UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Critério de Teste: Classes de Equivalência (PARTE 1)

Profa. Roberta Coelho

Critérios de Escolha

- Já que não podemos testar um sistema usando **TODAS ENTRADAS POSSíVEIS**...

- Precisamos definir um critério de escolha.
  - **Quais dados de entrada irei utilizar para testar o sistema??**

Software Unit Test Coverage and Adequacy

HONG ZHU

Nanjing University

PATRICK A. V. HALL AND JOHN H. R. MAY

The Open University, Milton Keynes, UK

Objective measurement of test quality is one of the key issues in software testing. It has been a major research focus for the last two decades. Many test criteria have been proposed and studied for this purpose. Various kinds of rationales have been presented in support of one criterion or another. We survey the research work in this area. The notion of adequacy criteria is examined together with its role in software dynamic testing. A review of criteria classification is followed by a summary of the methods for comparison and assessment of criteria.

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

General Terms: Measurement, Performance, Reliability, Verification

Additional Key Words and Phrases: Comparing testing effectiveness, fault-detection, software unit test, test adequacy criteria, test coverage, testing methods

1. INTRODUCTION

In 1972, Dijkstra claimed that "program testing can be used to show the presence of bugs, but never their absence" to persuade us that a testing approach is not acceptable [Dijkstra 1972]. However, the last two decades have seen rapid growth of research in software testing as well as intensive practice and experiments. It has been developed into a validation and verification technique indispensable to software engineering discipline. Then, where are we today? What can we claim about software testing?

In the mid-'70s, in an examination of the capability of testing for demonstrating the absence of errors in a program, Goodenough and Gerhart [1975, 1977] made an early breakthrough in research on software testing by pointing out that the central question of software testing is "what is a test criterion?", that is, the criterion that defines what constitutes an adequate test. Since then, test criteria have been a major research focus. A great number of such criteria have been proposed and investigated. Considerable research effort has attempted to provide support for the use of one criterion or another. How should we understand these different criteria? What are the future directions for the subject?

In contrast to the constant attention given to test adequacy criteria by aca-

Exemplos de Critérios

- ...

Exemplos de Critérios

- Posso escolher os valores a partir da minha experiência.

- “error-guessing”

Exemplos de Critérios

- Posso escolher os valores usados com mais freqüência.

- Posso escolher valores muito grandes ou muito pequenos.

Exemplos de Critérios

- Posso também escolher um conjunto de valores que exercitem todas as chamadas de método.

- Ou mesmo todas as linhas de código do programa.

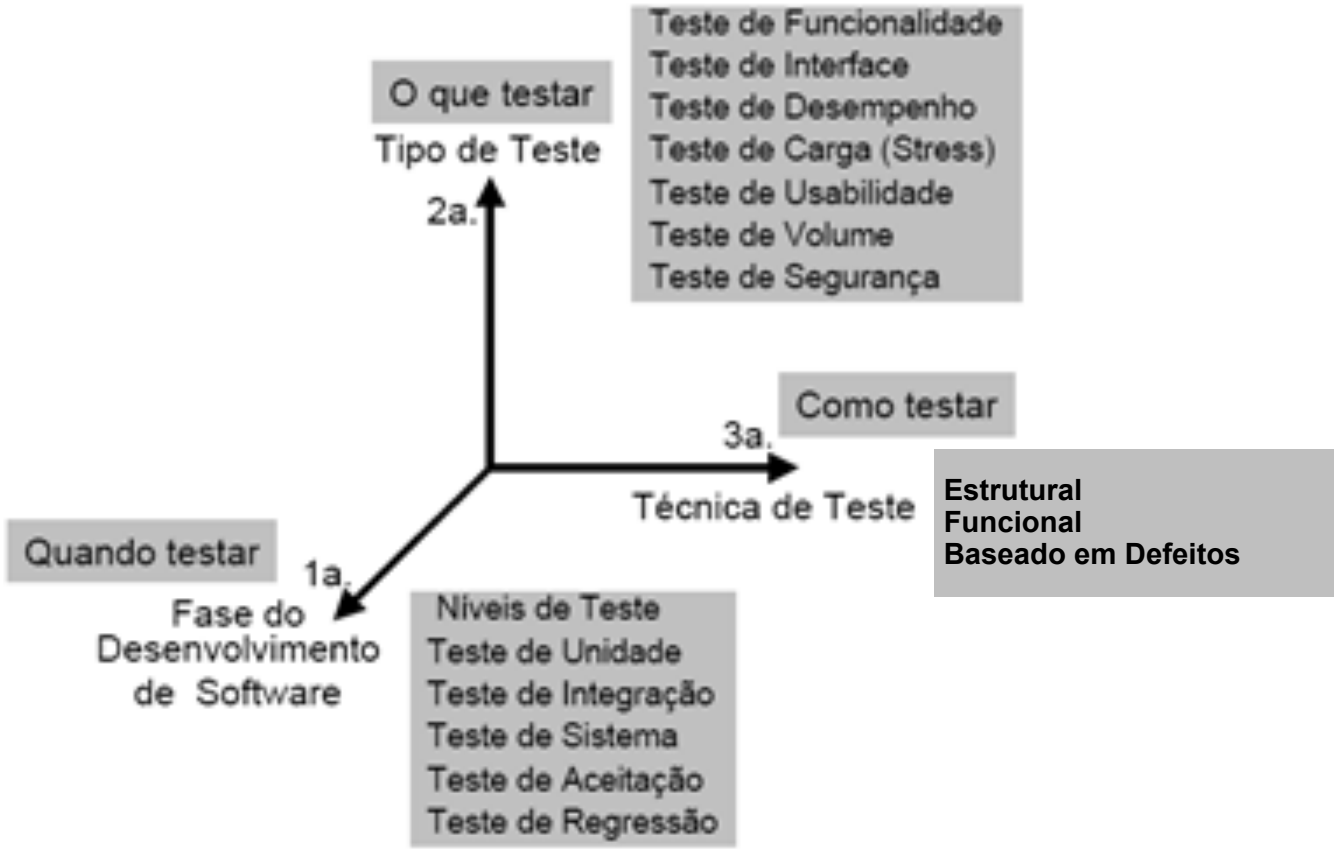
Critérios de Teste

- Estes critérios podem ser usados como critério de parada.

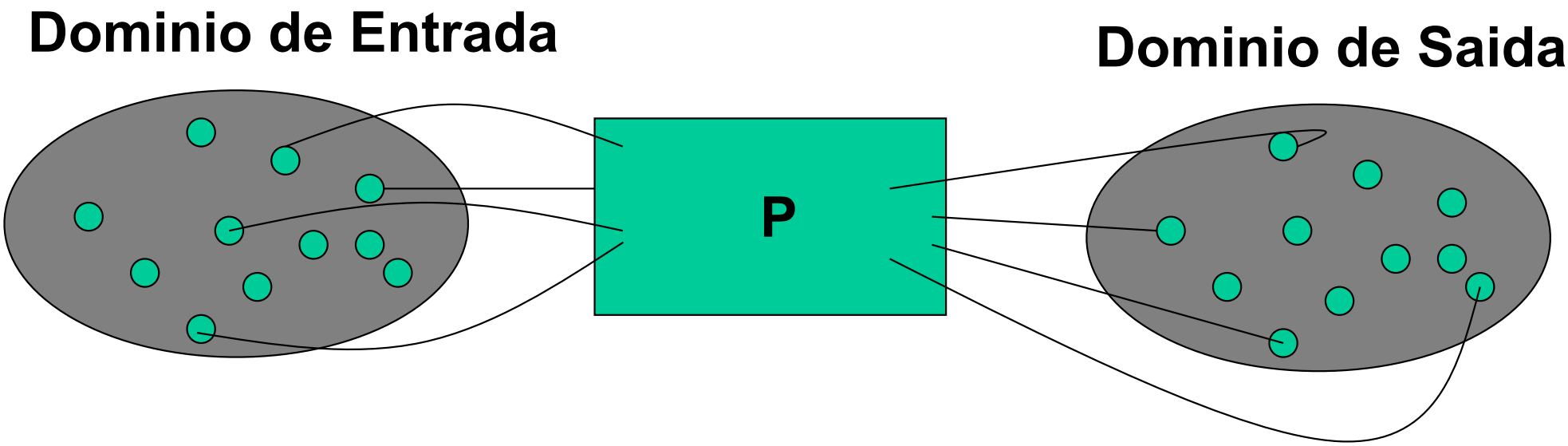
- Quando parar de testar?

  - Ex: Quando todos os requisitos de testes de um criterio forem atendidos pelos testes.

Mapa Mental

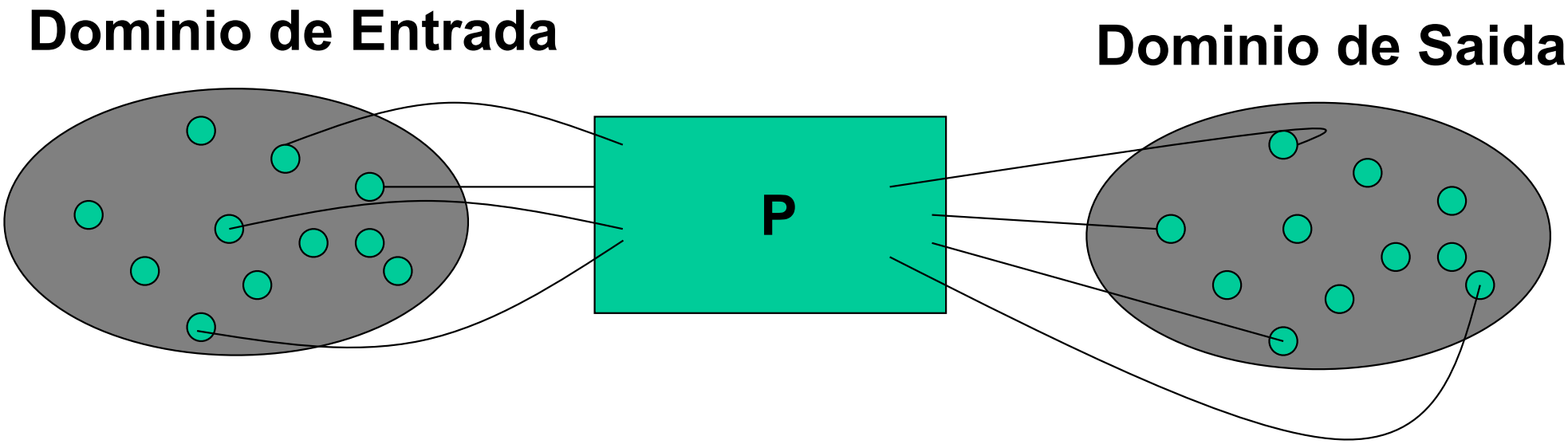

O que testar
Tipo de Teste
2a.
Teste de Funcionalidade
Teste de Interface
Teste de Desempenho
Teste de Carga (Stress)
Teste de Usabilidade
Teste de Volume
Teste de Segurança
Como testar
3a.
Técnica de Teste
Estrutural
Funcional
Baseado em Defeitos
Quando testar
1a.
Fase do Desenvolvimento de Software
Níveis de Teste
Teste de Unidade
Teste de Integração
Teste de Sistema
Teste de Aceitação
Teste de Regressão

O Problema



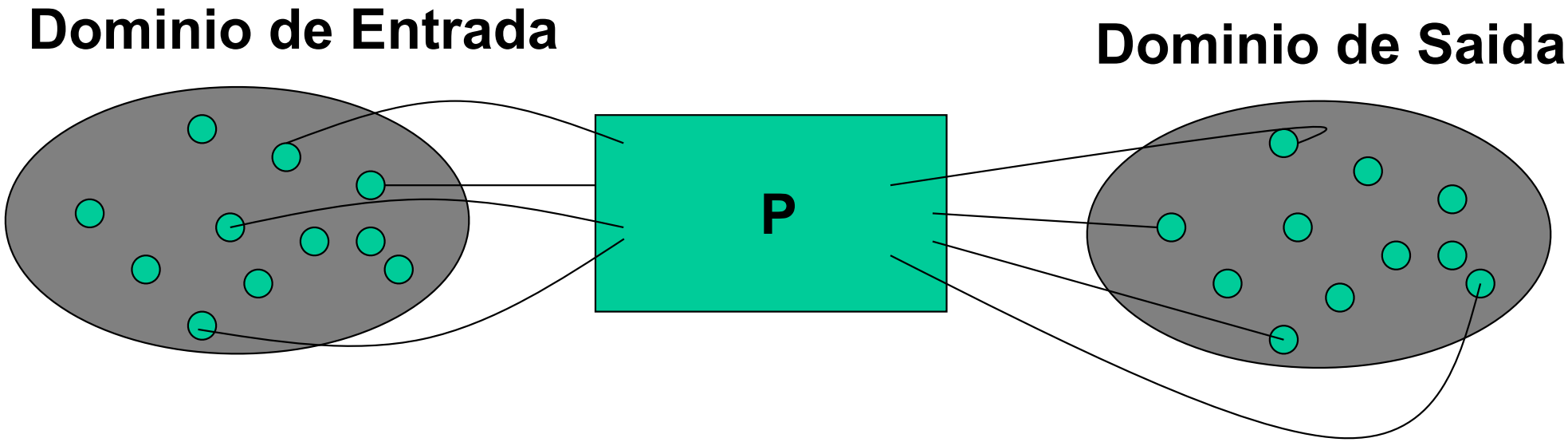
- Idealmente o programa em testes deveria ser executado com todos os elementos do domínio.
- Assim iríamos garantir que esta correto.
- Mas isto é quase sempre inviável…

O Problema



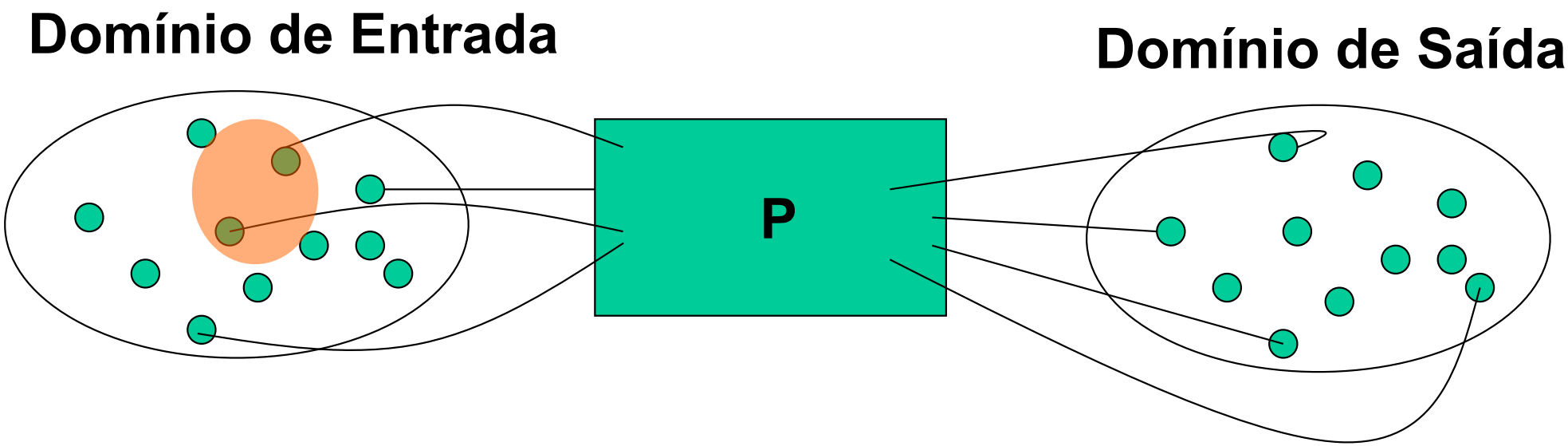
- Testar todas as entradas é quase sempre inviável...

Possíveis Soluções

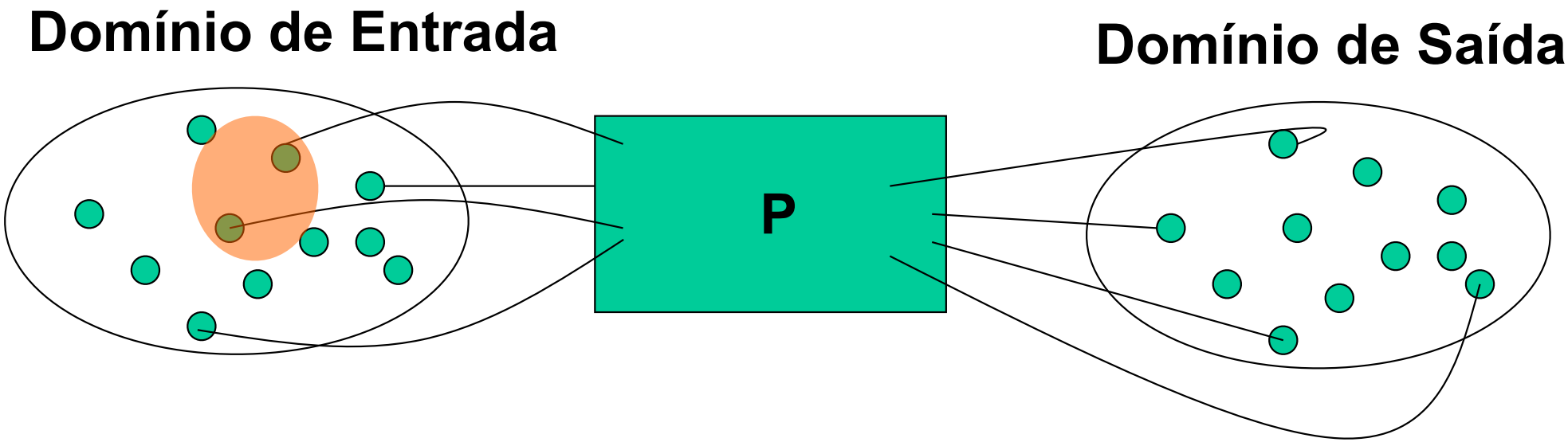


- Assim é importante encontrar formas de utilizar apenas um **subconjunto** de **D(P).**

Mas como encontrar esse subconjunto?

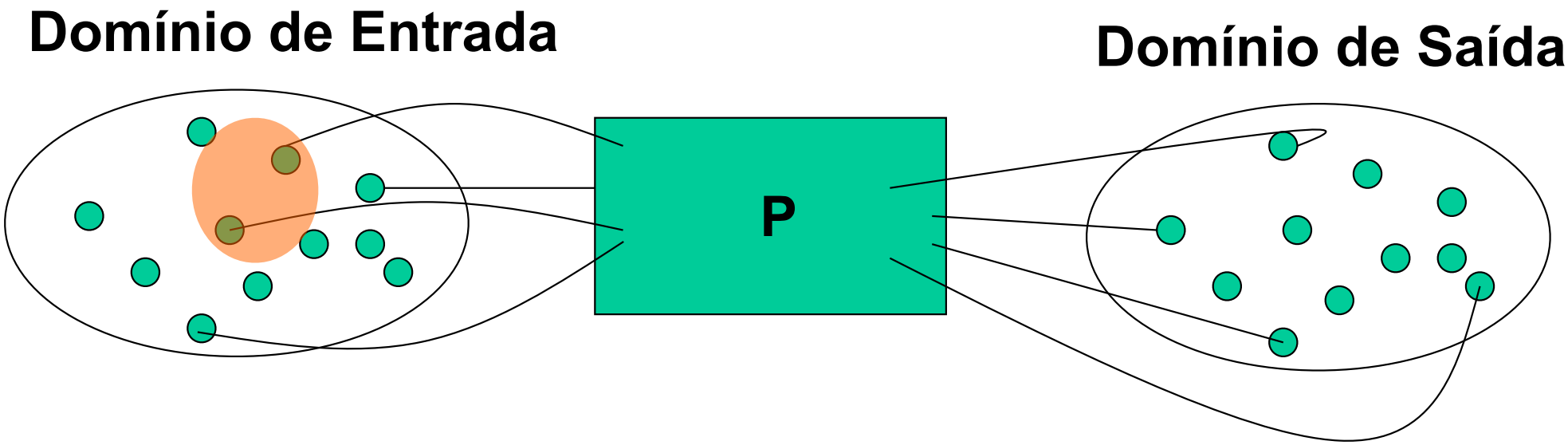


Random Testing



- Estrategia 1: Usar valores quaisquer - aleatorios (*Random Testing*, ou *Monkey Testing*)

Problema do *Random Testing*



- Estudos tem mostrado que esta é a forma menos efetiva.
- Não é sistemática!!!!

O Particionamento em Classes de Equivalência.

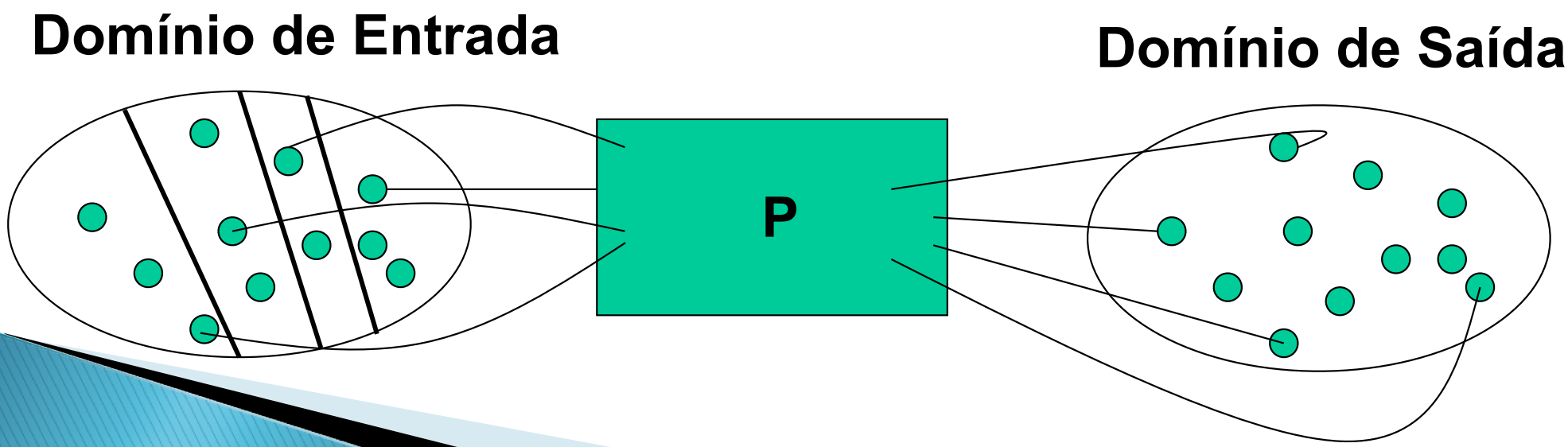
Uma das formas sistemáticas selecionar dados de teste...

O Particionamento do Domínio de Entrada.

Uma das formas sistemáticas selecionar dados de teste...

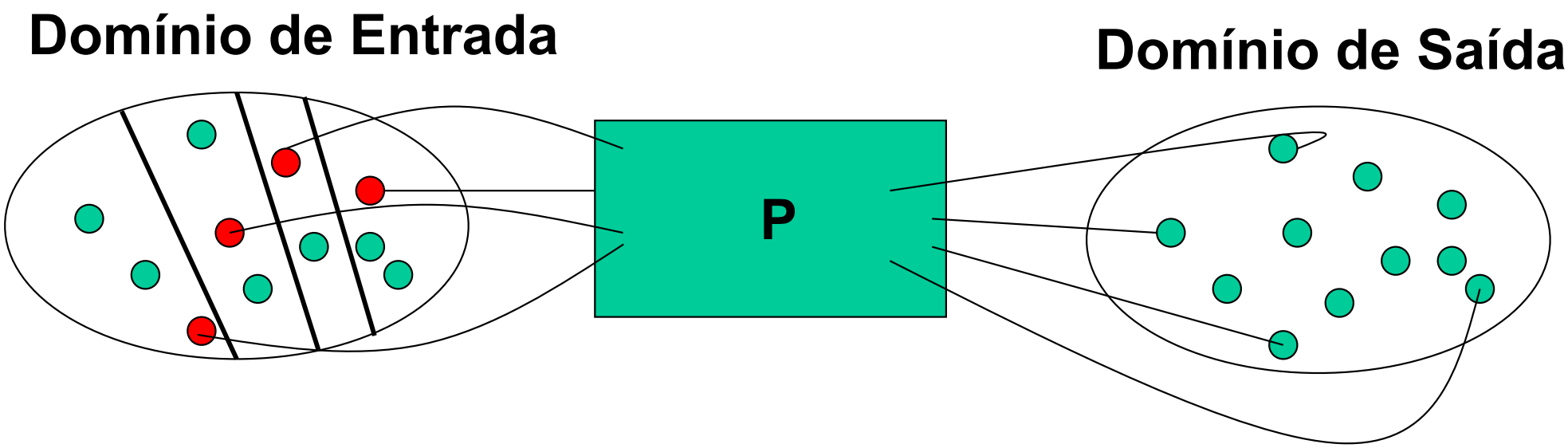
Particionamento em Classes de Equivalência

- Tomando como base informações a respeito da entrada e da saída do programa.
  - Definimos partições para o domínio de entrada.



Conseqüência

- **Durante os testes**
  - selecionamos um ou mais dados de cada partição.

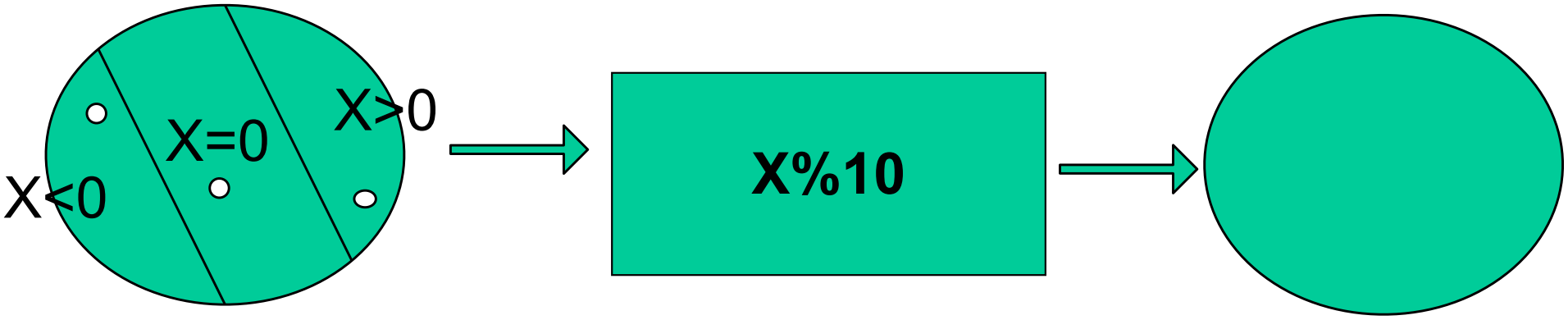


- **Critério de parada:** cada partição deve ser considerada ao menos 1 vez.

Visão Geral

- Esta técnica particiona o domínio de entrada segundo alguma propriedade.

- E seleciona representantes de cada região.



q
(todos
inteiros)

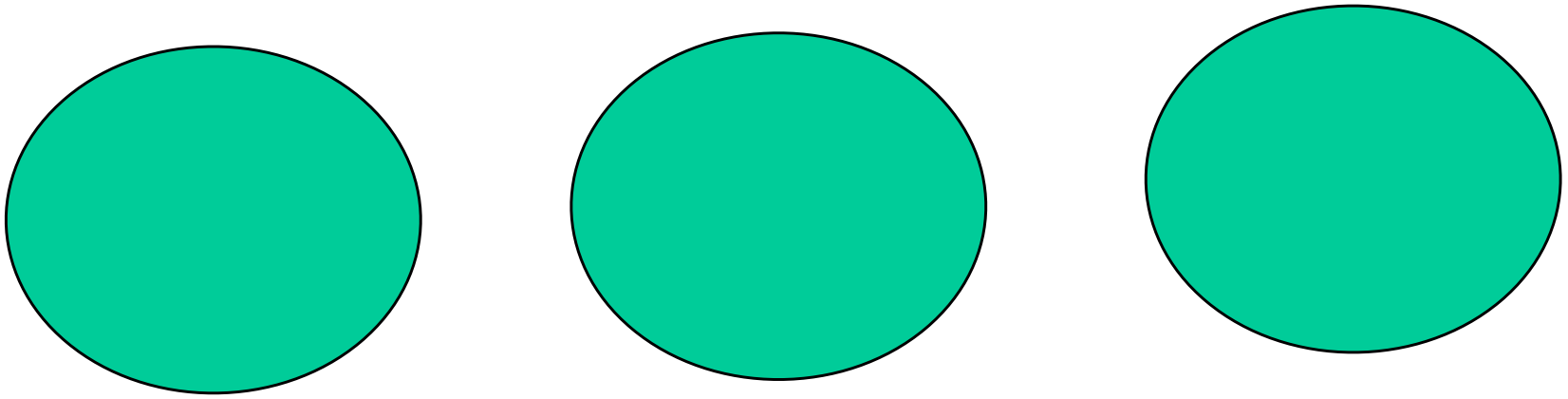
QUIZ

Considere o:

public String tipoTriangulo(int ladoA, int ladoB, int ladoC)

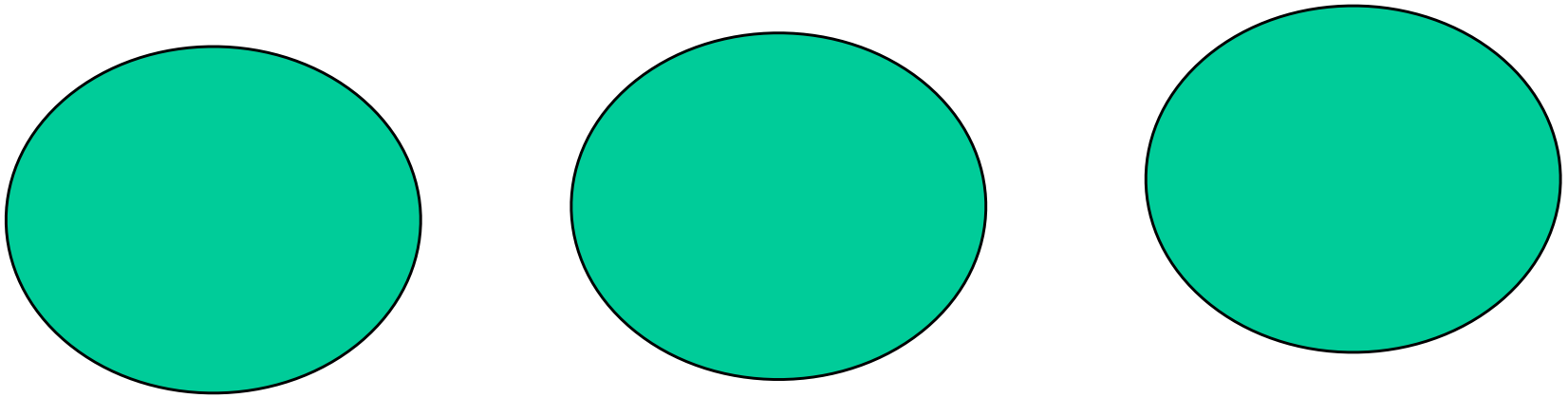
QUIZ

Considere o tipoTriangulo(int ladoA, int ladoB, int ladoC)



QUIZ

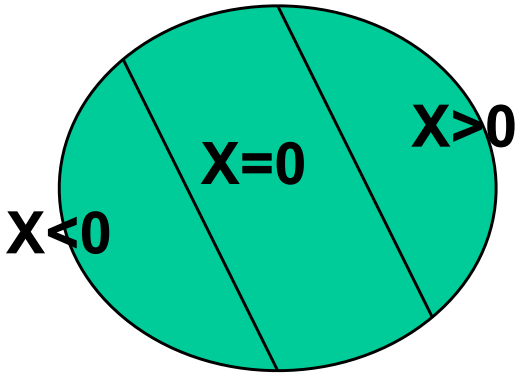
Considere o tipoTriangulo(int ladoA, int ladoB, int ladoC)



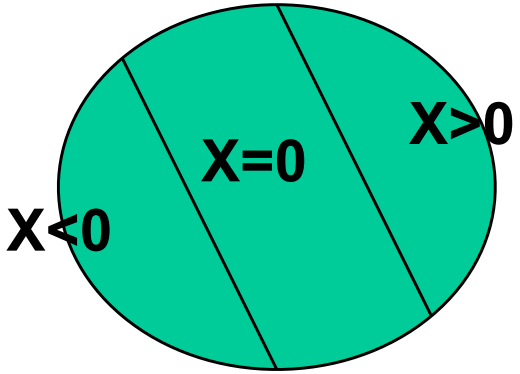
Domínio de entrada para cada parâmetro é idêntico

Característica: *Relação do lado com valor zero*

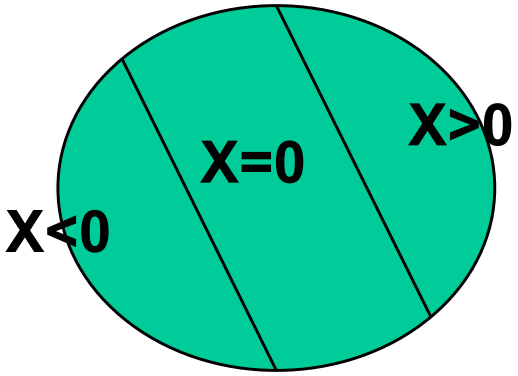
QUIZ - RESPOSTA



Dominio para ladoA
(todos inteiros)



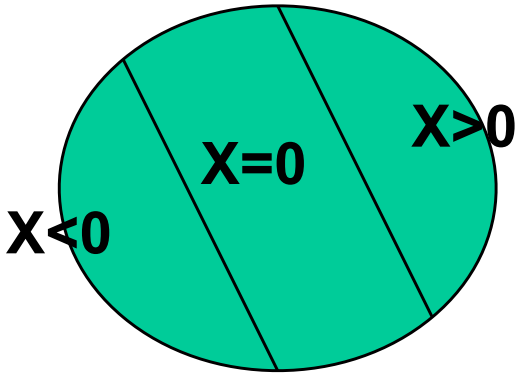
Dominio para ladoB
(todos inteiros)



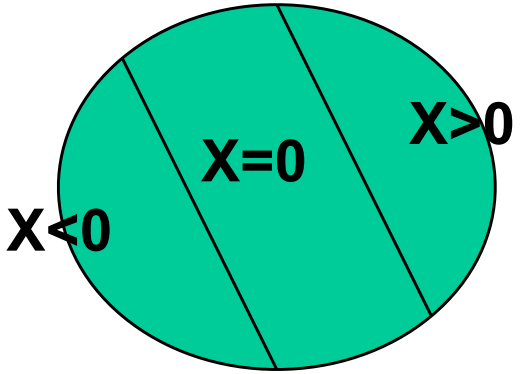
Dominio para ladoC
(todos inteiros)

Como construimos os casos de testes depois de particionar?

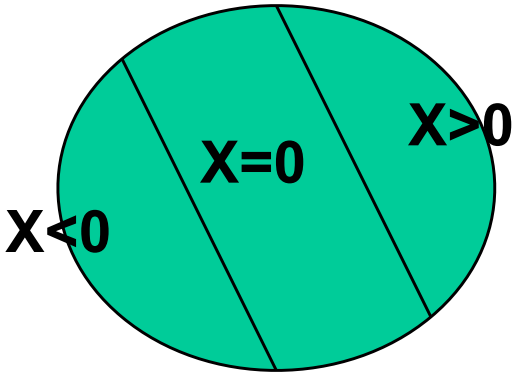
QUIZ - RESPOSTA



Dominio para ladoA (todos inteiros)



Dominio para ladoB (todos inteiros)



Dominio para ladoC (todos inteiros)

T1 (1,1,1),
T2 (0,2,2),
T 3 (-1,-1,-1), ...

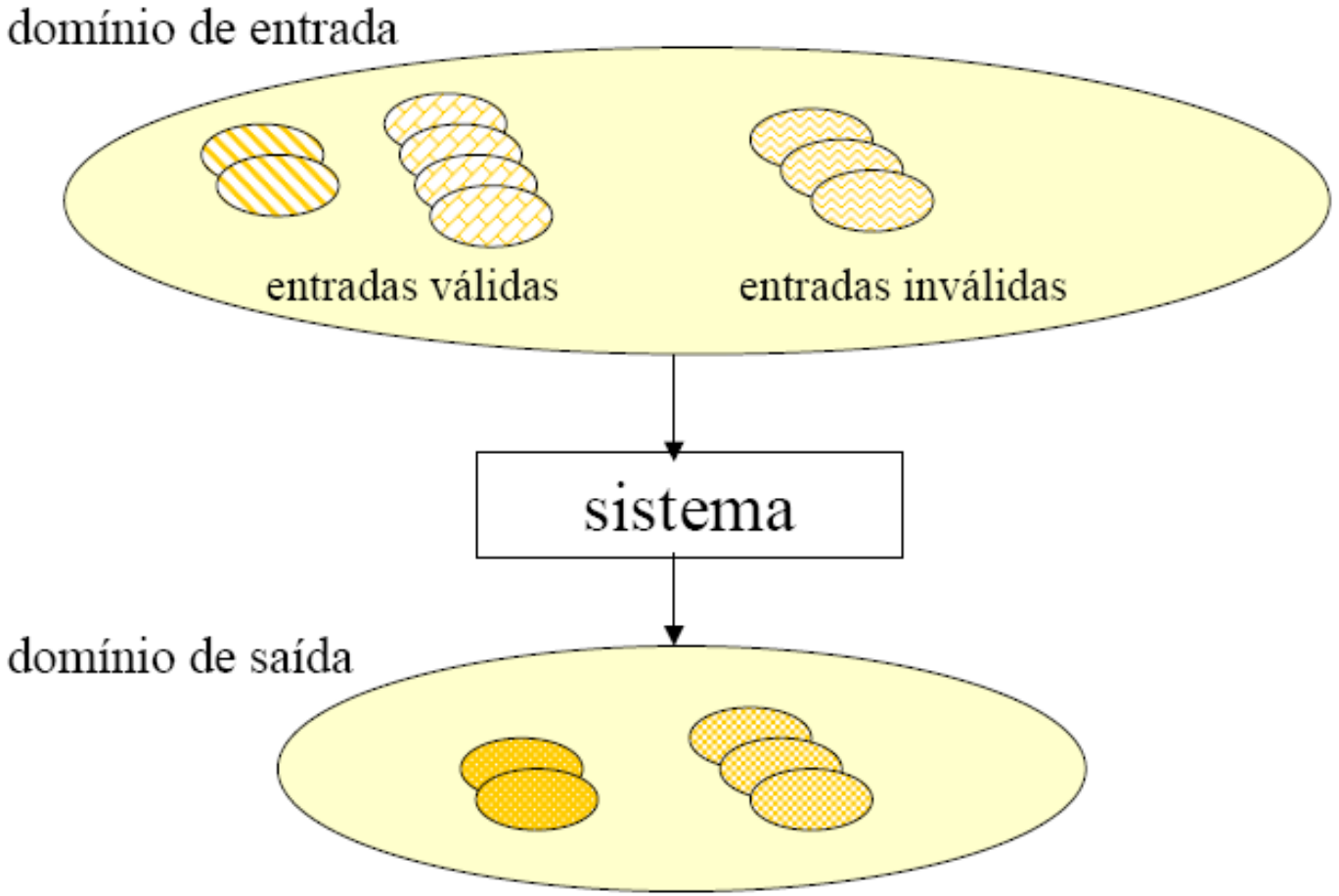
Suposição que fundamenta este critério

- Supõe-se que dados pertencentes a uma partição têm capacidade de revelar as mesmas falhas

Particionamento em Classes de Equivalência

- Essas partições são classificadas em:
  - **classes válidas:** valores válidos do domínio de entrada
  - **classes inválidas:** valores inválidos do domínio de entrada.

Resumo - Particionamento em Classes de Equivalência



Passo a Passo: Entrada

Passo 1: Entender o comportamento da função e identificar as variáveis que determinam o comportamento de cada função

Passo 2: Particionar os valores de cada variável em classes de equivalência (válidas e inválidas)

Passo 3: Construir os casos de teste selecionando elementos de cada classe.

Passo a Passo: Possíveis Saídas

Passo 1: Entender o comportamento da função e identificar as possíveis saídas de cada função

Passo 2: Particionar os valores de saída em classes de equivalência (válidas e inválidas)

Passo 3: Construir os casos de teste para cobrir elementos de cada classe.

Dicas

Vejamos algumas dicas que nos ajudam a encontrar as partições...

Dicas 1 e 2

Definição da variável de entrada	Classes de equivalência
Intervalo	• Uma classe válida para valores pertencentes ao intervalo • Uma classe inválida para valores menores que o limite inferior • Uma classe inválida para valores maiores que o limite superior
Lista de valores válidos	• Uma classe válida para os valores incluídos na lista • Uma classe inválida para todos os outros valores

Dicas 3 e 4

Número de valores válidos	• Uma classe válida para número de valores igual ao número previsto • Uma classe inválida para número de valores = 0 • Uma classe inválida para número de valores maior ou menor que o valor previsto
Restrições (expressão lógica; sintaxe; valor específico; compatibilidade com outras variáveis)	• Uma classe válida para os valores que satisfazem às restrições • Uma classe inválida para os outros valores

Quiz

- Considere uma função:

```
public double calculaMensalidadeEspecial
                                    ( int num_entradas,
                                      int valor)
```

num_entradas [4, 6]

valor [10, 99]

Passo a Passo

Passo 1: Decompor o programa em funções

Passo 2: Identificar as variáveis que determinam o comportamento de cada função

Passo 3: Particionar os valores de cada variável em classes de equivalência (válidas e inválidas)

Passo 4: Especificar os casos de teste.

Definindo as Classes de Equivalência

Variável	Classes Válidas	Classes Inválidas
nro_entradas	C1. $4 \leq$ nro_entradas ≤ 6	C3. nro_entradas < 4
		C4. nro_entradas > 6
valor	C2. $10 \leq$ valor ≤ 99	C5. valor < 10
		C6. valor > 99

Passo 4: Especificar os casos de teste.

- Como especificar os casos de teste?

Passo 4: Especificar os casos de teste.

- Como especificar os casos de teste?

1. selecionar casos de testes cobrindo as classes válidas das diferentes variáveis

2. para cada classe inválida escolha um caso de teste que cubra 1 e somente 1 de cada vez

Passo 4: Especificar os casos de teste.

nro_entradas	C1	C2	C3	C4	C5	C6
4 ... 6	✓					
< 4						
> 6						
valor						
10 ... 99		✓				
< 10						
> 99						

nro_entradas	valores
5	11, 12, 45, 78, 95

Passo 4: Especificar os casos de teste.

nro_entradas	C1	C2	C3	C4	C5	C6
4 ... 6	✓					
< 4			✓			
> 6						
valor						
10 ... 99		✓				
< 10						
> 99						

	nro_entradas	valores
1.	5	11, 12, 45, 78, 95
2.	3	11, 12, 45

Passo 4: Especificar os casos de teste.

nro_entradas	C1	C2	C3	C4	C5	C6
4 ... 6	✓					
< 4			✓			
> 6				✓		
valor						
10 ... 99		✓				
< 10					✓	
> 99						✓

	nro_entradas	valores
1.	5	11, 12, 45, 78, 95
2.	2	11, 12
3.	8	11, 12, 45, 78, 95, 67, 77, 54
4.	5	5, 11, 12, 45, 6
5.	5	110, 45, 78, 34, 95

Resumo

- Como especificar os casos de teste?

1. selecionar casos de testes cobrindo as classes válidas das diferentes variáveis

2. para cada classe inválida escolha um caso de teste que cubra 1 e somente 1 de cada vez