Software Testing and Quality Assurance

Theory and Practice

Chapter 6

Domain Testing

Outline of the Chapter

- Domain Error
- Testing for Domain Errors
- Sources of Domains
- Types of Domain Errors
- ON and OFF Points
- Test Selection Criterion
- Summary

Domain Error

- Two fundamental elements of a program
  - Input domain: The set of all input data to the program
  - Program path: A sequence of instructions from entry to exit
    - Feasible path: If there exists input data which causes the path to execute.
    - Infeasible path: No input data exists to cause the path to execute.
- Howden identified 2 classes of errors by combining input domain and program paths.
  - Computation error
  - Domain error
- Computation error
  - A computation error occurs when a specific input data causes the correct path to execute, but the output value is wrong.
- Domain error
  - A domain error occurs when a specific input data causes the program to execute a wrong (i.e. undesired) path.

Domain Error

- Idea
  - An input causes a path to execute.
  - Different input data (i.e. a set of inputs) cause the same path to execute.
  - Domain: A domain is a set of input values for which the program performs the same computation for every member of the set.
  - We are interested in maximal domains such that the program performs different computations in different domains.

Domain Error

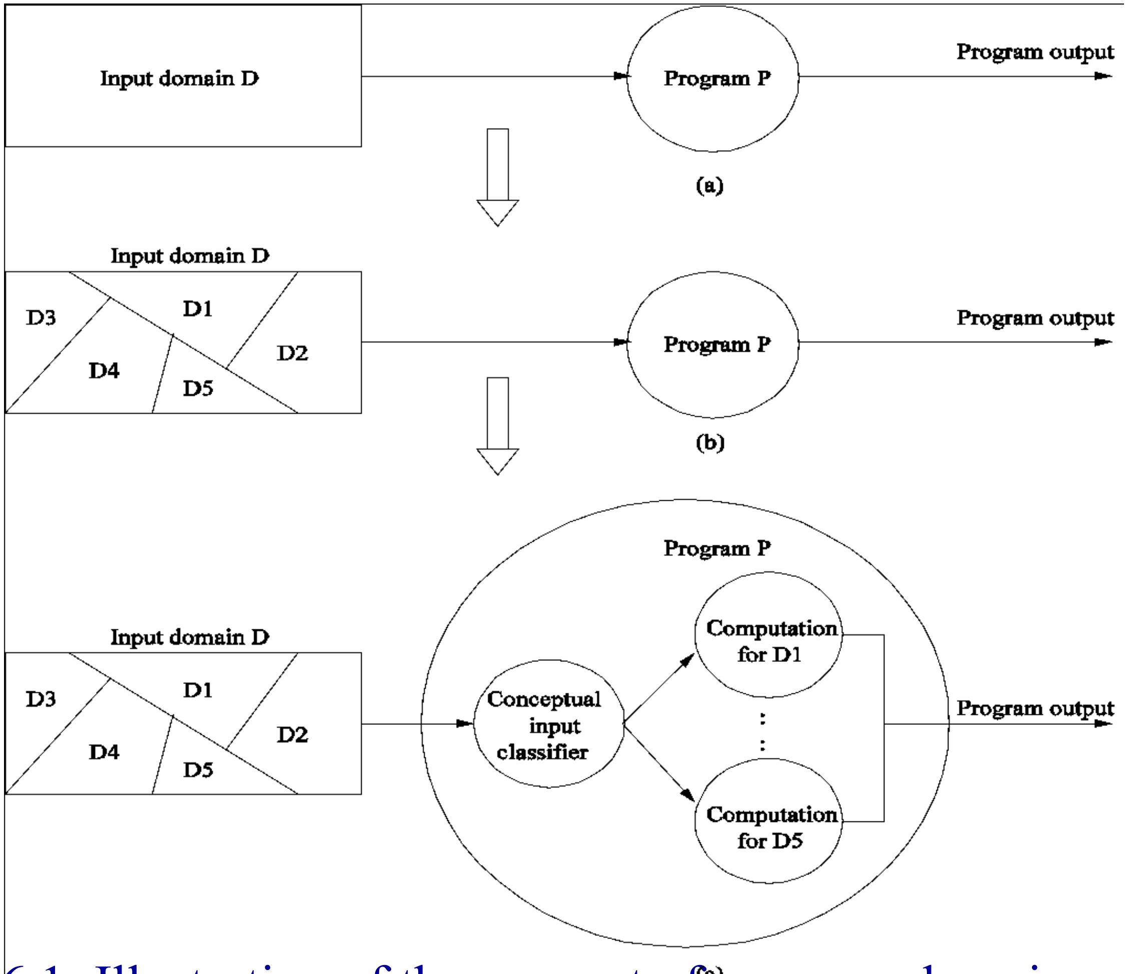

Figure 6.1: Illustration of the concept of program domains.

Domain Error

- A program is considered as an input classifier.

- A program is said to have a domain error if it incorrectly performs input classification.
  – Wrong classification: This means a wrong path is executed for a specific input data.

Testing for Domain Errors

- The idea of domain testing was first studied by White and Cohen.
- Domain testing differs from control/data flow testing.
- Flashback: control/data flow testing
  - **Note:** No assumption is made about any kind of error in the program.
  - Draw a graph – control flow or data flow.
  - Select paths based on path selection criteria: statement, branch, all-use ...
  - Generate input data from the selected paths.
- In contrast, in domain testing
  - One identifies a category of faults, known as domain errors.
- We will discuss the following.
  - Sources of domain
  - Types of domain errors
  - Selecting test data to reveal domain errors

Sources of Domains

```
int codedomain(int x, int y){
        int c, d, k
        c = x + y;
        if (c > 5) d = c - x/2;
        else       d = c + x/2;
        if (d >= c + 2) k = x + d/2;
        else            k = y + d/4;
        return(k);
    }
```

Fig. 6.2: A function to explain program domains.

Domain Error

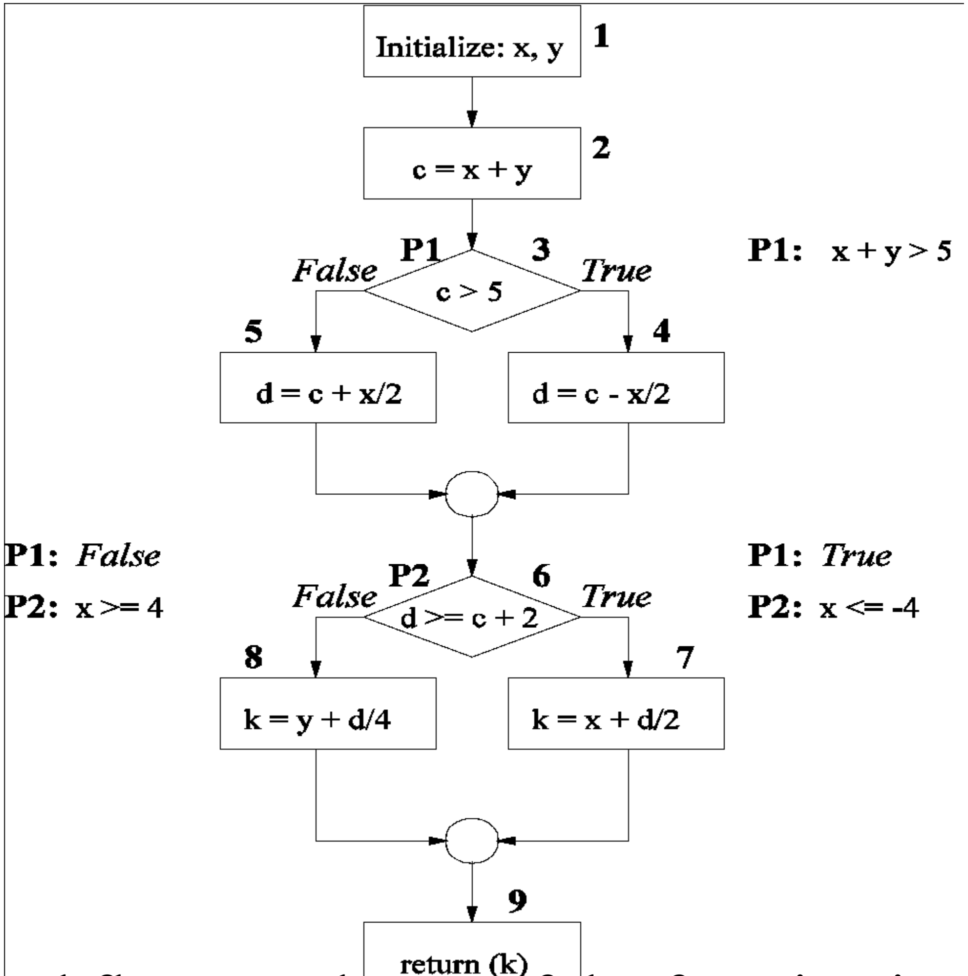

Figure 6.3: Control flow graph rep. of the function in Fig. 6.2.

Domain Error

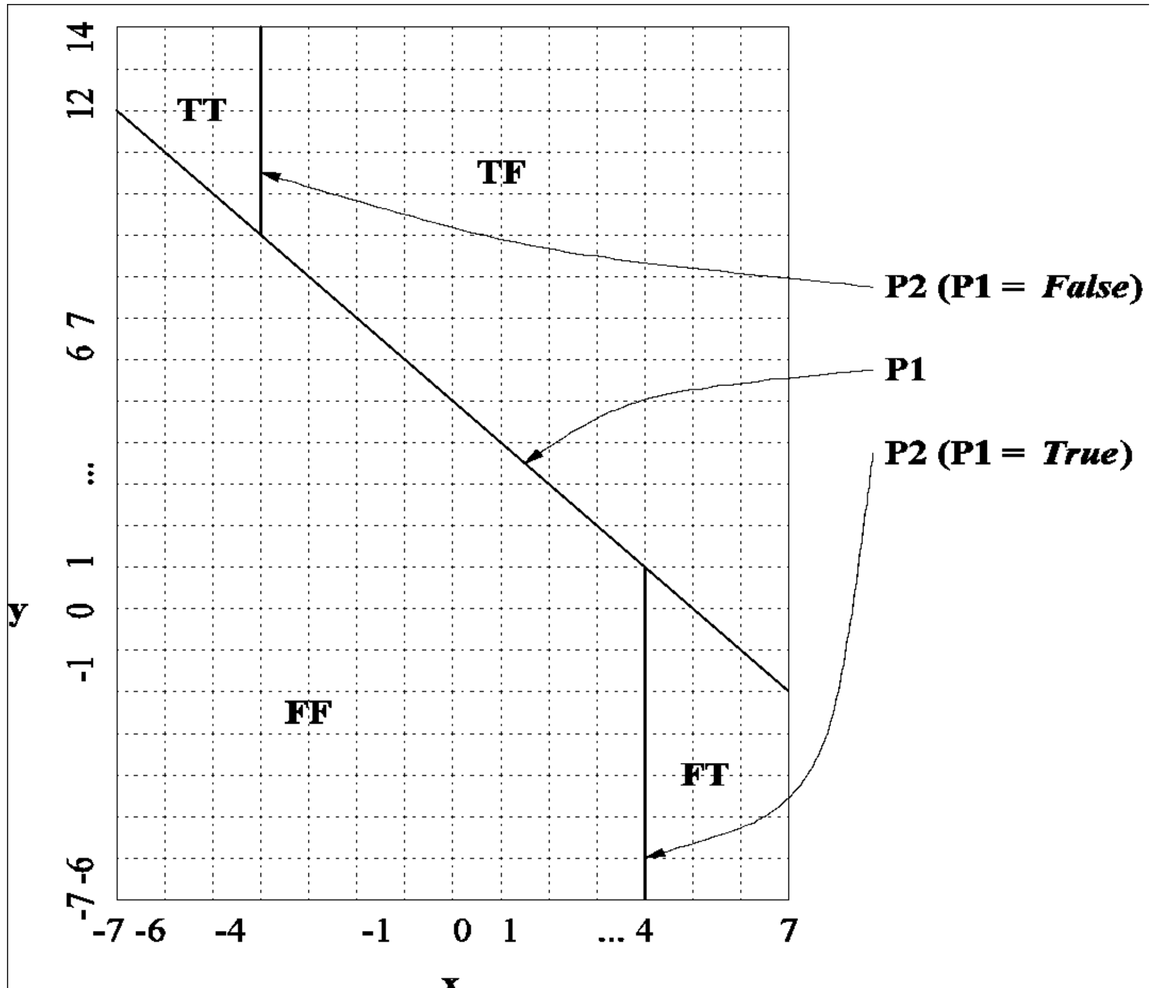

Figure 6.3: Control flow graph rep. of the function in Fig. 6.2.

Types of Domain Errors

- Recall that
  - A domain is a set of values for which the program performs identical computations.
  - A domain can be represented by a set of predicates. Individual elements of a domain satisfy the corresponding predicates.
- From a geometrical perspective, a domain is defined by a set of constraints, called *boundary inequalities*.
- The properties of a domain are discussed in terms of the properties of its boundaries.
  - Closed boundary
  - Open boundary
  - Closed domain
  - Open domain
  - Extreme point
  - Adjacent domain

Types of Domain Errors

- Closed boundary
  - A boundary is said to be **closed** if the points on the boundary are included in the domain of interest.
    - Example: Consider the TT domain in Fig. 6.4 and the boundary defined by the inequality P2: $x \leq -4$. This is a closed boundary of domain TT.
- Open boundary
  - A boundary is said to be **open** if the points of the boundary do not belong to the domain of interest.
    - Example: Consider the domain TT in Fig. 6.4 and its boundary defined by the inequality P1: $x + y > 5$. This is an open boundary of the domain TT.
- Closed domain
  - A domain is said to be closed if all of its boundaries are closed.
- Open domain
  - A domain is said to be open if some of its boundaries are open.

Types of Domain Errors

- Extreme point
  - An extreme point is a point where two or more boundaries cross.
- Adjacent domains
  - Two domains are said to be adjacent if they have a boundary inequality in common.

Types of Domain Errors

- Note
  - A program path will have a domain error if there is **incorrect formulation** of a path predicate.
  - An incorrect predicate expression causes a boundary segment to
    - be shifted from its correct position, or
    - have an incorrect relational operator
- A domain error can be caused by
  - An incorrectly specified predicate, or
  - An incorrect assignment which affects a variable used in the predicate.
- We focus on the following kinds of boundary errors.
  - Closure error
  - Shifted-boundary error
  - Tilted-boundary error

Types of Domain Errors

- Closure error
  - A closure error occurs if a boundary is open when the intention is to have a closed boundary, or vice versa.
  - Example: The relational operator $\leq$ is implemented as $<$.
- Shifted-boundary error
  - A shifted boundary error occurs when the implemented boundary is parallel to the intended boundary.
  - Example: Let the intended boundary be $x + y > 4$, whereas the actual boundary is $x + y > 5$.
- Tilted-boundary error
  - A tilted-boundary error occurs if the constant coefficients of the variables in a predicate defining a boundary take up wrong values.
  - Example: Let the intended boundary be $x + 0.5*y > 5$, whereas the actual boundary is $x + y > 5$.

ON and OFF Points

- Idea
  - Data points on or near a boundary are most **sensitive** to domain errors.
  - Sensitive means a data point falling in the wrong domain.
  - The objective is to identify the data points most sensitive to domain errors so that errors can be detected by examining the program with those input values.
  - Based on the above idea, we define two kinds of data points: ON and OFF.
- ON point
  - It is a point on the boundary or very close to the boundary.
    - If a point can be chosen to lie exactly on the boundary, then choose it. This requires the boundary inequality to have an **exact** solution.
    - If an inequality leads to an **approximate** solution, choose a point very close to the boundary.
- OFF point

ON and OFF Points

- ON point
  - It is a point on the boundary or very close to the boundary.
    - If a point can be chosen to lie exactly on the boundary, then choose it. This requires the boundary inequality to have an **exact** solution.
    - If an inequality leads to an **approximate** solution, choose a point very close to the boundary.
  - Example: Consider the boundary $x + 7*y \geq 6$.
    - For x = -1, the predicate gives us an **exact** solution of y = 1. Therefore the point (-1, 1) lies **on** the boundary.
    - For x = 0, the predicate leads us to an approximate solution y = 0.8571428… . Since y does not have an exact solution, we can truncate it to 0.857 or round it off to 0.858. Notice that (0, 0.857) does not satisfy the predicate, whereas (0, 0.858) does satisfy. Thus, (0, 0.858) is an ON point which lies **very close** to the boundary. And, the on point lies **outside** the domain.

ON and OFF Points

- OFF point
  - An OFF point of a boundary lies **away** from the boundary.
  - While choosing an OFF point, we must consider whether the boundary is open or closed w.r.t. the domain of interest.
    - **Open**: An OFF point of the boundary is an **interior** point inside the domain within an ε-distance from the boundary. ($\varepsilon \equiv$ small)
    - **Closed**: An OFF point of that boundary is an **exterior** point outside the boundary with an ε-distance.
  - Example (Closed): Consider a domain D1 with a boundary $x + 7*y \geq 6$. An OFF point lies **outside** the domain. (-1, 0.99) lies outside D1.
  - Example (Open): Consider a domain D2 that is adjacent to D1 above with an open boundary $x + 7*y < 6$. (-1, 0.99) lies inside D2.

ON and OFF Points

- Summary of ON and OFF points
  - For a **closed** boundary: an **ON** point lies **within the domain** of interest, whereas an **OFF** point lies in an **adjacent domain**.
  - For an **open** boundary, an **ON** point lies in an **adjacent domain**, whereas an **OFF** point lies **within the domain** of interest.

ON and OFF Points

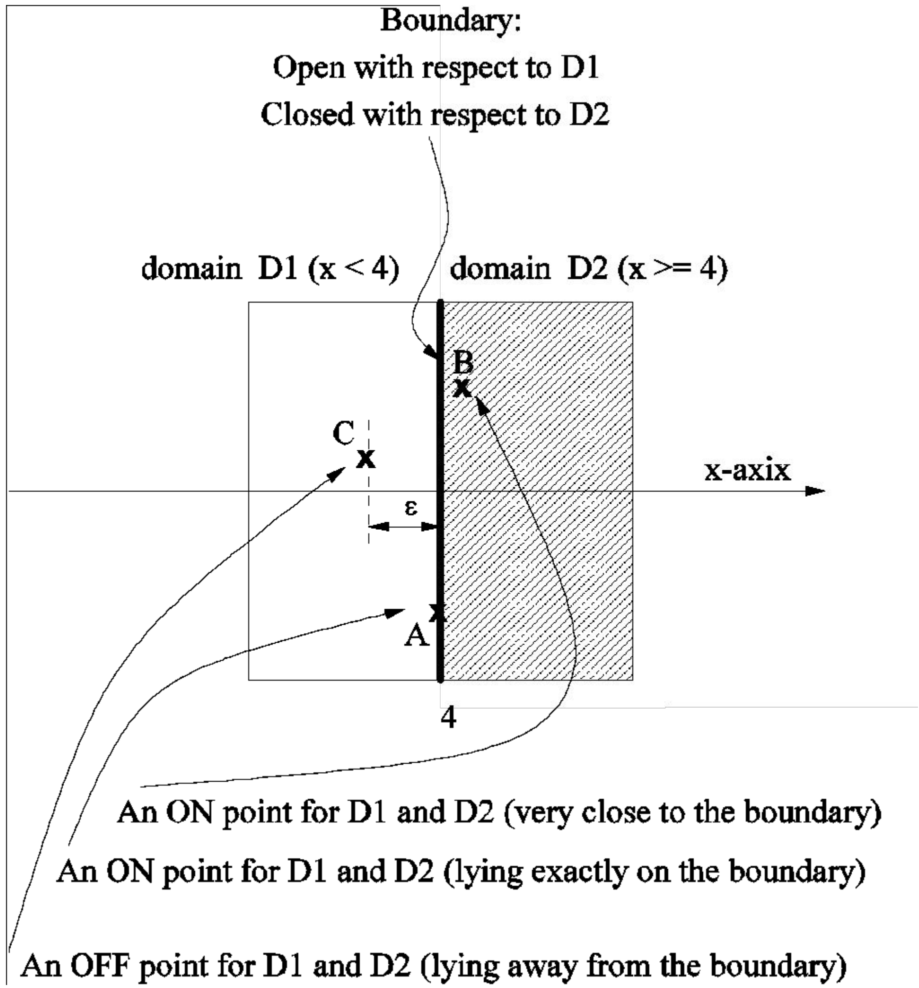

Figure 6.6: ON and OFF points.

Test Selection Criterion

- Selection criterion: For each domain and for each boundary
  - select three points A, C, and B in an ON-OFF-ON sequence.
- We will consider the following kinds of errors.
  - Closed inequality boundary
    - 1.a Boundary shift resulting in a reduced domain
    - 1.b Boundary shift resulting in an enlarged domain
    - 1.c Boundary tilt
    - 1.d Closure error
  - Open inequality boundary
    - 2.a Boundary shift resulting in a reduced domain
    - 2.b Boundary shift resulting in an enlarged domain
    - 2.c Boundary tilt
    - 2.d Closure error
  - Equality boundary

Test Selection Criterion

- Closed inequality boundary
  - 1.a Boundary shift resulting in a reduced domain

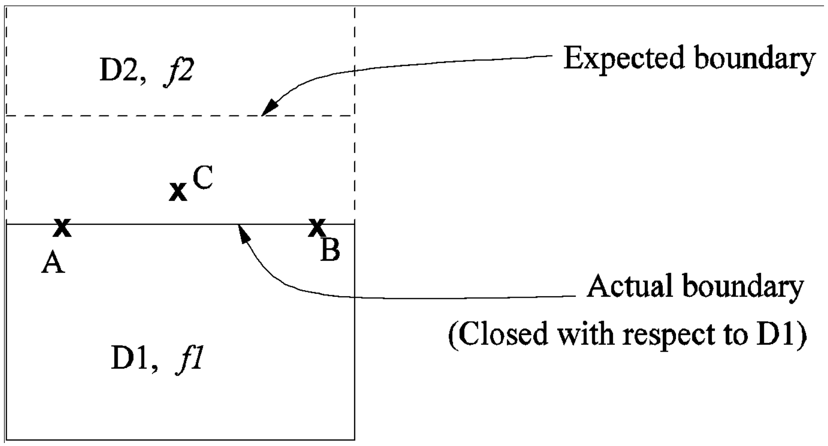

Figure 6.7: Boundary shift resulting in a reduced domain (closed inequality).

Test data	Actual output	Expected output	Fault detected
A	f1(A)	f1(A)	No
B	f1(B)	f1(B)	No
C	f2(C)	f1(C)	Yes

Table 6.2: Detection of boundary shift resulting in a reduced domain (closed inequality).

Test Selection Criterion

- Closed inequality boundary
  - 1.b Boundary shift resulting in an enlarged domain

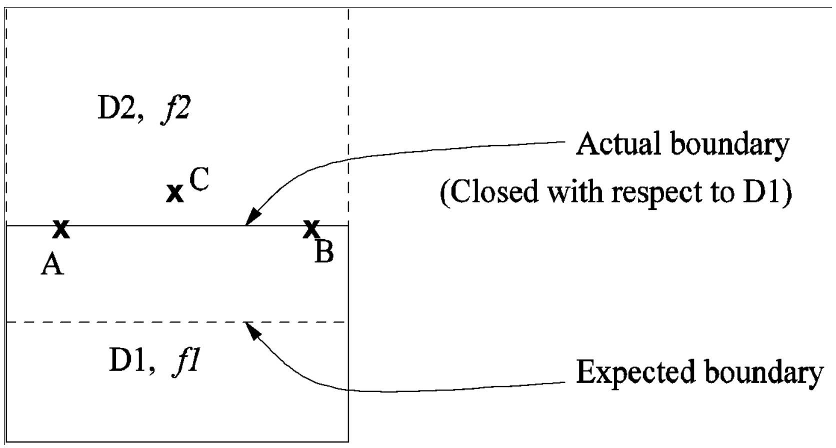

Figure 6.8: Boundary shift resulting in an enlarged domain (closed inequality).

Test data	Actual output	Expected output	Fault detected
A	f1(A)	f2(A)	Yes
B	f1(B)	f2(B)	Yes
C	f2(C)	f2(C)	No

Table 6.3: Detection of boundary shift resulting in an enlarged domain (closed inequality).

Test Selection Criterion

- Closed inequality boundary
  - 1.c Tilted boundary

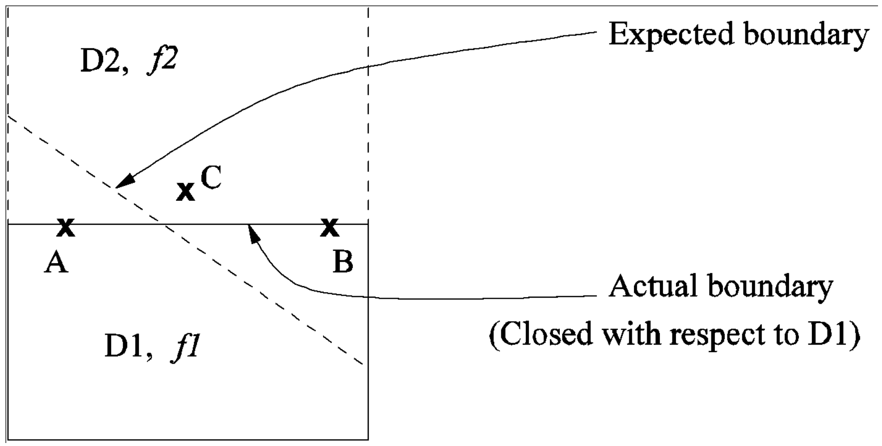

Figure 6.9: Tilted boundary (closed inequality).

Test data	Actual output	Expected output	Fault detected
A	f1(A)	f1(A)	No
B	f1(B)	f2(B)	Yes
C	f2(C)	f2(C)	No

Table 6.4: Detection of tilted boundary (closed inequality).

Test Selection Criterion

- Closed inequality boundary
  - 1.d Closure error

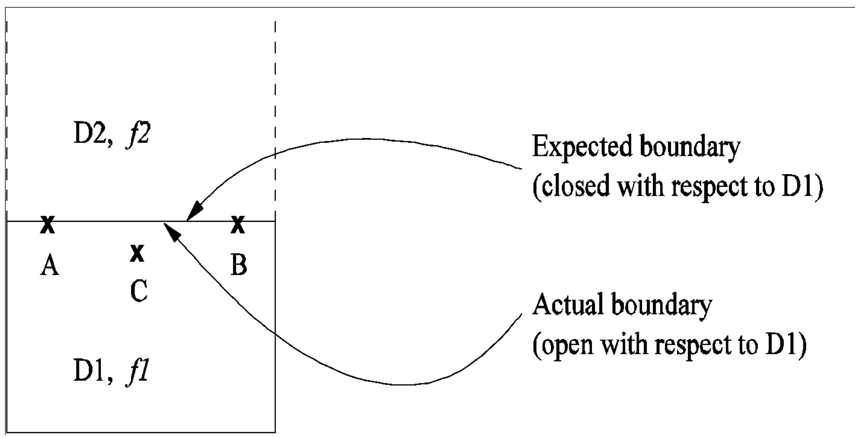

Figure 6.10: Closure error (closed inequality).

Test data	Actual output	Expected output	Fault detected
A	f2(A)	f1(A)	Yes
B	f2(B)	f1(B)	Yes
C	f1(C)	f1(C)	No

Table 6.5: Detection of closure error (closed inequality).

Test Selection Criterion

- Open inequality boundary
  - 2.a Boundary shift resulting in a reduced domain

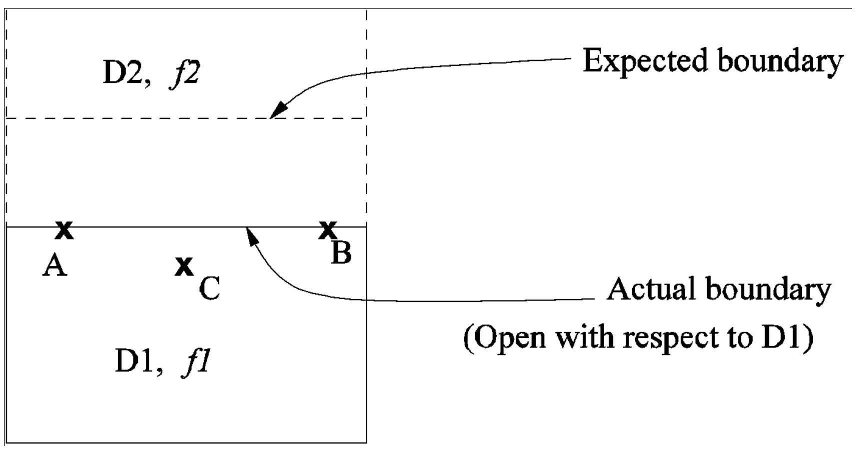

Figure 6.11: Boundary shift resulting in a reduced domain (open inequality).

Test data	Actual output	Expected output	Fault detected
A	f2(A)	f1(A)	Yes
B	f2(B)	f1(B)	Yes
C	f1(C)	f1(C)	No

Table 6.6: Detection of boundary shift resulting in a reduced domain (open inequality).

Test Selection Criterion

- Open inequality boundary
  - 2.b Boundary shift resulting in an enlarged domain

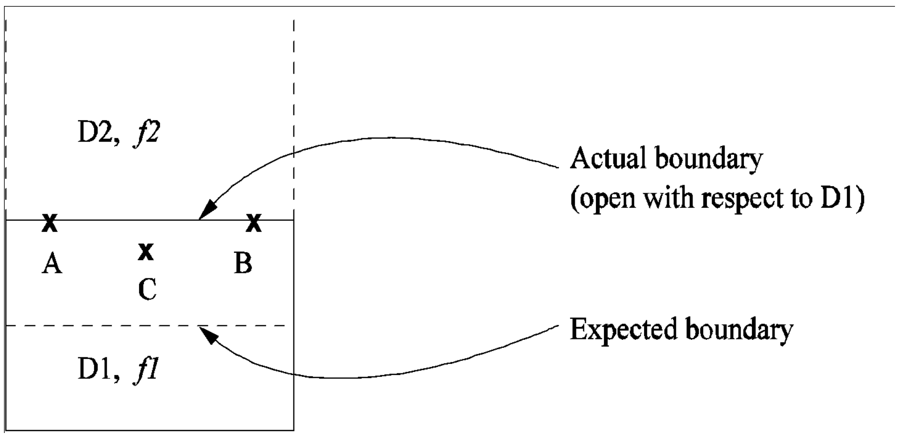

Figure 6.12: Boundary shift resulting in an enlarged domain (open inequality).

Test data	Actual output	Expected output	Fault detected
A	f2(A)	f2(A)	No
B	f2(B)	f2(B)	No
C	f1(C)	f2(C)	Yes

Table 6.7: Detection of boundary shift resulting in an enlarged domain (open inequality).

Test Selection Criterion

- Open inequality boundary
  - 2.c Tilted boundary

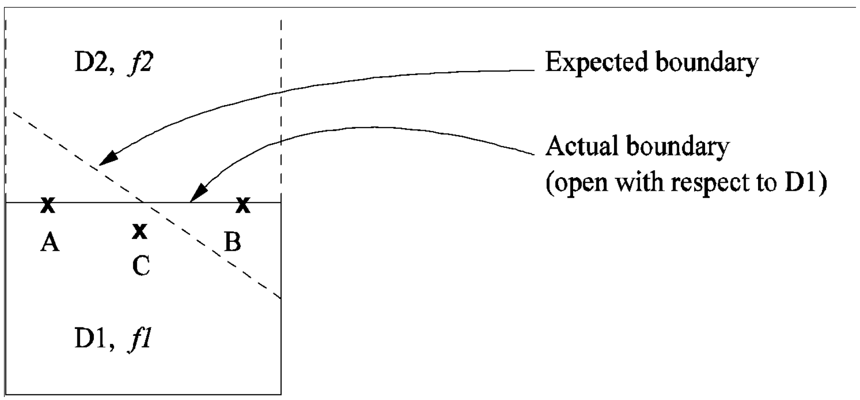

Figure 6.13: Tilted boundary (open inequality).

Test data	Actual output	Expected output	Fault detected
A	f2(A)	f1(A)	Yes
B	f2(B)	f2(B)	No
C	f1(C)	f1(C)	No

Table 6.8: Detection of tilted boundary (open inequality).

Test Selection Criterion

- Open inequality boundary
  - 2.d Closure error

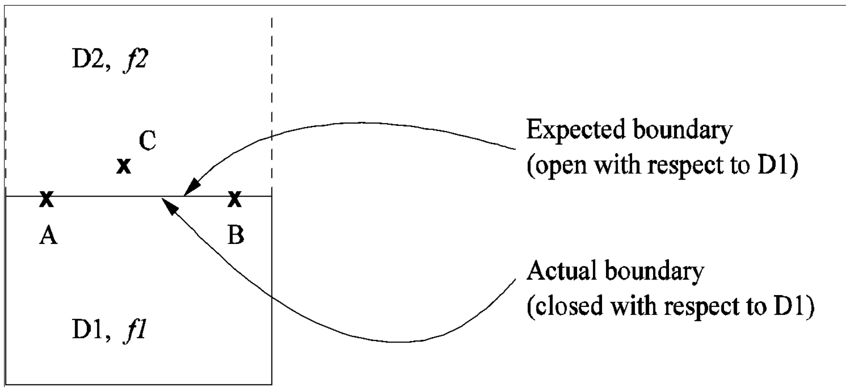

Figure 6.14: Closure error (open inequality).

Test data	Actual output	Expected output	Fault detected
A	f1(A)	f2(A)	Yes
B	f1(B)	f2(B)	Yes
C	f2(C)	f2(C)	No

Table 6.9: Detection of closure error (open inequality).

Test Selection Criterion

- Equality border

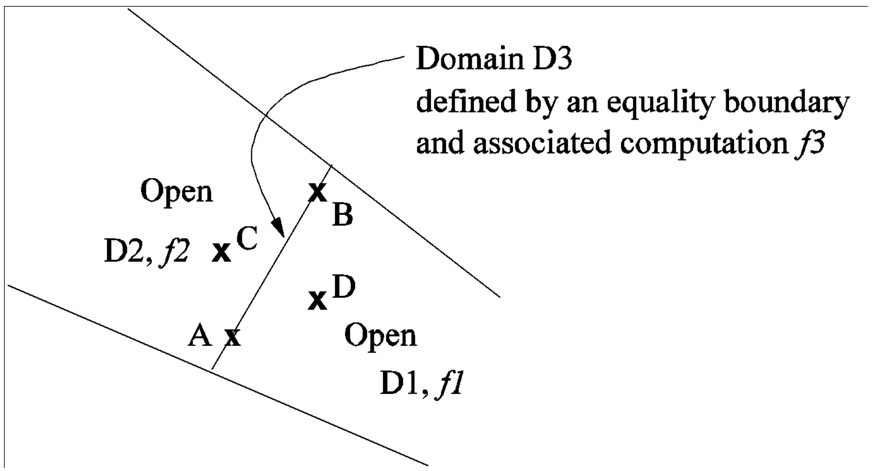

Figure 6.15: Equality border

Summary

- Concepts
  - Two kinds of program error: **computation** errors and **domain** errors
  - Domain: A set of inputs for which the program executes the same path.
  - Domain error: A domain error occurs when an input value causes the program to execute the **wrong** path.
  - A program is viewed as an input classifier. It classifies the input into a set of (sub)domains such that the program executes a different path for each domain.
  - Each domain is identified by a set of boundaries, and each boundary is expressed as a boundary condition.
  - Properties of boundaries: closed, open, closed domain, …
  - Three kinds of boundary errors were identified.
  - closure error, shifted boundary, tilted boundary
  - ON and OFF points
- Test selection criterion: For each domain and for each boundary, select three points A, C, and B in an ON-OFF-ON sequence.