Domain Testing

Input Domains

- An exhaustive testing of values in the input domains is impossible.
- One is limited to a small subset of all possible input values.
- One wants to select a subset with the highest probability of finding the most errors.

An Example

- Consider a program that calculates the root of quadratic equations in the form of:

  $a\,x^2 + b\,x + c = 0$

  with the solution for the root to be:

  $r = (-b \pm \sqrt{b^2 - 4\,a\,c})\,/\,(2\,a)$.

- If each variable is represented by a 32 bit floating point number, the number of all possible input value combinations is then

  $2^{32} \times 2^{32} \times 2^{32} = 2^{96}$.

Equivalence Classes

- A well-selected set of input values should covers a large set of other input values.
- This property implies that one should partition the input domains into a finite number of equivalence classes.
- A test of a representative value of each class is equivalent to a test of any other value.

Valid and Invalid Equivalence Classes

- The equivalence classes are identified by taking each input condition and partitioning the input domain into two or more groups.
- Two types of equivalence classes are identified.
- Valid equivalence classes represent valid inputs to the program.
- Invalid equivalence classes represent all other possible states of the condition.

An Example

- If an input condition specifies a range of values (e.g., the count can be from 1 to 999), it identifies one valid equivalence class ($1 \leq count \leq 999$) and two invalid equivalence classes ($count < 1$ and $count > 999$)

Partitioning Valid Equivalence Classes

- If elements in a valid equivalence class are not handled in an identical manner by the program, partition the equivalence class into smaller equivalence classes.
- Generate a test case for each valid and invalid equivalence class.

An Example

- For the quadratic equation example, the types of the roots for the equation depend on the condition $d = b^2 - 4\,a\,c$.
- The equation has two different real roots if $d > 0$.
- The equation has two identical real roots if $d = 0$.
- The equation has no real root if $d < 0$.

An Example

Test Case	Condition $d = b^2 - 4ac$	Input a	b	c
1	$d > 0$	1	2	-1
2	$d = 0$	1	2	1
3	$d < 0$	1	2	3

Input Spaces, Vectors, Points

- Let x_1, x_2, …, x_n denote the input variables. Then these *n* variables form an *n*-dimensional space that we call input space.
- The input space can be represented by a vector X, we call input vector, where $X = [x_1, x_2, \ldots, x_n]$.
- When the input vector X takes a specific value, we call it a test point or a test case, which corresponds to a point in the input space.

Input Domains and Sub-Domains

- The input domain consists of all the points representing all the allowable input combinations specified for the program in the product specification.
- An input sub-domain is a subset of the input domain. In general, a sun-domain can be defined by a set of inequalities in the form of

  $f(x_1, x_2, \ldots, x_n) < K$,

  where "<" can also be replaced by other relational operators.

Input Domain Partition

- An input domain partition is a partition of the input domain into a number of sub-domains.
- These partitioned sub-domains are mutually exclusive, and collectively exhaustive.

Boundary

- A boundary is where two sub-domains meet.
- A boundary is a linear boundary if it is defined by:

  $$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = K.$$

  Otherwise, it is called a nonlinear boundary.
- A sub-domain is called a linear sub-domain if its boundaries are all linear ones.
- A point on a boundary is called a boundary point.

Open and Closed Boundary

- A boundary is a closed one with respect to a specific sub-domain if all the boundary points belong to the sub-domain.
- A boundary is an open one with respect to a specific sub-domain if none of the boundary points belong to the sub-domain.
- A sub-domain with all open boundaries is called an open sub-domain; One with all closed boundaries is called a closed sub-domain; otherwise it is a mixed sub-domain.

Interior and Exterior Points

- A point belonging to a sub-domain but not on the boundary is called an interior point.
- A point not belonging to a sub-domain and not on the boundary is called an exterior point.
- A point where two or more boundaries intersect is called a vertex point.

General Problems with Input Values

- Some input values cannot be handled by the program. These input values are under-defined.
- Some input values result in different output. These input values are over-defined.
- These problems are most likely to happen at boundaries.

Boundary Problems

- Closure problem: whether the boundary points belong to the sub-domain.
- Boundary shift problem: where exactly a boundary is between the intended and the actual boundary.

  $f(x_1, x_2, \ldots, x_n) = K$,

  where a small change in K.
- Boundary tilt problem: $f(x_1, x_2, \ldots, x_n) = K$, where a small change in some parameters.

Boundary Problems

- Missing boundary problem: a boundary missing means that two neighboring sub-domains collapse into one sub-domain.
- Extra boundary problem: An extra boundary further partitions a sub-domain into two smaller sub-domains.

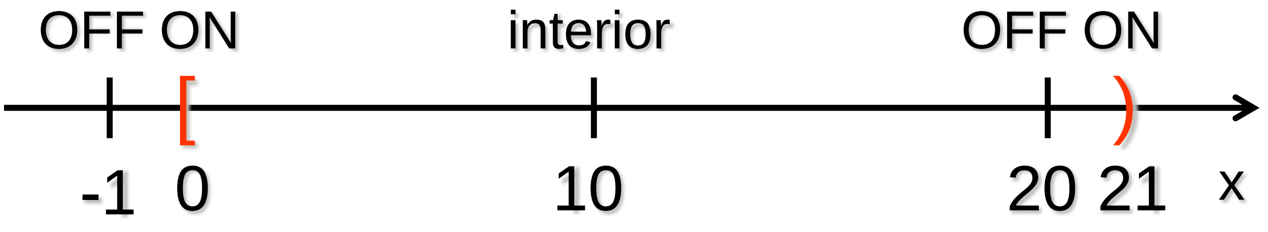
Weak N × 1 Strategy

- In an *n*-dimensional space, a boundary defined by a linear equation in the form of
  $f(x_1, x_2, \ldots, x_n) = K$
  would need *n* linearly independent points to define it.
- We can select *n* such boundary points, called ON points, to precisely define the boundary.
- We can also select a point, called an OFF point, that receives different processing.

The OFF Points

- If the boundary is a closed boundary with respect to the sub-domain under consideration, the OFF point will be outside the sub-domain or be an exterior point.
- If the boundary is an open boundary with respect to the sub-domain under consideration, the OFF point will be inside the sub-domain or be an interior point.

An Example



Distance of the OFF Points

- The idea is to pick the OFF point so close to the boundary that any small amount of boundary change would affect the processing of the OFF point.
- In practice, the distance ε to the boundary is set to the precision of the data type. For integers, $\varepsilon = 1$. For numbers with n binary digits after the decimal point, $\varepsilon = 1/2^n$.

Position of the OFF Points

- The selected OFF point should be central to all the ON points.
- For two-dimensional space, it should be chosen by:
- Choosing the midpoint between the two ON points.
- Then moving ε distance off the boundary, outward or inward for closed or open boundary, respectively.

Total Test Points

- In general, an interior point is also sampled as the representative of the equivalence class representing all the points in the sub-domain under consideration, resulting in

  $$(n + 1) \times b + 1$$

  test points for each n-dimensional domain with b boundaries.

An Example

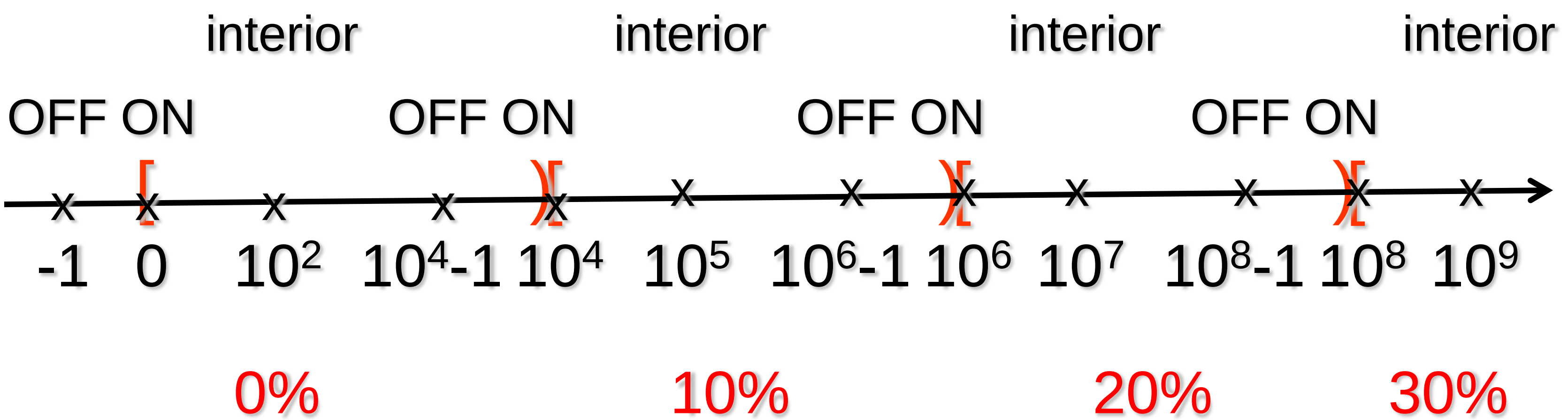
Tax Rate:

0%: 0~9999

10%: 10000~999999

20%: 1000000~99999999

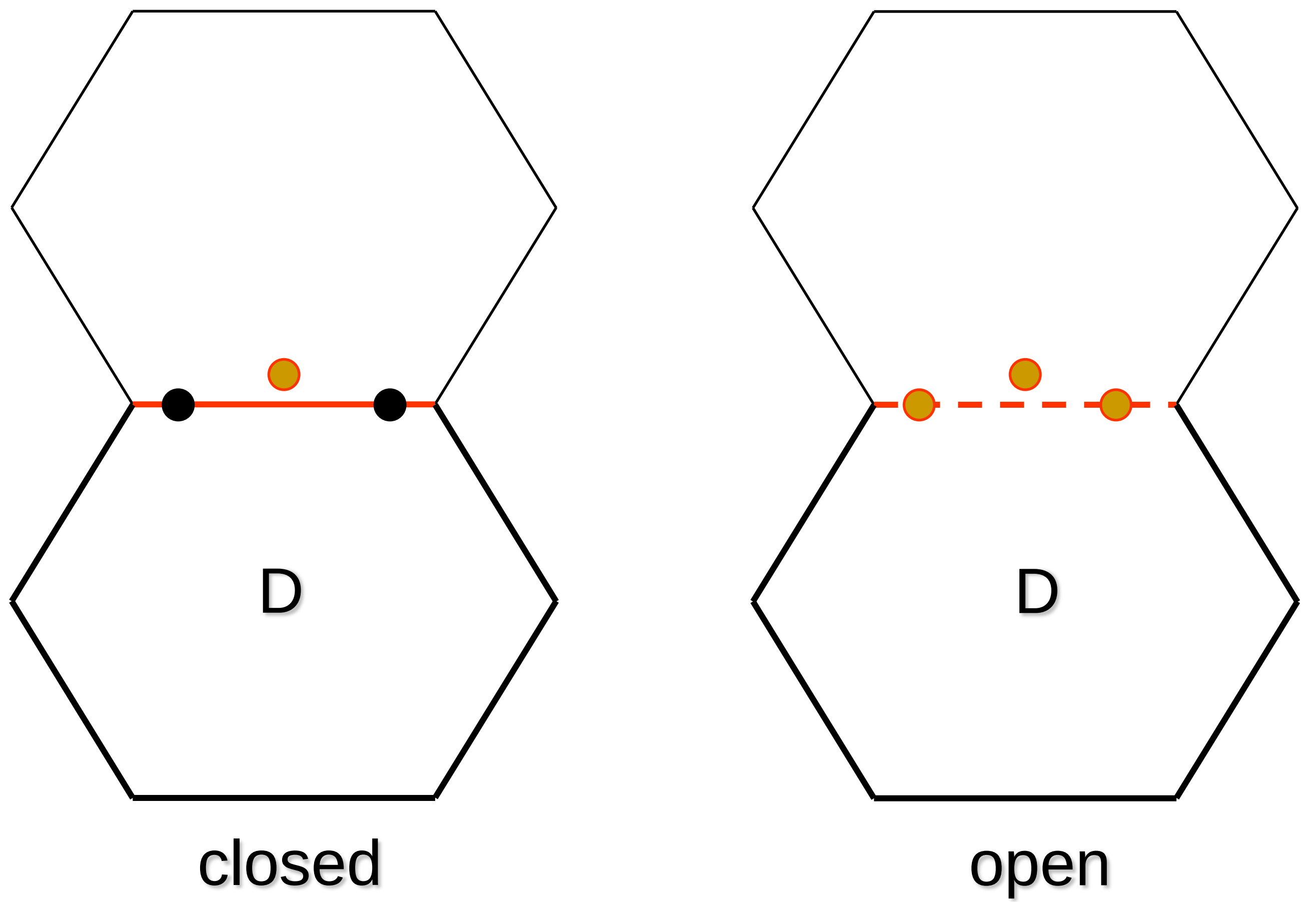
30%: 100000000~



Boundary Problem Detection of Weak N × 1 Strategy

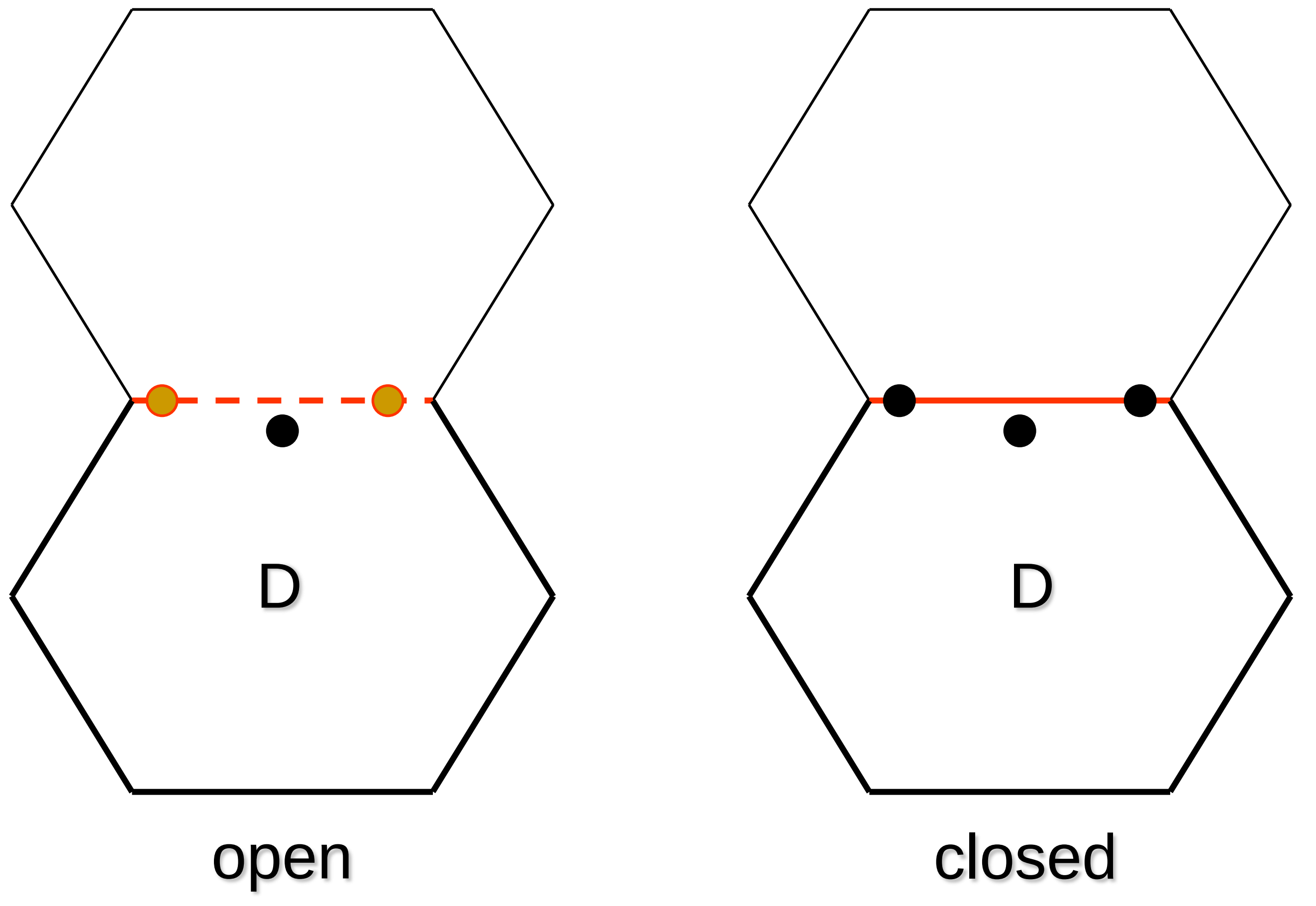
- Closure problem
- Boundary shift problem
- Boundary tilt problem
- Missing boundary problem
- Extra boundary problem

Closure Problem



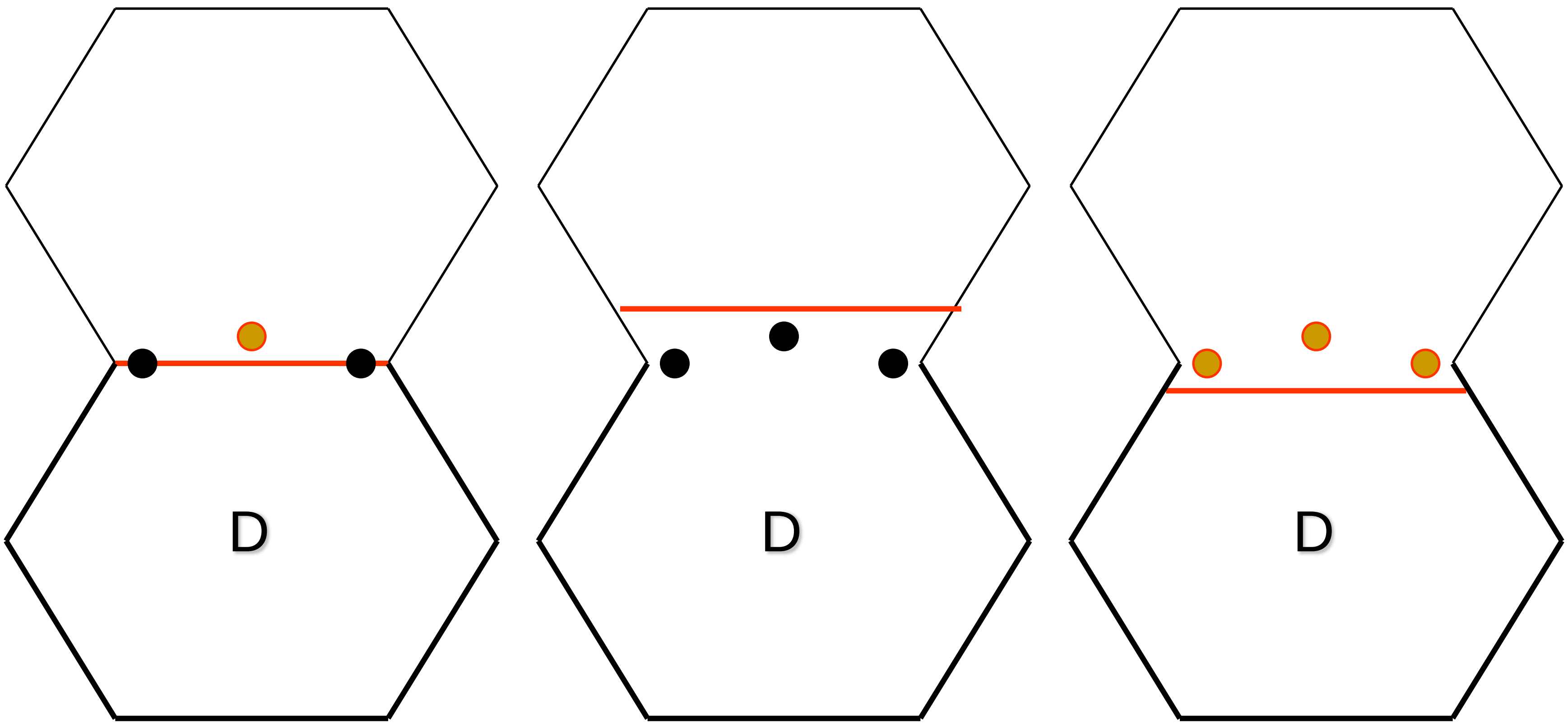
exterior interior

Closure Problem



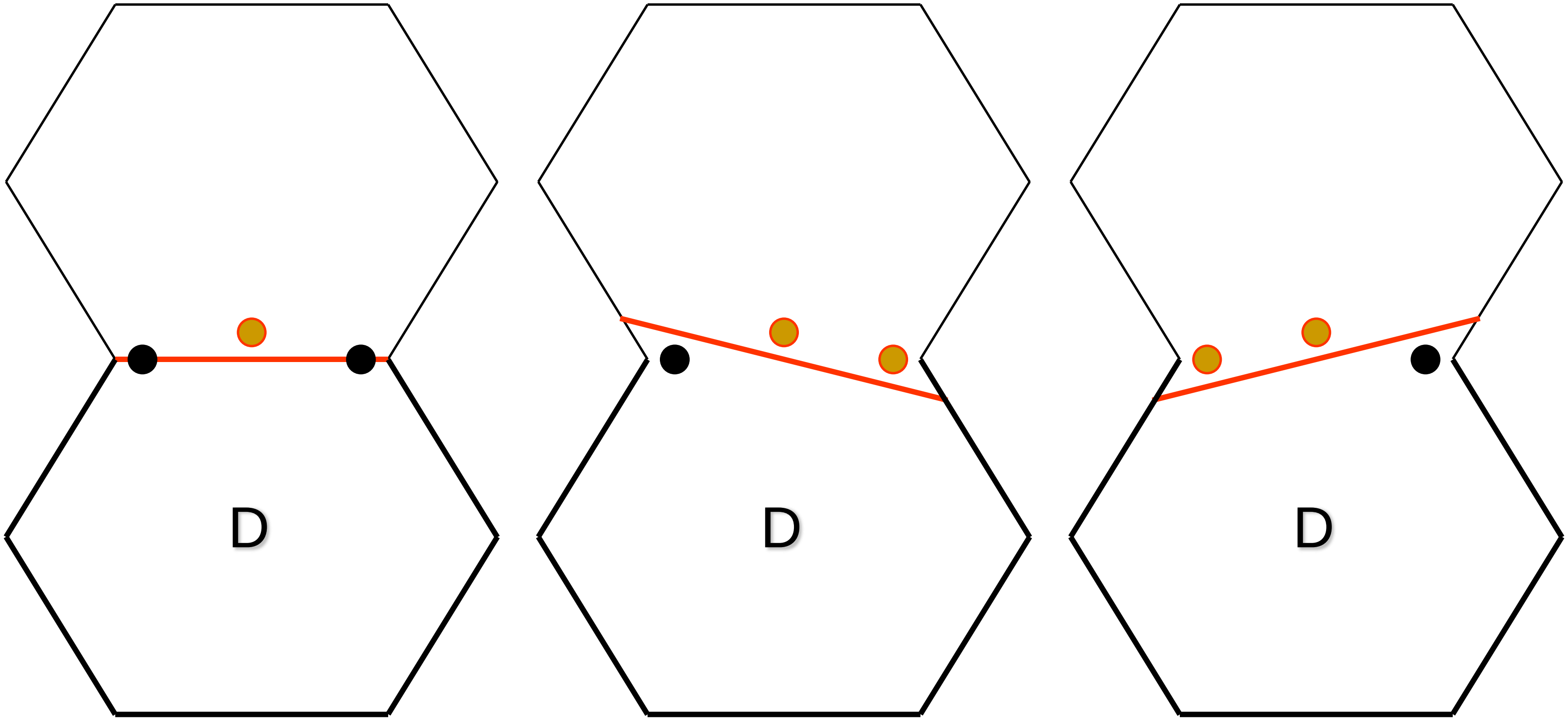
exterior   interior

Boundary Shift Problem



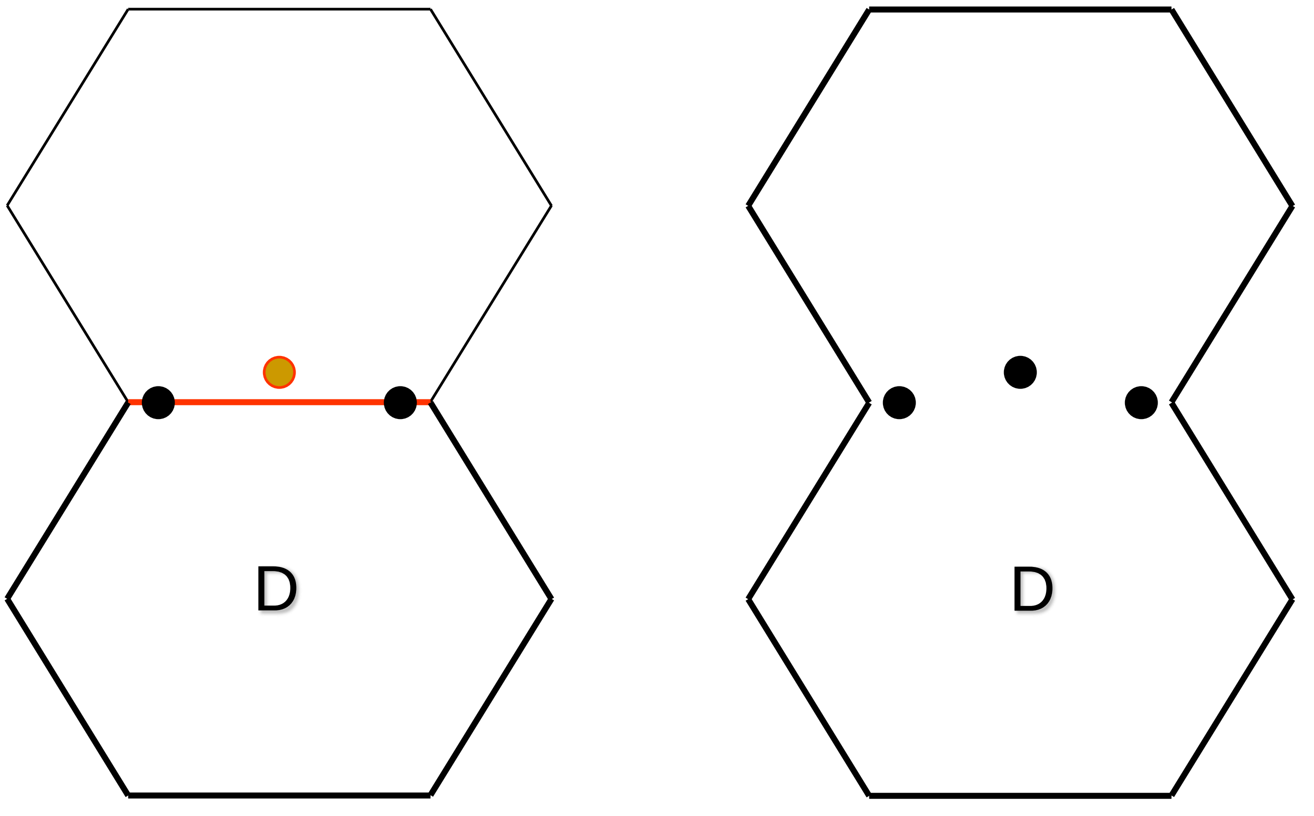
exterior interior

Boundary Tilt Problem



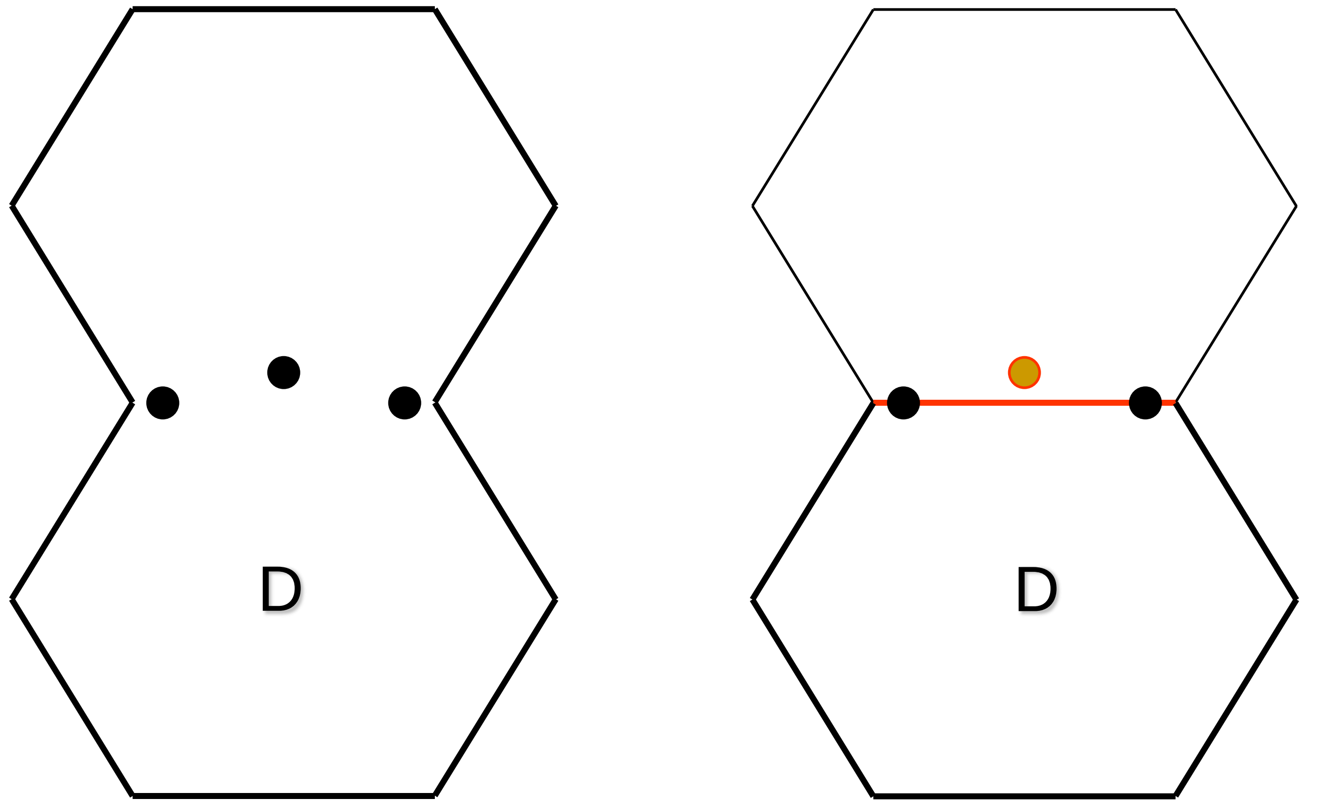
exterior   interior

Missing Boundary Problem



- exterior
- interior

Extra Boundary Problem



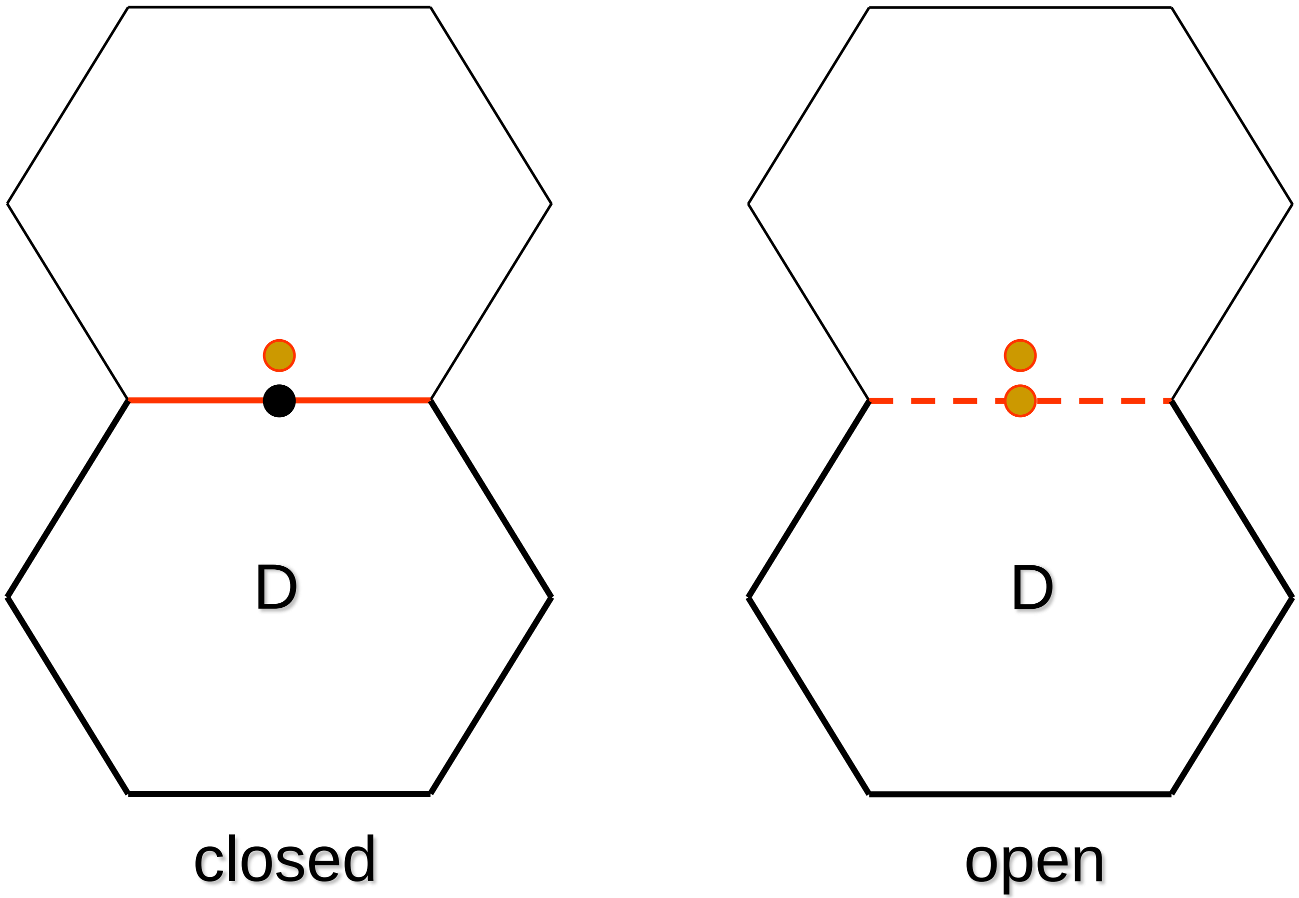
Weak 1×1 Strategy

- One of the major drawbacks of weak $N \times 1$ strategy is the number of test points used,
  $(n + 1) \times b + 1$
  for n input variables and b boundaries.
- Weak 1×1 strategy uses just one ON point for each boundary, thus reducing the total number of test points to $2 \times b + 1$.
- The OFF point is just ε distance from the ON point and perpendicular to the boundary.

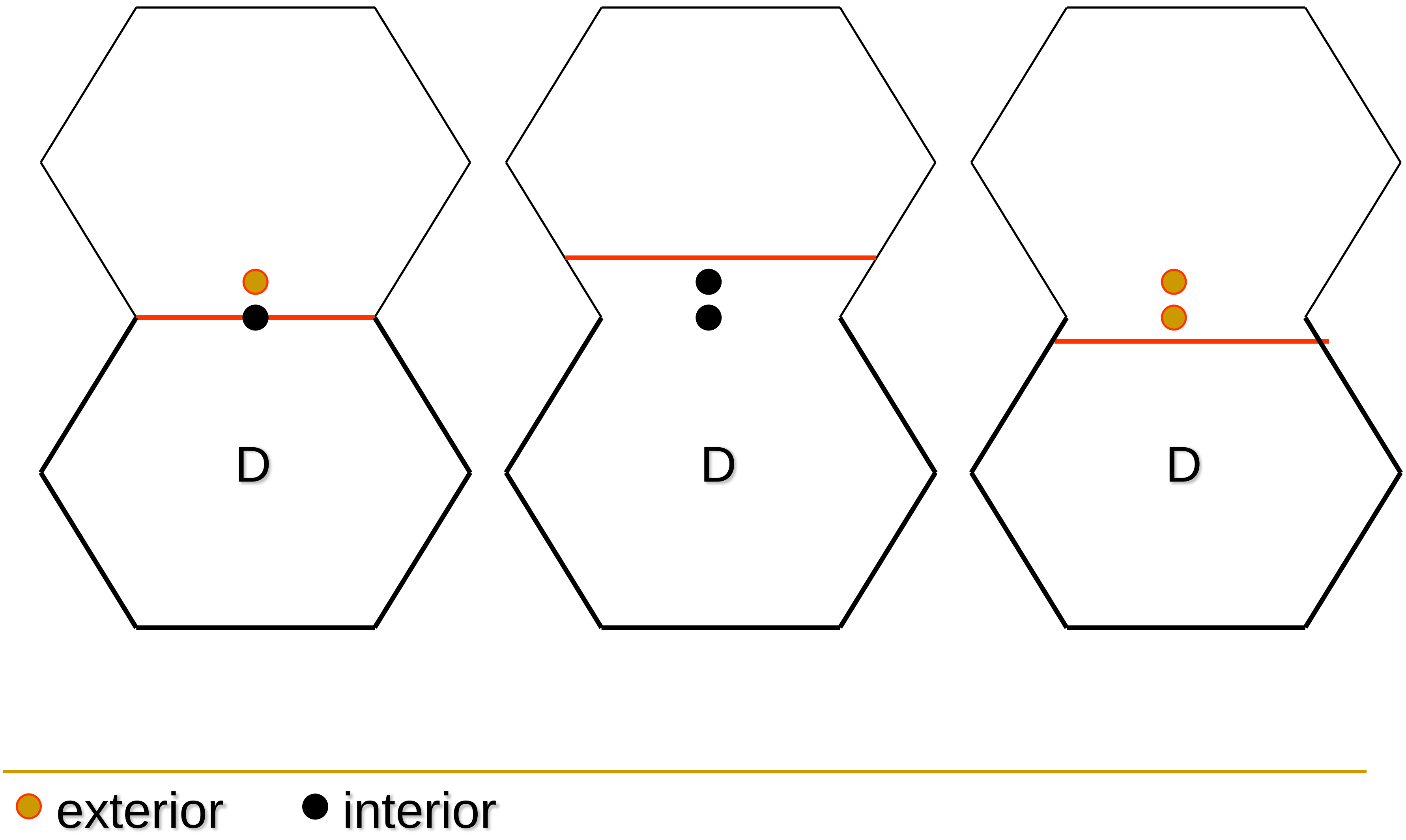
Boundary Problem Detection of Weak 1 × 1 Strategy

- Closure problem
- Boundary shift problem
- Boundary tilt problem
- Missing boundary problem
- Extra boundary problem

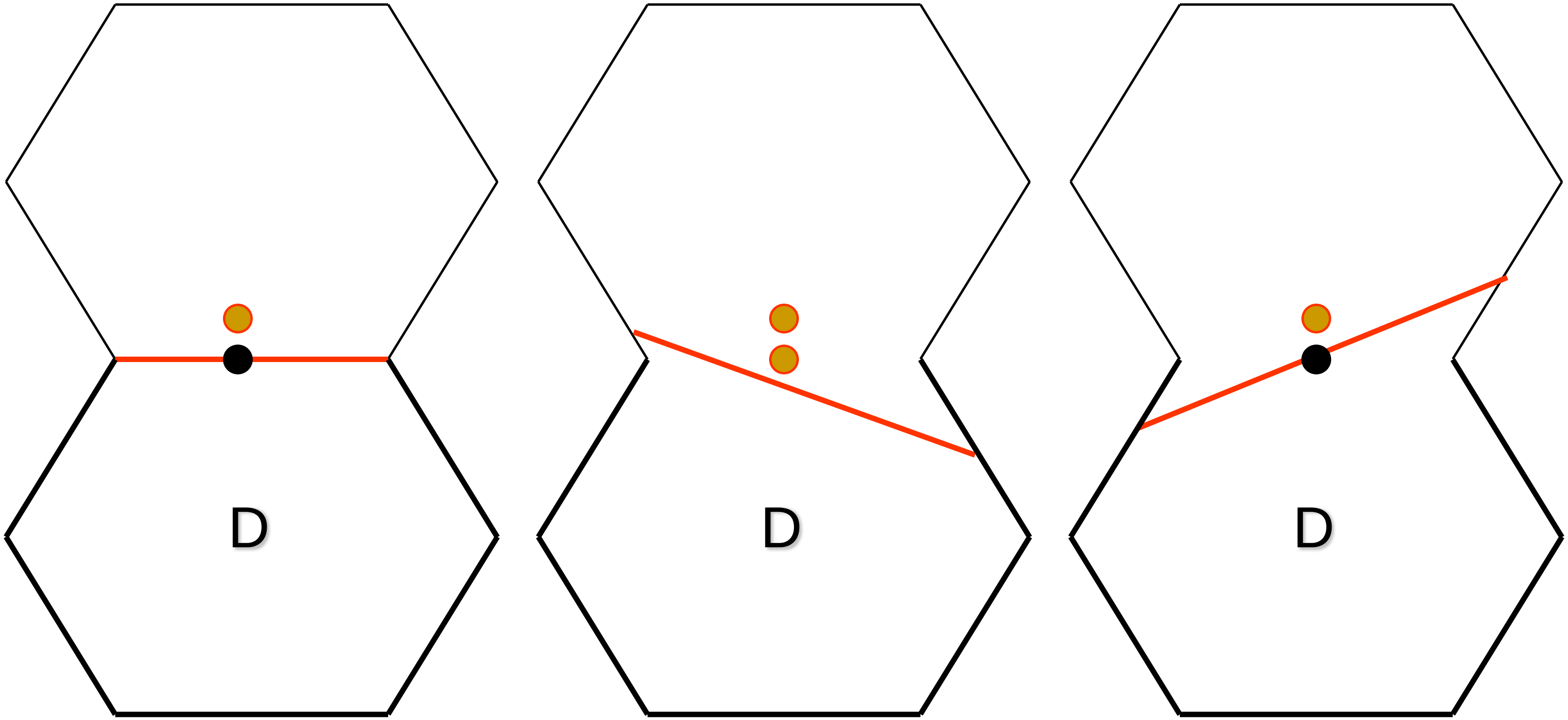
Closure Problem



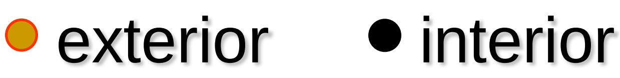
Boundary Shift Problem



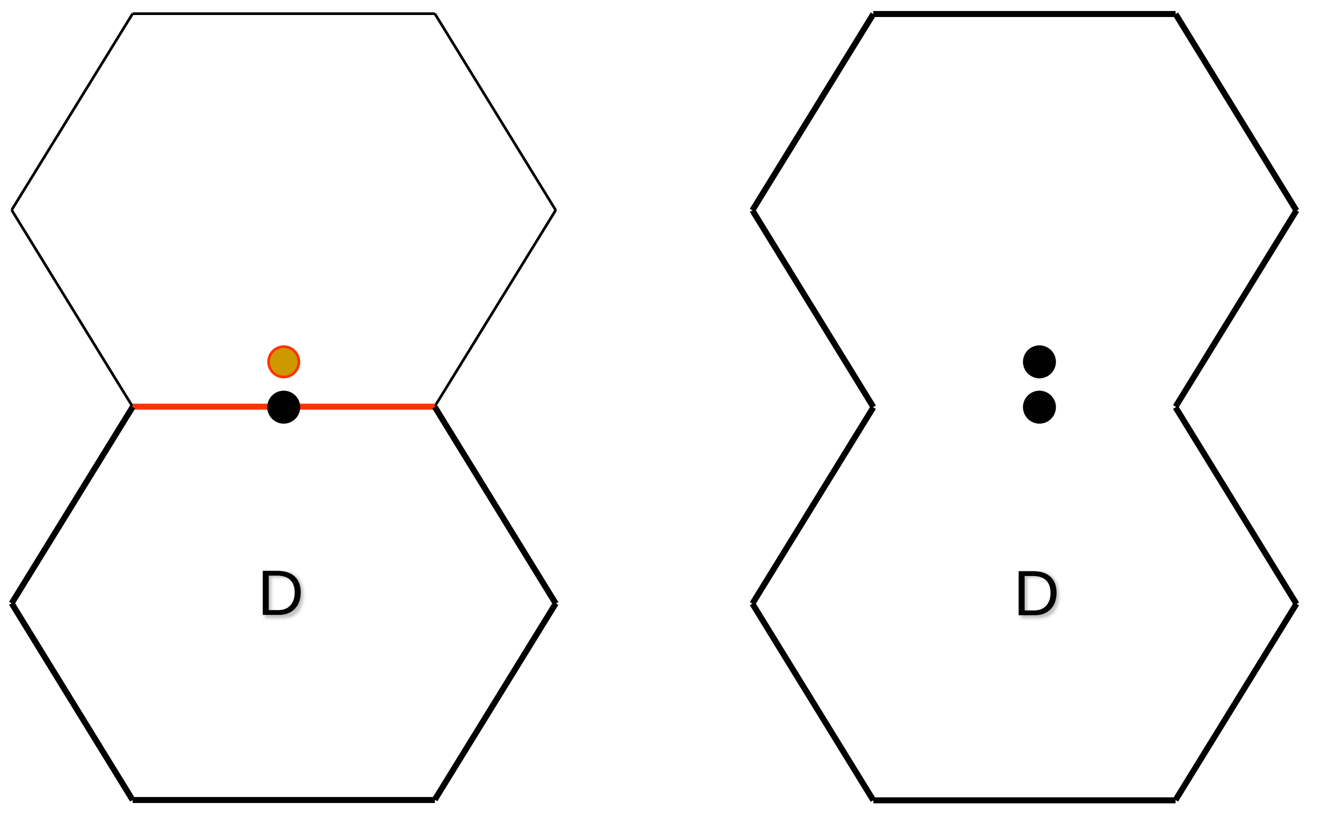
Boundary Tilt Problem



Such cases are rare

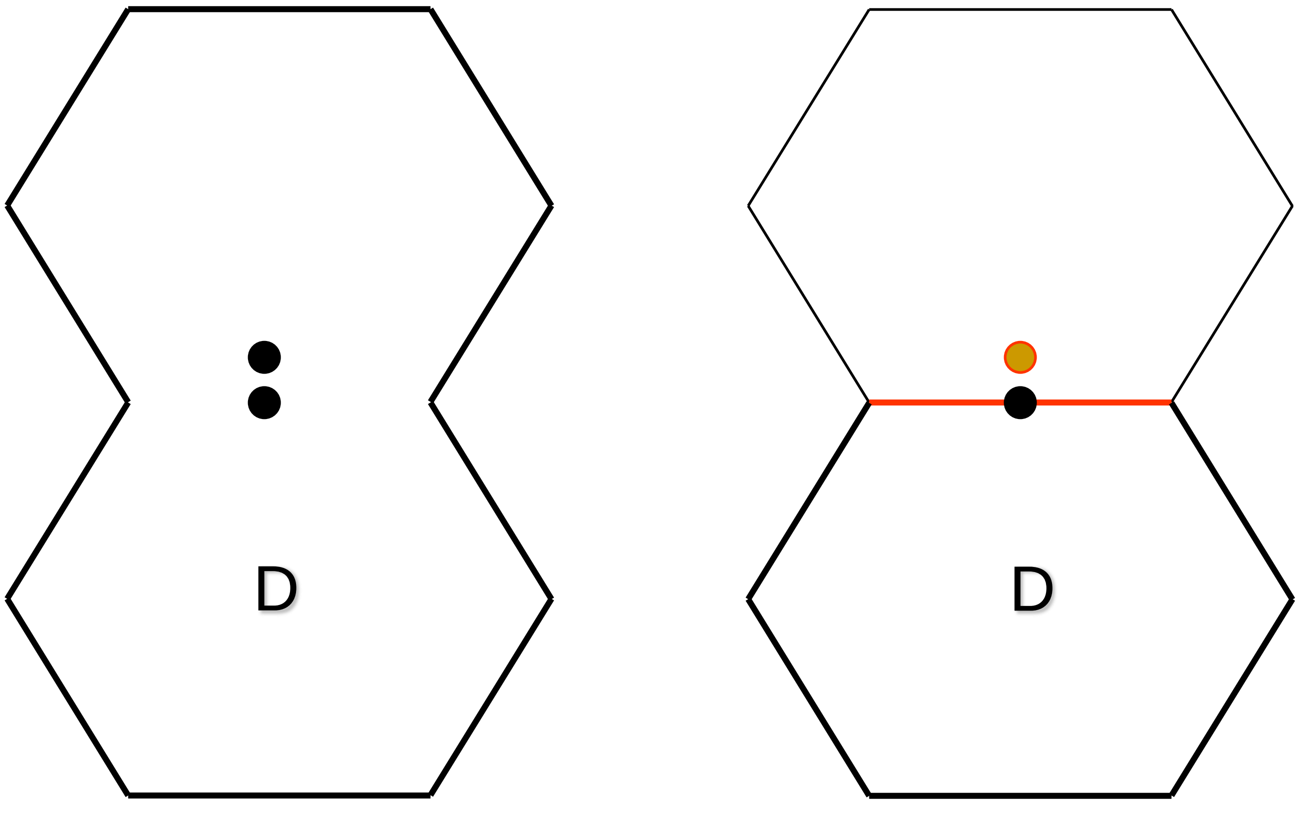


Missing Boundary Problem



exterior   interior

Extra Boundary Problem



exterior   interior

Looking for Equivalence Classes

- Don’t forget equivalence classes for invalid inputs.
- Organize your classifications into a table or an outline.
- Look for ranges of numbers.
- Look for membership in a group.
- Analyze responses to lists and menus.

Looking for Equivalence Classes

- Look for variables that must be equal.
- Create time-determined equivalence classes.
- Look for variable groups that must calculate to a certain value or range.
- Look for equivalent output events.
- Look for equivalent operating environments.

Don’t Forget Equivalence Classes for Invalid Inputs

- This is often your best source of bugs.
- For example, for a program that is supposed to accept any number between 1 and 99, there are at least four equivalence classes:
- 1~99.
- < 1.
- > 99.
- If it’s not a number, it is not accepted. (Is this true for all non-numbers?)

Organize Your Classifications into a Table or an Outline

- You will find so many input and output conditions and equivalence classes associated with them that you'll need a way to organize them.
- We use a table or an outline.

Table

Input or Output Event	Valid Equivalence Classes	Invalid Equivalence Classes
Enter a number	1~99	> 99
		0
		Negative numbers
		An expression that yields an invalid number, such as 5 – 5, which yields 0
		Letters and other non-numeric characters

Outline

1. Enter a number
1.1 Valid Case
1.1.1 1~99
1.2 Invalid Cases
1.2.1 > 99
1.2.2 0
1.2.3 Negative numbers
1.2.4 An expression that yields an invalid number, such as 5 – 5, which yields 0
1.2.5 Letters and other non-numeric characters

Look for Ranges of Numbers

- Every time you find a range (like 1~99), you've found several equivalence classes.
- There are usually three invalid equivalence classes: everything below the smallest number, everything above the largest number, and non-numbers.
- Look for multiple ranges (like tax rates). There is an invalid range below the lowest range and another above the highest range.

Look for Membership in a Group

- If an input must belong to a group, one equivalence class includes all members of the group.
- Another includes everything else.
- It might be possible to subdivide both classes further.
- For example, if you enter the name of a country, the valid equivalence class includes all countries' names. The invalid class includes all inputs that aren't country names.

Look for Membership in a Group

- But what of abbreviations, almost correct spelling, native language spelling, or names that are now out of date but were country names?
- Should you test these separately?
- The odds are good that the specification won’t anticipate all of these issues, and that you’ll find errors in test cases like these.

Analyze Responses to Lists and Menus

- You must enter one of a list of possible inputs. The program responds differently to each.
- Each input is its own equivalence class.
- The invalid equivalence class includes any inputs not on the list.
- For example, the input Are you sure? (Y/N). One class contains Y. Another contains N. Anything else is invalid.

Look for Variables That Must Be Equal

- You can enter any color you want as long as it's black. Not-black is the invalid equivalence class.
- Sometimes this restriction arises unexpectedly in the field: everything but black is sold out.
- Choices that used to be valid, but no longer are, belong in their own equivalence class.

Create Time-Determined Equivalence Classes

- Suppose you press the space bar just before, during, and just after the computer finishes reading a program from the disk. Tests like this crash some systems.
- Everything you do just before the program starts reading is another class.
- Everything you do long before the task is done is probably one equivalence class.
- Everything you do within some short time interval before the program finishes is another class.

Look for Variable Groups That Must Calculate to a Certain Value or Range

- Enter the three angles of a triangle.
- In the class of valid input, they sum to 180 degrees.
- In one invalid equivalence class, they sum to less than 180 degrees.
- In another they sum to more.

Look for Equivalent Output Events

- So far, we've stressed input events, because they're simpler to think about.
- A program drives a plotter that can draw lines up to four inches long.
- A line might be within the valid range.
- The program might try to plot a line longer than four inches
- There might be no line.
- It might try to plot something else altogether, like a circle.

Look for Equivalent Operating Environments

- The program is specified to work if the computer has between 64 and 256K of available memory.
- That's an equivalence class.
- Another class includes RAM configurations of less than 64K.
- A third includes more than 256K.