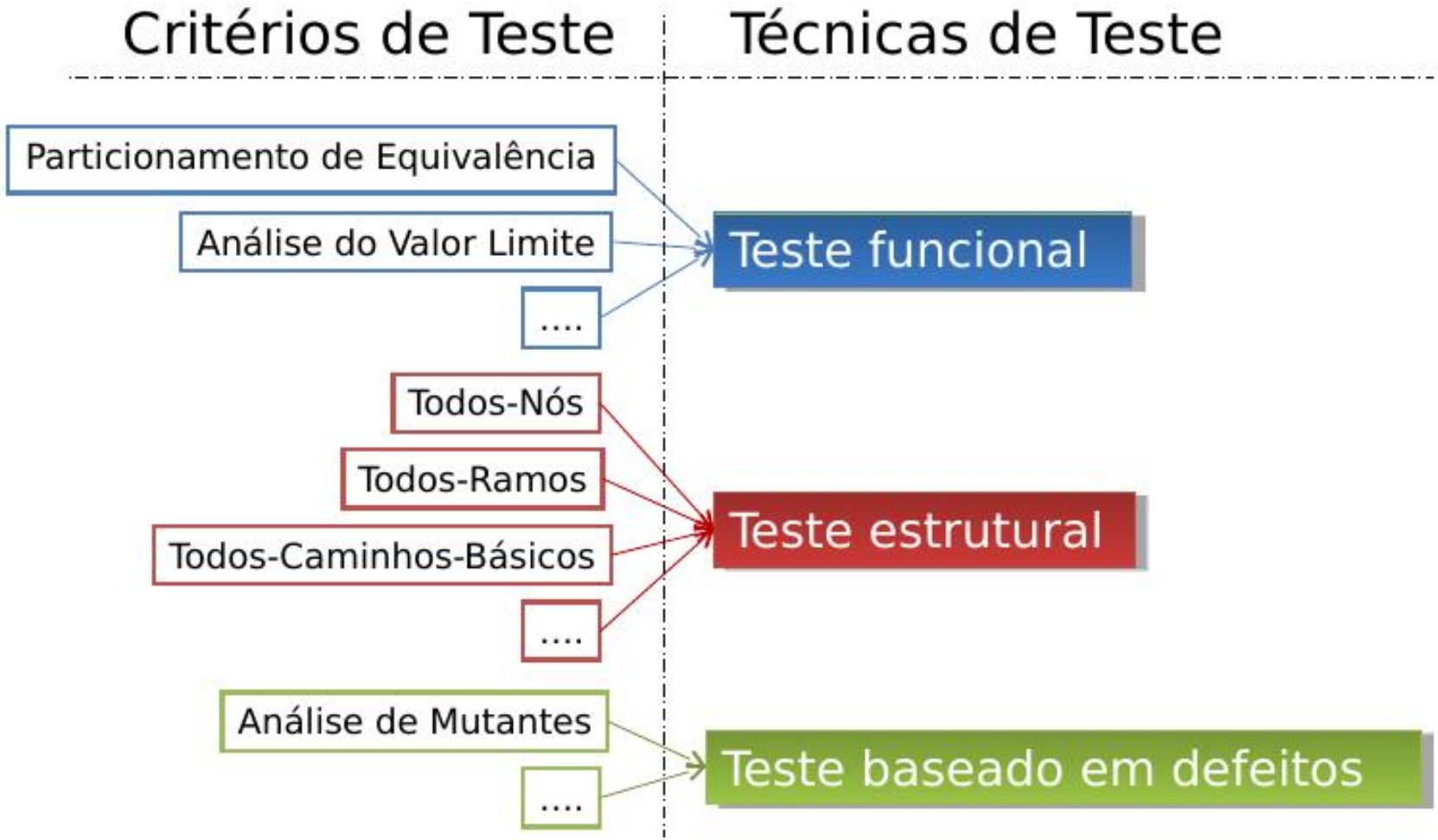
Teste Estrutural

Prof. Otávio Lemos (UNIFESP)
Prof. Fabiano Ferrari (UFSCar)

Introdução

- Técnicas e Critérios de teste $\rightarrow$ abordagem sistemática e teoricamente fundamentada para a condução da atividade de teste
- Além disso, auxiliam para a garantia da qualidade dos CTs
  $\rightarrow$ maior probabilidade em revelar defeitos
- Técnicas se diferenciam pela origem das informações para estabelecer requisitos de teste

São complementares, pois identificam tipos diferentes de defeitos.


Critérios de Teste
Técnicas de Teste
Particionamento de Equivalência
Análise do Valor Limite
....
Teste funcional
Todos-Nós
Todos-Ramos
Todos-Caminhos-Básicos
....
Teste estrutural
Análise de Mutantes
....
Teste baseado em defeitos

- Técnica estrutural – requisitos de teste com base em uma implementação
- Conhecido com **teste caixa branca**
- Requer execução de partes ou componentes elementares do programa
- Caminhos lógicos são testados $\rightarrow$ verificam conjuntos de condições, laços e pares definição-uso de variáveis

- Limitações do teste de software que afetam o teste estrutural:
  - não existe um procedimento de teste de propósito geral para provar a correção de um programa;
  - dados dois programas, é indecidível se eles computam a mesma função;
  - é indecidível, em geral, se dois caminhos de um programa, ou de programas diferentes, computam a mesma função; e
  - é indecidível, em geral, se um dado caminho é executável $\rightarrow$ se existe um conjunto de dados de entrada que leve à execução do caminho.

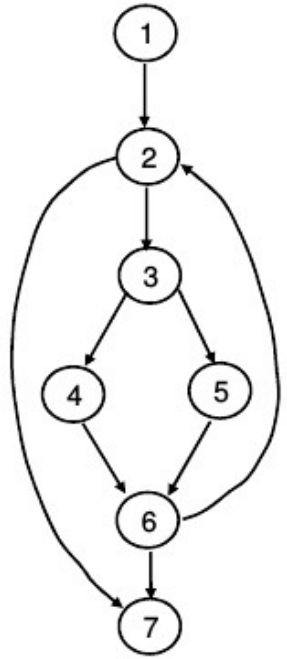
- Outras limitações inerentes do teste estrutural:
  - caminhos ausentes: quando o programa não implementa algumas condições (ou funcionalidades) – caminho não existe; e
  - correção coincidente: o programa pode apresentar um resultado correto para um dado em particular de entrada, satisfazendo um requisito de teste e não revelando a presença de um defeito.

- Independentemente dessas desvantagens, o teste estrutural complementa as demais técnicas – classes distintas de defeitos
- Conceitos e experiência com a técnica podem ser abstraídos e aplicados às outras técnicas
- Além disso, informações obtidas têm sido consideradas relevantes para manutenção, depuração e avaliação da confiabilidade

Exemplo

Pseudo-código

```
início                                          } 1
        leia nro
        enquanto nro ≠ 0                        } 2
            se nro > 0                          } 3
                    raiz = raiz-quadrada(nro)   } 4
                    escreva raiz
            senão
                    escreva mensagem de erro    } 5
            fim-se
            leia nro                            } 6
        fim-enqto
fim                                             } 7
```



Outro exemplo

Programa Identifier (escrito em C)

```
        main () {
/*01*/    char  achar; int length, valid_id; length = 0; valid_id = 1;
/*01*/    printf ("Identificador: ");
/*01*/    achar = fgetc (stdin);
/*01*/    valid_id = valid_s(achar);
/*01*/    if(valid_id)
/*02*/      length = 1;
/*03*/    achar = fgetc (stdin);
/*04*/    while(achar != '\n') {
/*05*/      if(!(valid_f(achar)))
/*06*/         valid_id = 0;
/*07*/      length++;
/*07*/      achar = fgetc (stdin);
/*07*/    }
/*08*/    if(valid_id && (length >= 1) && (length <= 6))
/*09*/      printf ("Valido\n");
/*10*/    else printf ("Inválido\n");
/*11*/ } // fim main

        int valid_s(char ch) {
/*01*/    if(((ch >= 'A') && (ch <= 'Z')) || ((ch >= 'a') && (ch <= 'z')))
/*02*/       return (1);
/*03*/    else return (0);
/*04*/ } // fim valid_s

        int valid_f(char ch) {
/*01*/    if(((ch >= 'A') && (ch <= 'Z')) || ((ch >= 'a') && (ch <= 'z')) ||
/*01*/       ((ch >= '0') && (ch <= '9')))
/*02*/       return (1);
/*03*/    else return (0);
/*04*/ } // fim valid_f
```

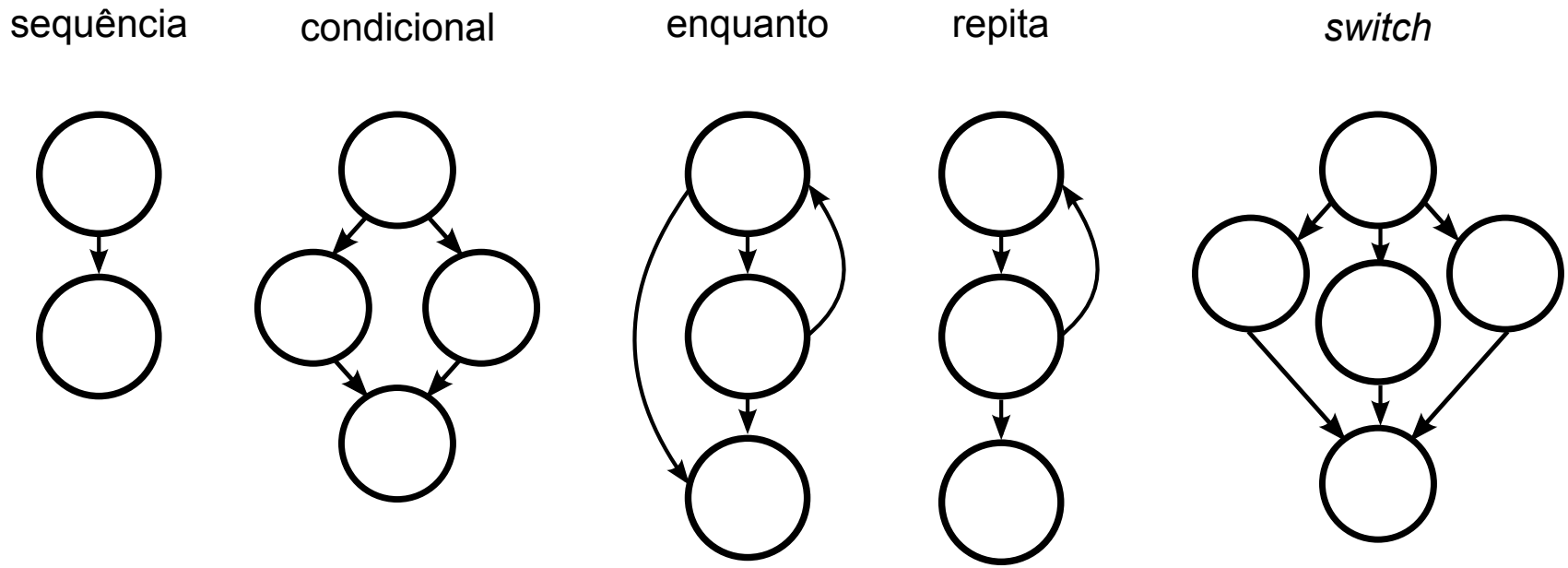
Definições

- Teste estrutural: conhecimento da estrutura interna do programa $\rightarrow$ aspectos de implementação são fundamentais para a geração/seleção de CTs
- Em geral, abordagens utilizam Grafo de Fluxo de Controle (GFC) ou "grafo de programa":
  - programa P pode ser decomposto em conjunto de blocos de comandos;
  - execução do primeiro comando de um bloco acarreta a execução de todos os outros comandos desse bloco;
  - todos os comandos de um bloco, possivelmente com exceção do primeiro, têm um único predecessor e exatamente um único sucessor, exceto possivelmente o último comando

Definições

- Representação de P como um GFC ($G = (N, E, s)$) consiste em estabelecer:
  - correspondência entre vértices (nós) e;
  - possíveis fluxos de controle entre nós por meio das arestas (arcos)
- GFC é dirigido, com um único nó de entrada $s \in N$ e um único nó de saída $o \in N$
  - Quando o primeiro comando do bloco é executado, todos os demais são executados sequencialmente;
  - Não existe desvio para nenhum comando dentro do bloco.
- A partir do GFC escolhem-se elementos que devem ser executados

Definições



Fluxo de controle

- "caminho" – sequência finita de nós $(n_1, \ldots, n_k)$, $k \geq 2$, tal que existe um arco de n_i para $n_i + 1$ para $i = 1, \ldots, k$
- "caminho simples" – se todos os nós são distintos, exceto possivelmente o primeiro e o último
- "caminho livre de laço" – se todos são distintos
- "caminho completo" – $n_1 = s$ e $n_k = o$
- "caminho não executável" – caminho para o qual não existe um dado de entrada que leve à sua execução

Fluxo de controle

- $IN(x)$ e $OUT(x)$ – número de arcos que entram e que saem do nó x, respectivamente
- Se $IN(x) = 0$, x é um nó de entrada, e se $OUT(x) = 0$, x é um nó de saída.

Fluxo de dados

- Ocorrências de variáveis em programas – "definição", "indefinição" ou "uso" – modelo de fluxo de dados
- Modelo de fluxo de dados de Maldonado – *definição*: quando um valor é armazenado em uma posição de memória
- Em geral, definição se variável está:
  1. no lado esquerdo de um comando de atribuição;
  2. em um comando de entrada;
  3. em chamadas de procedimentos como parâmetro de saída.

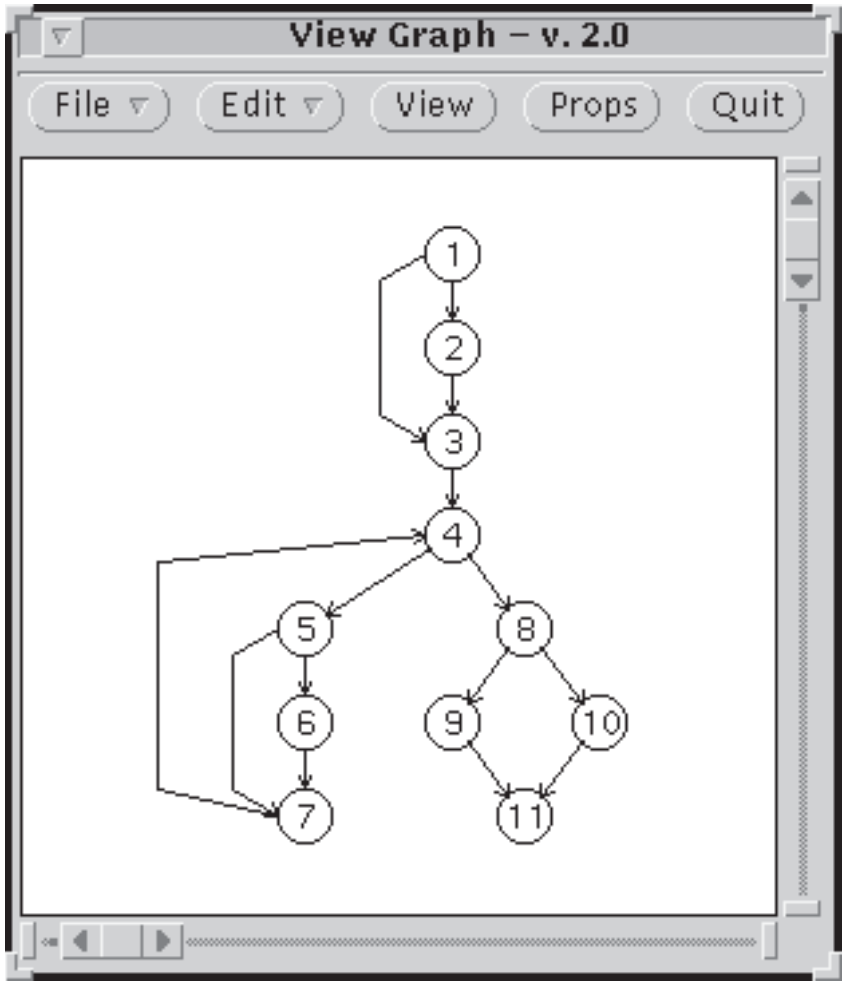
Fluxo de dados

- *uso*: quando referência não define valor à variável
- *c-uso*: uso em computação de valores
- *p-uso*: uso em predicado – afeta diretamente o fluxo de controle do programa
- Obs.: *c-uso*s são associados aos nós; *p-uso*s às arestas
- *indefinição*: quando, ou não se tem acesso ao seu valor, ou sua localização deixa de estar definida na memória

Fluxo de dados

- Considere uma variável x com definição no nó i
- "caminho livre de definição" com respeito a x: $(i, n_1, \ldots, n_m, j)$, $m \geq 0$, que não contenha definição de x nos nós $n_1, \ldots, n_m$, ou seja, do nó i ao nó j; do nó i ao arco (n_m, j)

GFC exemplo



Exemplo

- (2,3,4,5,6,7) é um caminho simples e livre de laços
- (1,2,3,4,5,7,4,8,9,11) é um caminho completo
- (1,3,4,8,9) é um "caminho não executável"
- Qualquer caminho completo que inclua tal caminho também é considerado não executável
- $\mathrm{length} = 0$ consiste em uma definição de variável
- achar !=‘\n’ e $\mathrm{length}{+}{+}$ representam p-uso e c-uso de variáveis

Exemplo

- Critérios estruturais – diferentes conceitos e elementos para definir requisitos. Exemplos:

Elemento	**Exemplo**	**Critério**
Nó	6	Todos-Nós
Arco	(5,6)	Todos-Arcos
Caminho	(1,2,3,4,8,9,11)	Todos-Caminhos
Definição de variáveis	length=0	Todas-Defs
Uso predicativo	achar !='\n'	Todos-P-Usos
Uso computacional	length++	Todos-C-Usos

Aplicabilidade

- Critérios da técnica estrutural podem ser utilizados em todas as fases de teste.
- Em geral, aplicados no teste de unidade pelo próprio desenvolvedor.
  - Garantir que a lógica da unidade em teste está correta.
- Entre as fases – teste de caminhos:
  - Caminhos dentro de uma unidade.
  - Caminhos entre unidades.
  - Caminhos entre sub-sistemas.
  - Caminhos entre o sistema todo.

Exercício

- Gere casos de teste para executar todos os nós e todos os arcos do programa Identifier. Identifique o caminho percorrido para cada caso de teste gerado.

```
main () {
/*01*/  char  achar; int length, valid_id; length = 0; valid_id = 1;
/*01*/  printf ("Identificador: ");
/*01*/  achar = fgetc (stdin); valid_id = valid_s(achar);
/*01*/  if(valid_id)
/*02*/    length = 1;
/*03*/  achar = fgetc (stdin);
/*04*/  while(achar != '\n') {
/*05*/    if(!(valid_f(achar)))
/*06*/       valid_id = 0;
/*07*/    length++;
/*07*/    achar = fgetc (stdin);
/*07*/  }
/*08*/  if(valid_id && (length >= 1) && (length <= 6))
/*09*/    printf ("Valido\n");
/*10*/  else printf ("Inválido\n");
/*11*/ }
```