Com S 417
Software Testing

Fall 2017 – Week 3, Lecture 6

Announcements

- Lab 2 is posted.
  - Due before class Sept. 19
- Exam 1
  - In class (1 hour long) Thursday, Sept. 21
  - Material covered in weeks 1-4.

Quiz

1. SUT
2. basic block
3. black box
4. integration testing
5. equivalence class
6. stress testing
7. load testing
8. @Before
9. coverage
10. static tests

a. A static test tool.
b. Demonstrates that the SUT can a handle a specific level of load.
c. Runs once per class.
d. Evaluate software without executing it.
e. Runs once per test.
f. A measure of test set quality.
g. Software under test.
h. Attempts to load the SUT until it breaks.
i. (I) Measures the long-term probability of SUT failure.
j. Noise injected in a test.
k. A code segment that executes sequentially
l. (EL) Tests the interoperation of modules
m. Tests designed from requirements only.
n. A subset of an input domain where all members are expected to trigger the same behavior.

Quiz: important distinctions

Black Box:

- Black box tests are dynamic tests!
  - Once designed, selected, and implemented, black box tests must be used to execute the SUT,
- They differ (from white box – also dynamic) only because black box tests don’t require implementation information (source and program design documents) to design.

Static Tests:

- Evaluate the code without running it.
  - compilers, style checkers, code reviews, etc.

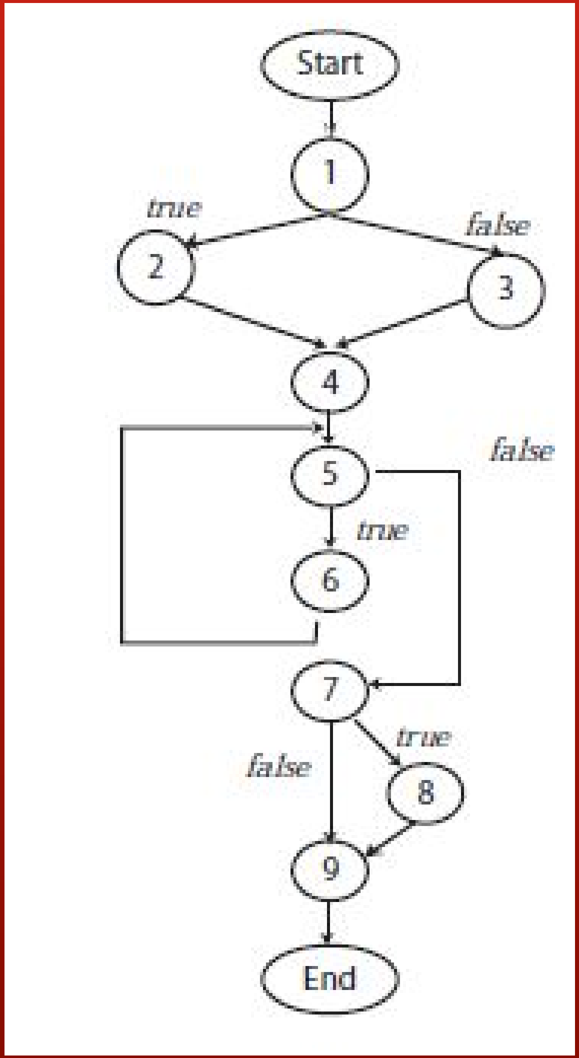
@Before vs @BeforeClass

- The more commonly used @Before executes the associated setup routine *before each test.*

Infeasible test cases

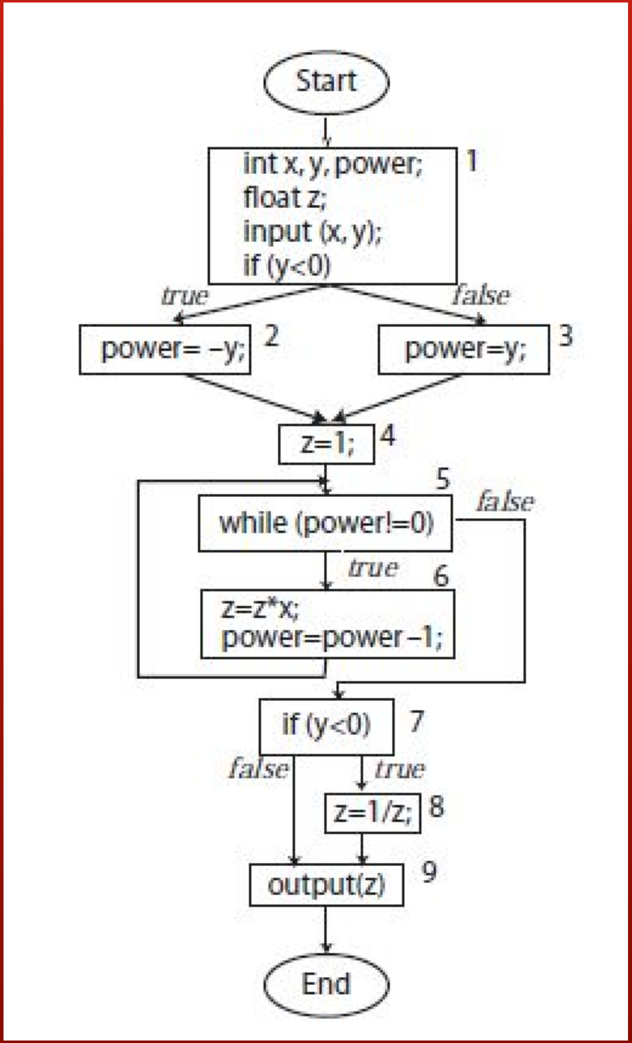
Why does Mathur say these paths are infeasible?

- p5 = (Start, 1, 3, 4, 5, 6, 5, 7, 8, 9, End)
- p6 = (Start, 1, 2, 4, 5, 7, 9, End)



Figures from: Mathur

Unfeasible test cases (con't)



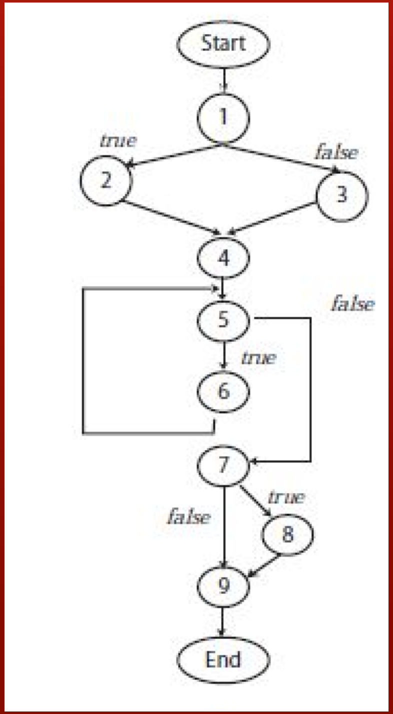
Figures from: Mathur

The answer lies in the original code.

- Track the value of the variables required to follow the path

  p6 = (Start, 1, 2, 4, 5, 7, 9, End)

- Power and y are the critical vars. Why?



Multi-D Partitioning.

Multi-dimensional input partitioning treats all input n-tuples as points in an n-dimensional space.

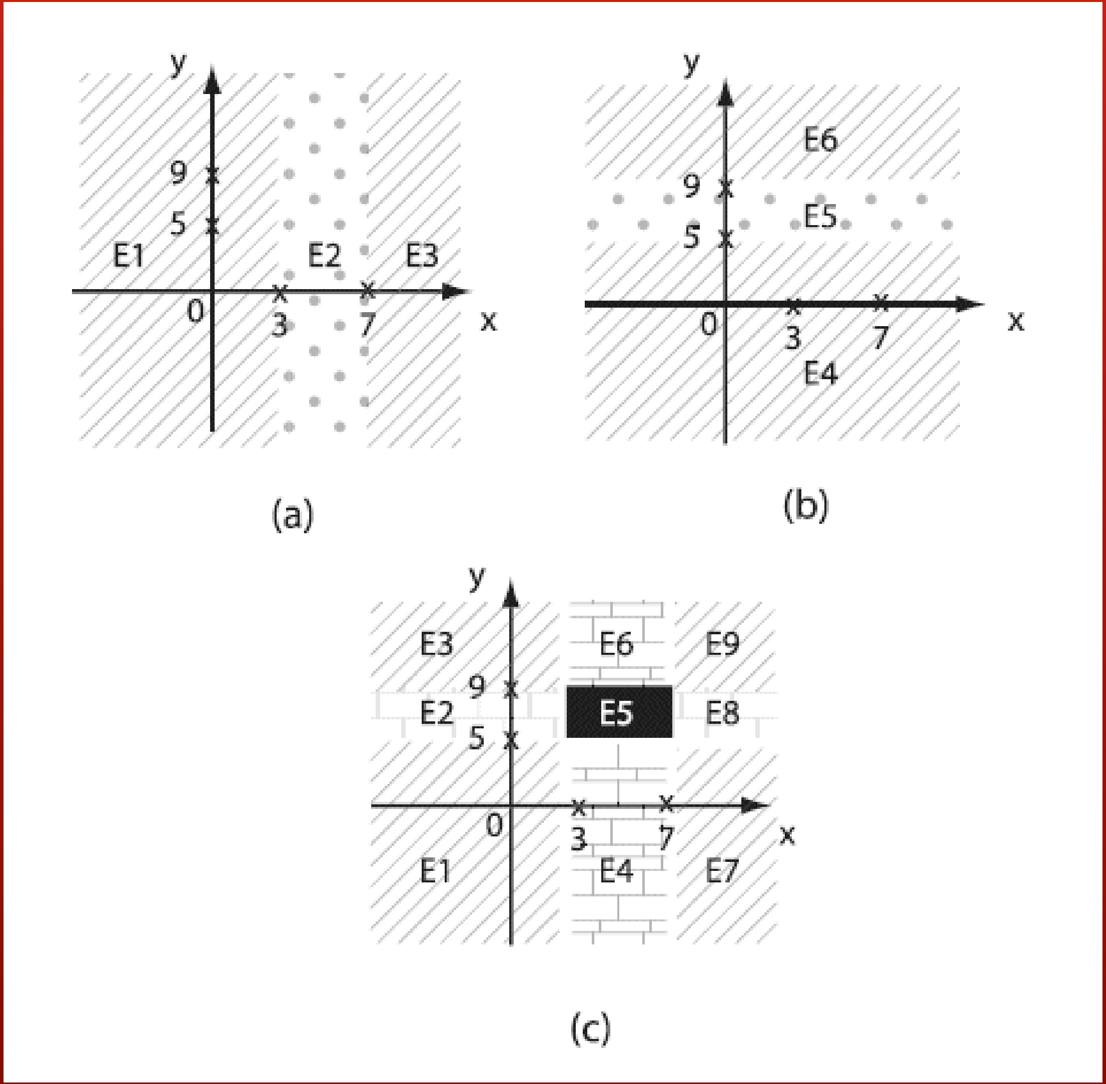
When Mathur discusses multi-d partitioning, he seems to assume the test selection process will produce one test per “region”.

Problem 2 from the homework makes a nice illustration.

- 2.11 An application takes two inputs x and y where $x \le y$ and $-5 \le y \le 4$. (a) Partition the input domain using unidimensional partitioning. Derive test sets based on the partitions created in (a).

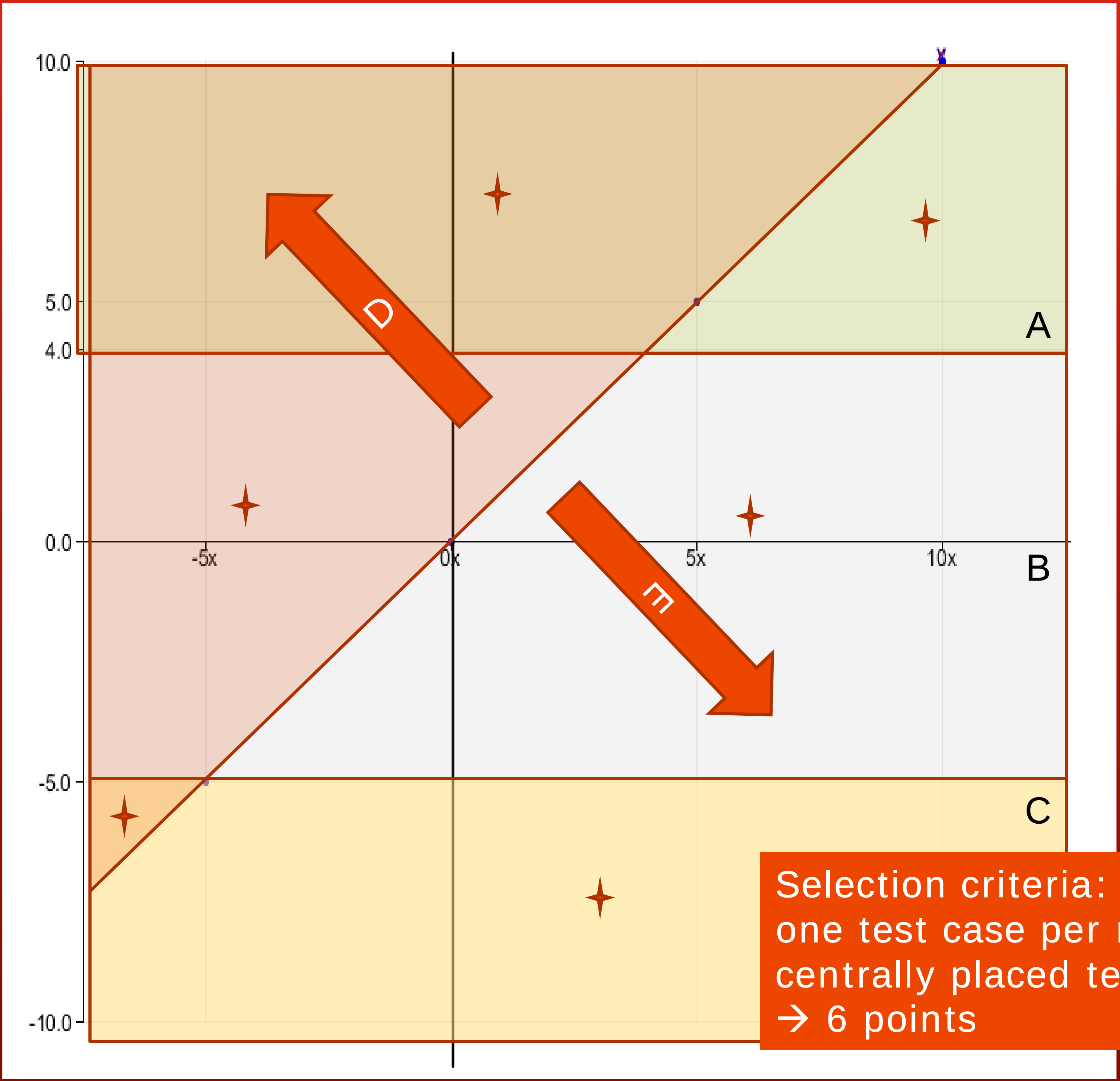
From Mathur, chapter 2 exercises.

Mathur’s Example



Under all combinations, both views yield the same number of tests.

But representative points chosen from the middle of each block don’t tell you much about boundaries.

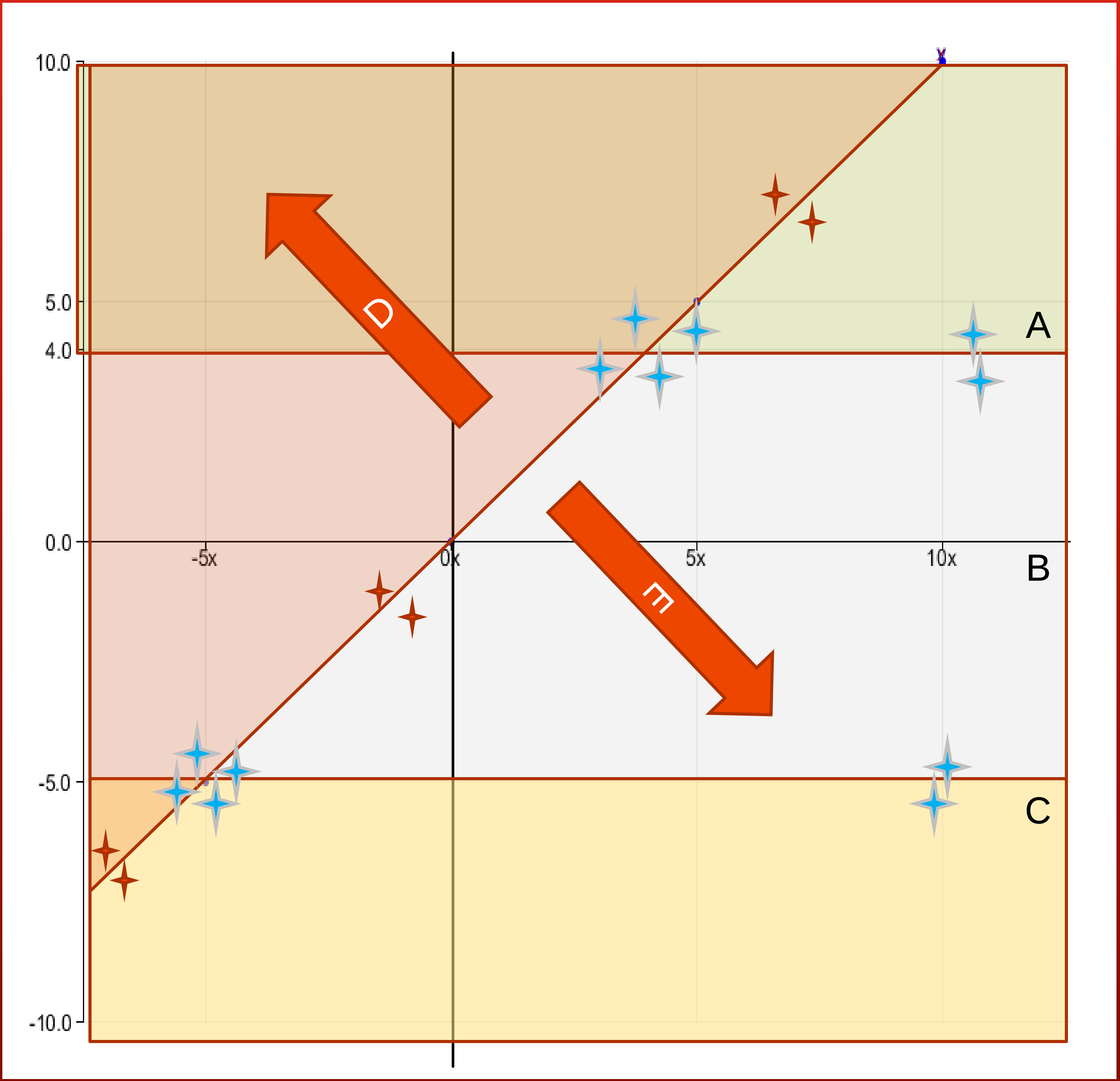


A: y>4
B: -5..4
C: y< -5

D : x <= y
E: x > y

How many points?

Selection criteria:
one test case per region,
centrally placed tests
→ 6 points



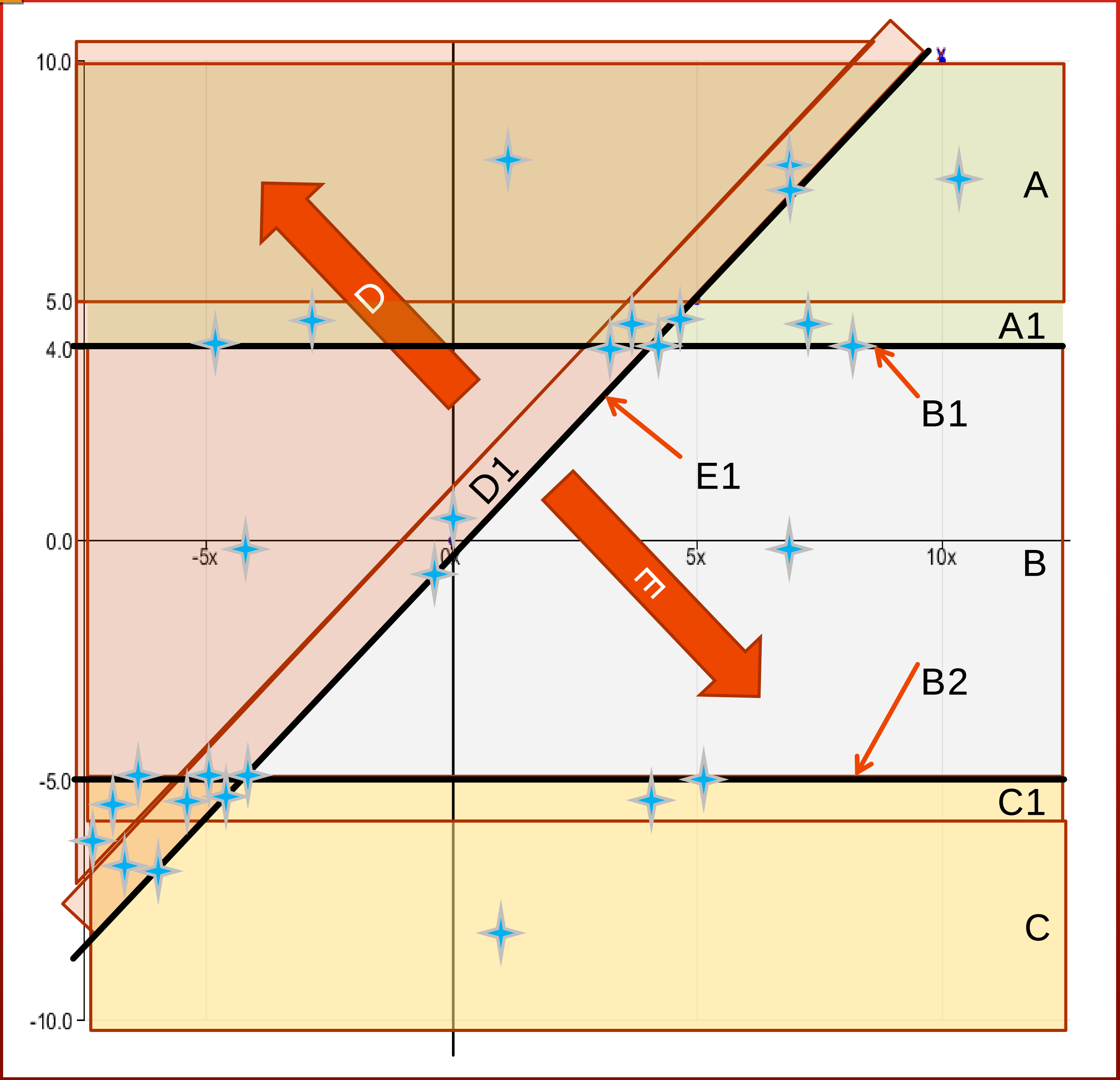
A: y>4
B: -5..4
C: y< -5

D : x <= y
E: x > y

6?
12?

Driving tests to specific regions

- The next slide shows how additional partitions near boundaries can be use to drive tests to areas likely to have certain bugs.
- In this case we choose positions that will position test points in places that would correspond to bugs caused by typing 4 instead of 5 or < instead of <=, for example.
- The test selection criteria in the next slide is one test per region, as it was in the first slide in this series.
- Note, however, that the number of tests grows from the original 6 to 28. That is a substantial increase in test execution cost.
- The heavy black lines (labelled B1, B2, and E1) denote partitions that are a single point wide.
- Note that the definition of these partitions works for reals as well as integers.



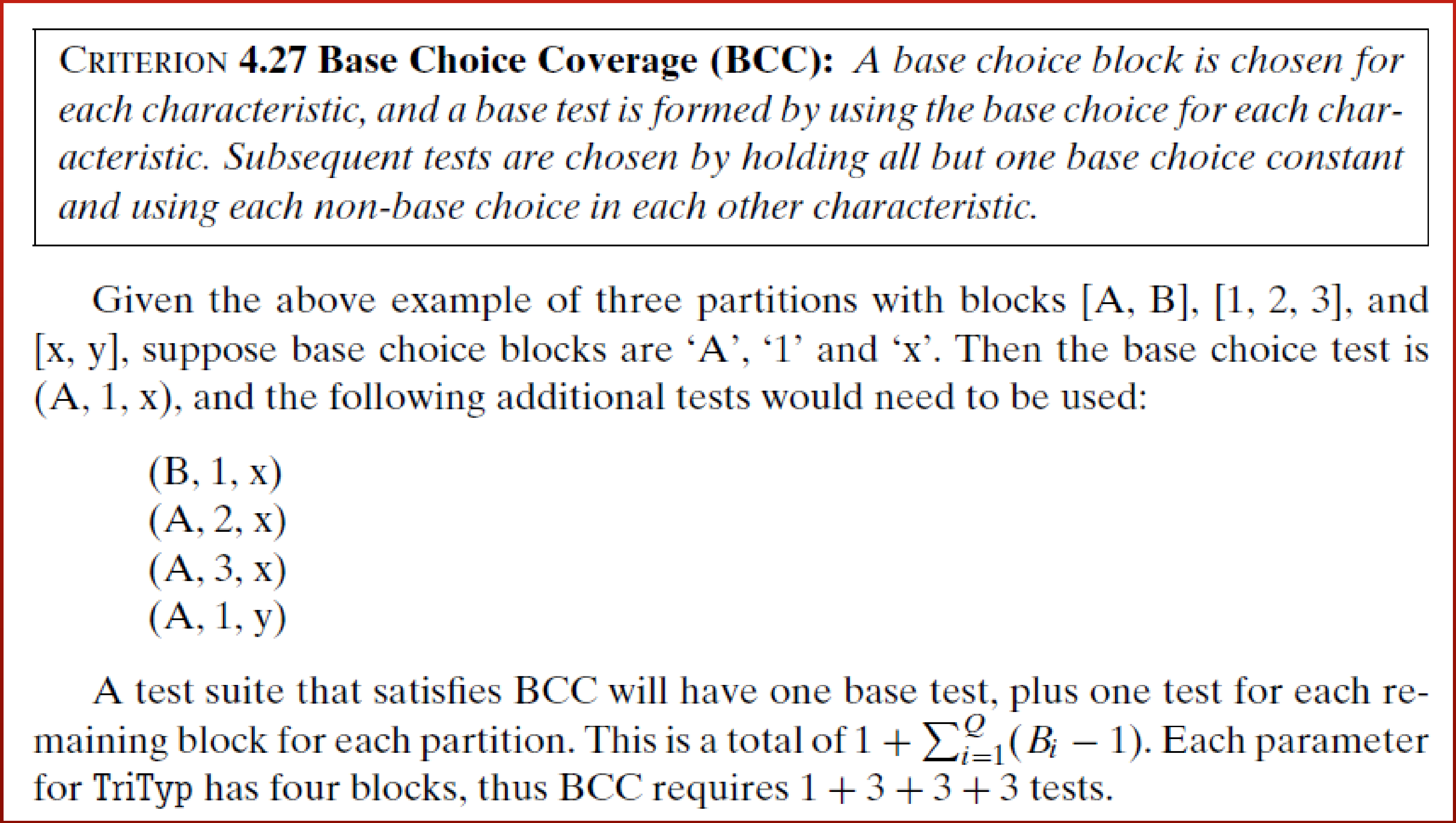
Added Paritions can “drive” tests to critical regions:

A: y > 5
A1: y∈[5..4)
B1: y = 4
B: y∈(-5..4)
B2: y = -5
C1: y∈[-6..-5)
C: y < -6

D : x < y-1
D1: x∈(y..y-1]
E1: x = y
E: x > y

Additional Combination Coverage Criteria

Base Choice Coverage



CRITERION **4.27 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.*

Given the above example of three partitions with blocks [A, B], [1, 2, 3], and [x, y], suppose base choice blocks are 'A', '1' and 'x'. Then the base choice test is (A, 1, x), and the following additional tests would need to be used:

(B, 1, x)
(A, 2, x)
(A, 3, x)
(A, 1, y)

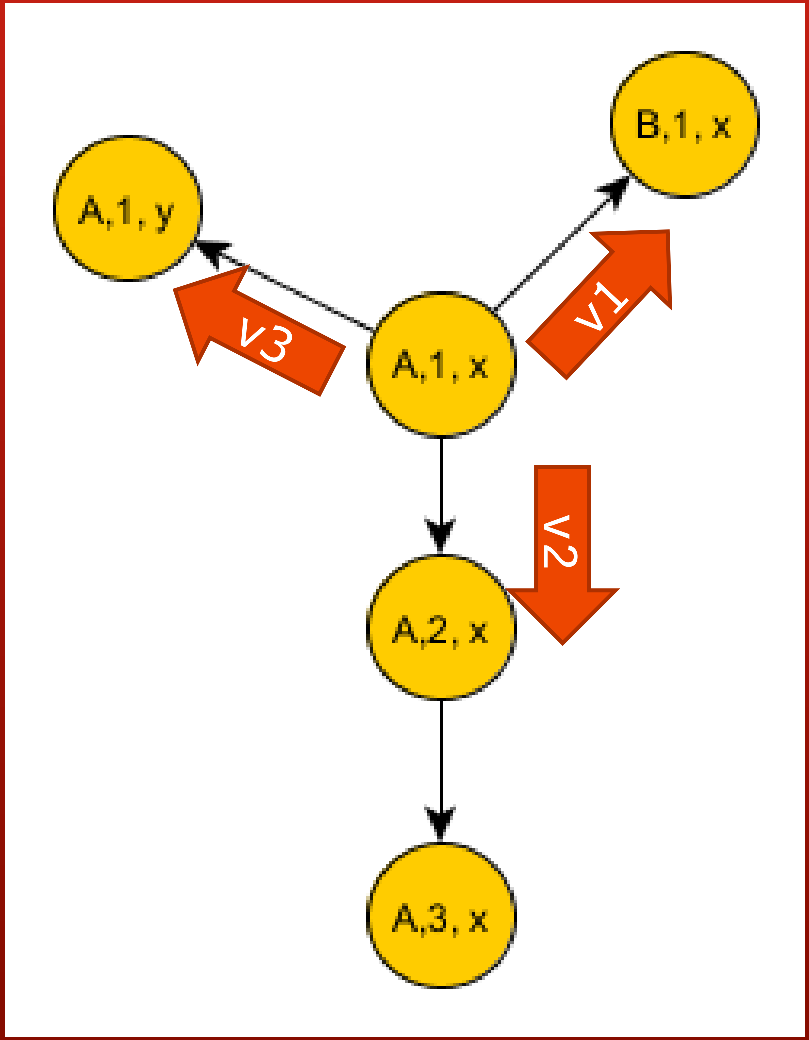
A test suite that satisfies BCC will have one base test, plus one test for each remaining block for each partition. This is a total of $1 + \sum_{i=1}^{Q}(B_i - 1)$. Each parameter for TriTyp has four blocks, thus BCC requires $1 + 3 + 3 + 3$ tests.

From Ammann and Offutt

Visualizing Base Choice

Param	Blocks
v1	A, B
v2	1, 2, 3
v3	x, y

Think of v1, v2, and v3 as edges of a cube. A,1,x is a external vertex -- i.e, the intersection of the three axes defined by the domains v1,v2,v3.



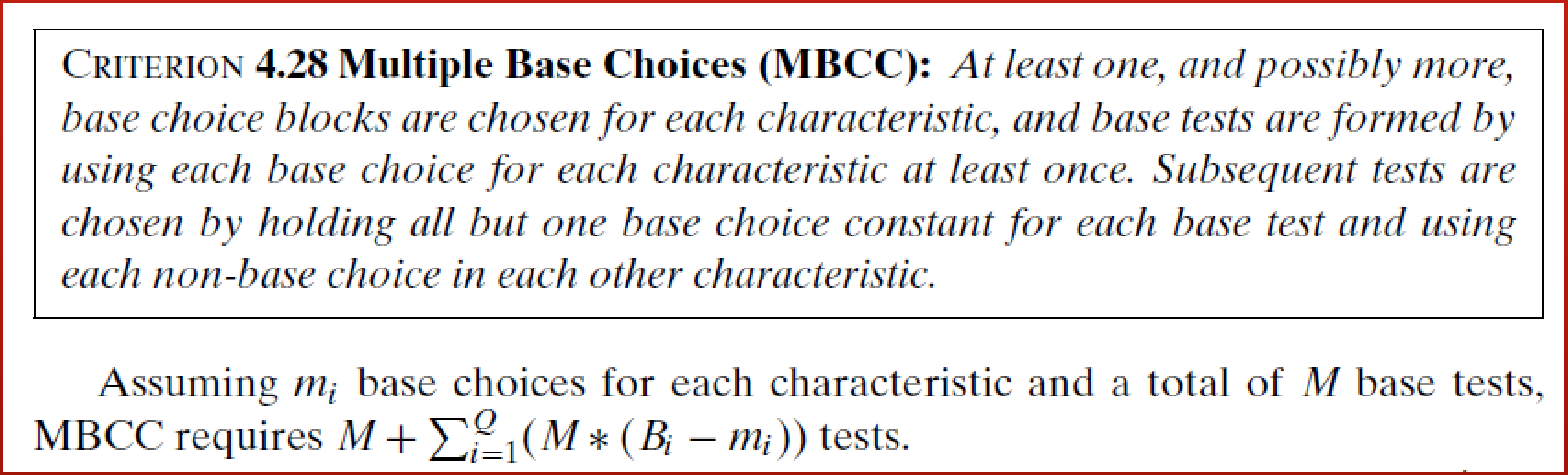
Additional Combination Coverage Criteria
Base Choice Coverage

- Pro
  - Benefits from knowledge about importance of certain points within the input space
    - The base choice, for example, might be the most likely from an end-user point of view. – perhaps the first in a pull-down or the default selection.
    - Especially useful with high-order partitions
  - Fewer tests than all combinations
- Variant
  - choose specific increments over a bounded range within the domain.

From Ammann and Offutt

Additional Combination Coverage Criteria

Multiple Base Choice Coverage



CRITERION **4.28 Multiple Base Choices (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.*

Assuming m_i base choices for each characteristic and a total of M base tests, MBCC requires $M + \sum_{i=1}^{Q}(M * (B_i - m_i))$ tests.

- Interesting theoretical generalization of Base Choice, but unlikely to be cost effective for M > 2.

From Ammann and Offutt

Questions

Reading, Last Lecture, Lab?