- If the score is between 100 and 200, the player gets 50 bonus points.
- If the total ordering is above $100.00, shipping costs are $5.00.
- …

Let me test it!
I do **“off-by-one”**
mistakes all the time!

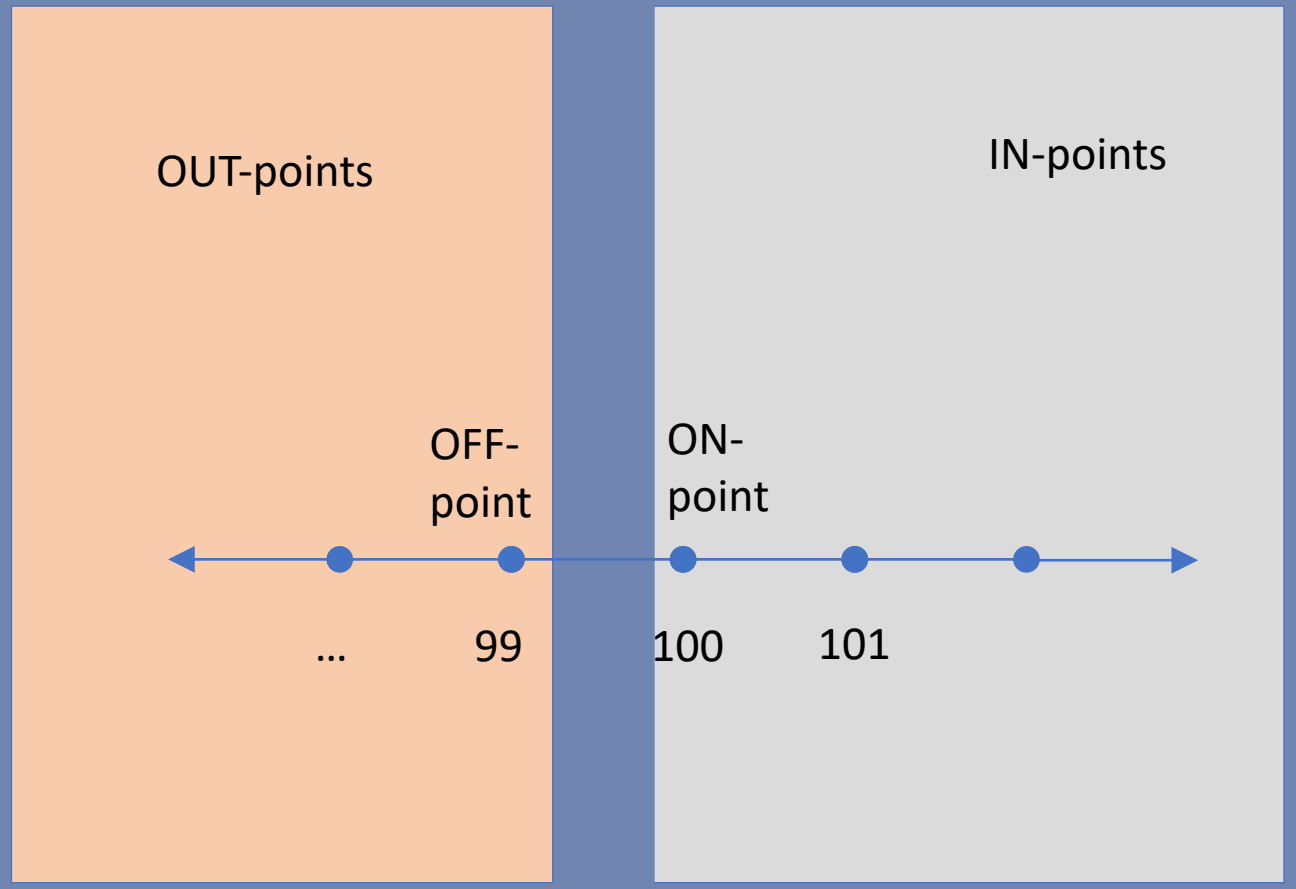
Let me test it

I do **“off-**

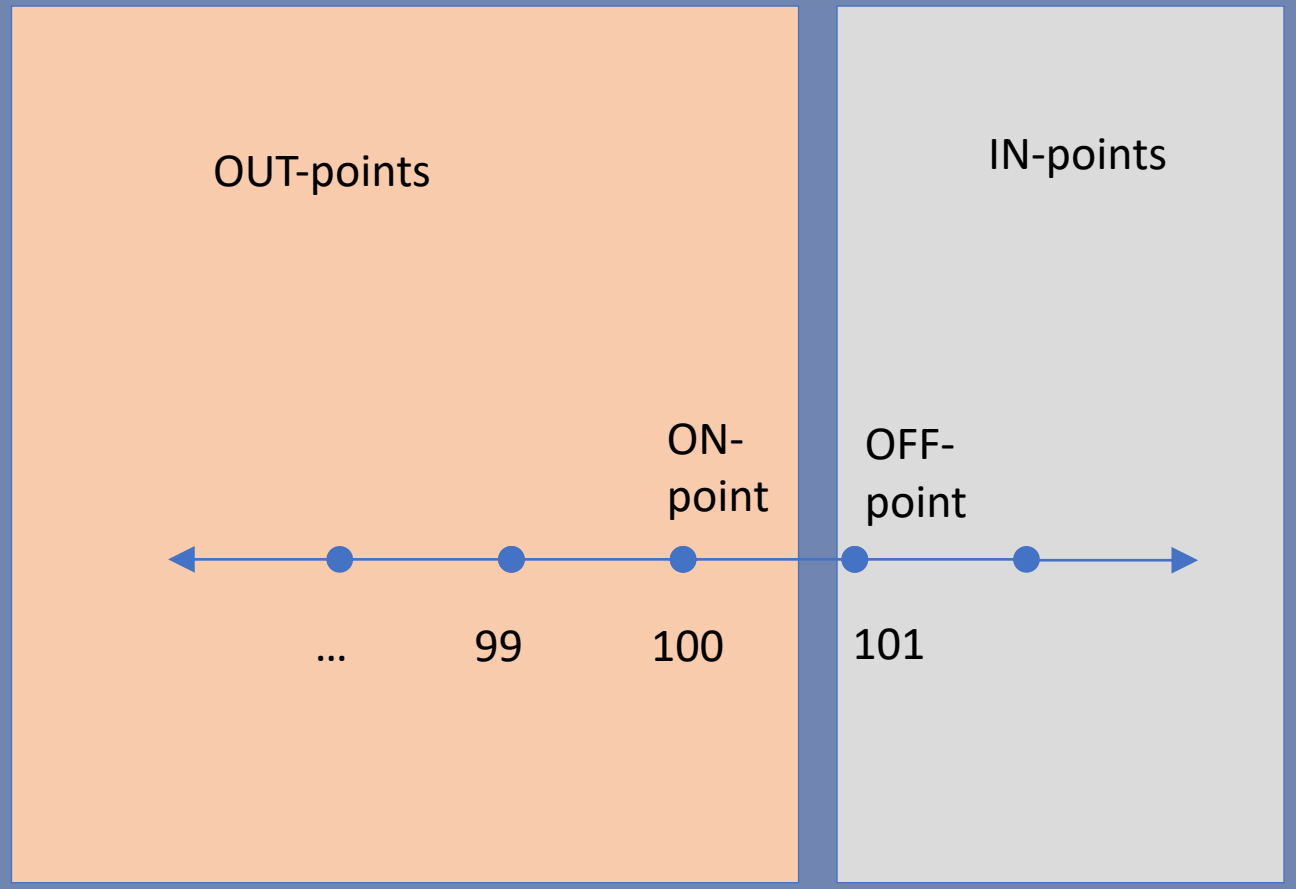
mista

Boundary analysis

X >= 100



X > 100



score >= 100

- **On point:** *Exactly on boundary*
- **In point:** *Makes the condition true*
- **Out point:** *Makes the condition false*
- **Off point:**
  - *Flips the outcome for on point* **and**
  - *Is as close to boundary as possible*

On is 100; In is e.g. 200;
Out is e.g. 50; Off is 99.

Multiple boundaries?

A Simplified Domain-Testing Strategy

BINGCHIANG JENG
Sun Yat-Sen University
and
ELAINE J. WEYUKER
New York University

A simplified form of domain testing is proposed that is substantially cheaper than previously proposed versions, and is applicable to a much larger class of programs. In particular, the traditional restrictions to programs containing only linear predicates and variables defined over continuous domains are removed. In addition, an approach to path selection is proposed to be used in conjunction with the new strategy.

Categories and Subject Descriptors: D.2.5 [**Software Engineering**]: Testing and Debugging

General Terms: Rehability, Theory, Verification

Additional Key Words and Phrases: Domain testing, software testing

1. INTRODUCTION

Domain testing is a fault-based software-testing strategy proposed by White and Cohen [1978]. Testers have frequently observed that subdomain boundaries are particularly fault-prone and should therefore be carefully checked. Domain testing was proposed as a relatively sophisticated form of boundary value testing, and is applicable whenever the input domain is subdivided into subdomains by the program's decision statements.

In this paper, a simplified version of domain testing is described which removes several limitations associated with earlier domain-testing strategies. In particular, our strategy is applicable to arbitrary types of predicates, detects both linear and nonlinear errors, for both discrete and continuous variable spaces. In addition, we will show that our new technique requires much smaller test suites than earlier versions, and will argue that its effectiveness is comparable to, and in some cases superior to, the others.

...and by NASA grant NAG-1-1238

Simplified domain-testing strategy

- Handle boundaries independently
- For each boundary, *pick on and off point*
- While testing one boundary, use varying *in points* for the remaining boundaries.
- Use *domain matrix.*

Boundary conditions for " x > 0 && x <= 10 && y >= 1.0"								
			test cases (x, y)					
Variable	**Condition**	**type**	t1	t2	t3	t4	t5	t6
x	> 0	on						
		off						
	<= 10	on						
		off						
	typical	in						
y	>= 1.0	on						
		off						
	typical	in						

Boundary conditions for ' x > 0 && x <= 10 && y >= 1.0 '								
			test cases (x, y)					
Variable	**Condition**	**type**	t1	t2	t3	t4	t5	t6
x	> 0	on						
		off						
	<= 10	on						
		off						
	typical	in						
y	>= 1.0	on						
		off						
	typical	in						

Boundary conditions for " x > 0 && x <= 10 && y >= 1.0"

			test cases (x, y)					
Variable	**Condition**	**type**	t1	t2	t3	t4	t5	t6
x	> 0	on	0					
		off		1				
	<= 10	on			10			
		off				11		
	typical	in					4	6
y	>= 1.0	on					1.0	
		off						0.9
	typical	in	10.0	16.0	109.3	2390.2		

Boundary conditions for " x > 0 && x <= 10 && y >= 1.0"								
			test cases (x, y)					
Variable	**Condition**	**type**	t1	t2	t3	t4	t5	t6
x	> 0	on	0					
		off		1				
	<= 10	on			10			
		off				11		
	typical	in					4	6
y	>= 1.0	on					1.0	
		off						0.9
	typical	in	10.0	16.0	109.3	2390.2		

Boundary conditions for " x > 0 && x <= 10 && y >= 1.0"								
			test cases (x, y)					
Variable	**Condition**	**type**	t1	t2	t3	t4	t5	t6
x	> 0	on	0					
		off		1				
	<= 10	on			10			
		off				11		
	typical	in					4	6
y	>= 1.0	on					1.0	
		off						0.9
	typical	in	10.0	16.0	109.3	2390.2		

Simplified domain-testing strategy

- Handle boundaries independently
- For each boundary, *pick on and off point*
- While testing one boundary, use varying *in points* for the remaining boundaries.
- Use *domain matrix.*