Geometric Classification"	scalene	isosceles not equilateral	equilateral	invalid

Choosing Values for triang()

- **Values** for this partitioning can be chosen as follows

Possible values for geometric partition

Characteristic	b_1	b_2	b_3	b_4
Triangle	(4, 5, 6)	(3, 3, 4)	(3, 3, 3)	(3, 4, 8)

Yet Another triang() IDM

- A **different approach** would be to break the geometric characterization into four separate characteristics

Four Characteristics for triang()

Characteristic	b_1	b_2
q_1 = "Scalene"	True	False
q_2 = "Isosceles"	True	False
q_3 = "Equilateral"	True	False
q_4 = "Valid"	True	False

- Use **constraints** to ensure that
  - **Equilateral = True** implies **Isosceles = True**
  - **Valid = False** implies **Scalene = Isosceles = Equilateral = False**

In-Class Exercise

Group Exercise

- Work with 2 or 3 classmates
- Which two properties must be satisfied for an input domain to be properly partitioned?

Step 4

Choosing Combinations of Values

- Once characteristics and partitions are defined, the next step is to **choose test values**
- We use **criteria** to choose **effective** subsets
- The most obvious criterion is to choose all combinations

All Combinations (ACoC)

All combinations of blocks form all characteristics must be used.

- Number of tests is the product of the number of blocks in each characteristic: $\prod_{i=1}^{Q} B_i$
- The second characterization of `triang()` results in 4*4*4 = **64 tests**
  - Too many?

ROAR

ISP Criteria - All Combinations

- Consider the "second characterization" of `triang()` as given before:

Characteristic	b_1	b_2	b_3	b_4
q_1 = "Relation of Side 1 to 0"	> 1	$= 1$	$= 0$	< 0
q_2 = "Relation of Side 2 to 0"	> 1	$= 1$	$= 0$	< 0
q_3 = "Relation of Side 3 to 0"	> 1	$= 1$	$= 0$	< 0

- For convenience, we relabel the blocks using abstractions

Characteristic	b_1	b_2	b_3	b_4
A	A1	A2	A3	A4
B	B1	B2	B3	B4
C	C1	C2	C3	C4

ISP Criteria - ACoC Tests

```
A1 B1 C1    A2 B1 C1    A3 B1 C1    A4 B1 C1
A1 B1 C2    A2 B1 C2    A3 B1 C2    A4 B1 C2
A1 B1 C3    A2 B1 C3    A3 B1 C3    A4 B1 C3
A1 B1 C4    A2 B1 C4    A3 B1 C4    A4 B1 C4

A1 B2 C1    A2 B2 C1    A3 B2 C1    A4 B2 C1
A1 B2 C2    A2 B2 C2    A3 B2 C2    A4 B2 C2
A1 B2 C3    A2 B2 C3    A3 B2 C3    A4 B2 C3
A1 B2 C4    A2 B2 C4    A3 B2 C4    A4 B2 C4

A1 B3 C1    A2 B3 C1    A3 B3 C1    A4 B3 C1
A1 B3 C2    A2 B3 C2    A3 B3 C2    A4 B3 C2
A1 B3 C3    A2 B3 C3    A3 B3 C3    A4 B3 C3
A1 B3 C4    A2 B3 C4    A3 B3 C4    A4 B3 C4

A1 B4 C1    A2 B4 C1    A3 B4 C1    A4 B4 C1
A1 B4 C2    A2 B4 C2    A3 B4 C2    A4 B4 C2
A1 B4 C3    A2 B4 C3    A3 B4 C3    A4 B4 C3
A1 B4 C4    A2 B4 C4    A3 B4 C4    A4 B4 C4
```

- ACOC yields 4 * 4 * 4 = **64 tests** for `triang()`
  - This is almost certainly more than we need.
- Only **8 are valid** (all sides greater than zero)
  - `A1 B1 C1    A2 B1 C1    A1 B1 C2    A2 B1 C2`
  - `A1 B2 C1    A2 B2 C1    A1 B2 C2    A2 B2 C2`

ROAR

ISP Criteria - Each Choice

- 64 tests for `triang()` is almost certainly way too many
- One criterion comes from the idea that we should try at **least one** value form each block

Each Choice Coverage (ECC)

One value from each block for each characteristic must be used in at least one test case

- Number of tests is the number of blocks in the **largest** characteristic: $\max_{i=1}^{Q} B_i$

For `triang()`

```
A1, B1, C1
A2, B2, C2
A2, B2, C3
A4, B4, C4
```

Substituting Values

```
2, 2, 2
1, 1, 1
0, 0, 0
-1, -1, -1
```

ISP Criteria - Base Choice

- Testers sometimes recognize that certain values are **important**
- This uses **domain knowledge** of the program

Base Choice Coverage (BCC)

A base choice block is chosen for each characteristic, and a base test is formed by using the base choice for each characteristic. Subsequent tests are chosen by holding all but one base choice constant and using each non-base choice in each other characteristic.

ISP Criteria - Base Choice

- Number of tests is one base test + one test for each other block: $1 + \sum_{i=1}^{Q} (B_i - 1)$

For `triang()`

Base: `A1, B1, C1`

`A1, B1, C2`	`A1, B2, C1`	`A2, B1, C1`
`A1, B1, C3`	`A1, B3, C1`	`A3, B1, C1`
`A1, B1, C4`	`A1, B4, C1`	`A4, B1, C1`

Base Choice Notes

- The base test must be **feasible**
  - That is, all base choices must be **compatible**
- **Base choices** can be
  - Most likely from an end-use point of view
  - Simplest
  - Smallest
  - First in some ordering
- **Happy path** tests often make good base choices
- The base choice is a **crucial design** decision
  - Test designers should **document** why the choices were made

ISP Criteria - Multiple Base Choice

- We sometimes have **more than one logical base choice**

Multiple Base Choice Coverage (MBCC)

At least one and possibly more, base choice blocks are chosen for each characteristic and base tests are formed by using each base choice for each characteristic at least once. Subsequent tests are chosen by holding all but one base choice constant for each base test and using each non-base choice in each other characteristic.

ISP Criteria - Multiple Base Choice

- If M base tests and m_i base choices for each characteristic: $M + \sum_{i=1}^{Q} (M * (B_i - m_i))$

For `triang()`

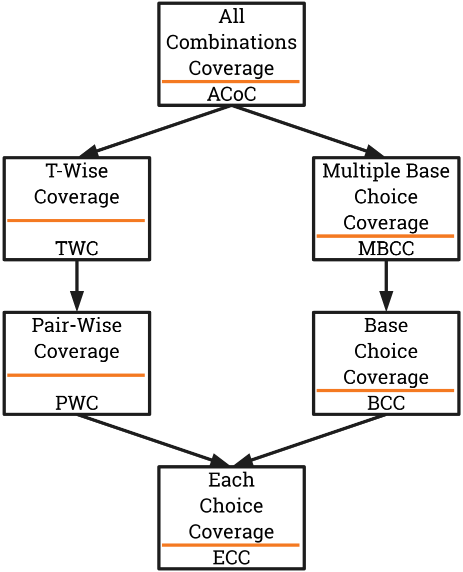
Base: `A1, B1, C1`

```
A1, B1, C3    A1, B3, C1    A3, B1, C1
A1, B1, C4    A1, B4, C1    A4, B1, C1
```

Base: `A2, B2, C2`

```
A2, B2, C3    A2, B3, C2    A3, B2, C2
A2, B2, C4    A2, B4, C2    A4, B2, C2
```

ISP Coverage Criteria Subsumption



Summary

- Fairly easy to apply, even with **no automation**
- Convenient ways to **add more or less testing**
- Applicable to **all levels** of testing - unit, class, integration, system, etc.
- Based only on the **input space** of the program, not the implementation

Simple, straightforward, effective, and widely used

Are there any questions?