MUTAÇÃO DE INTERFACE (MI)

JACKSON ANTONIO DO PRADO LIMA

SILVIA REGINA VERGILIO

DEFINIÇÃO

- O critério Mutação de Interface é uma extensão da Análise de Mutantes e preocupa-se em assegurar que as interações entre unidades de um programa sejam testadas – Exercitar as interações entre unidades num programa “integrado”.
- Possui como objetivo a inserção de perturbações nas conexões entre duas unidades de um programa – Erros de integração.
- Erros de integração ocorrem quando um valor incorreto é passado a uma unidade.

CLASSIFICAÇÃO

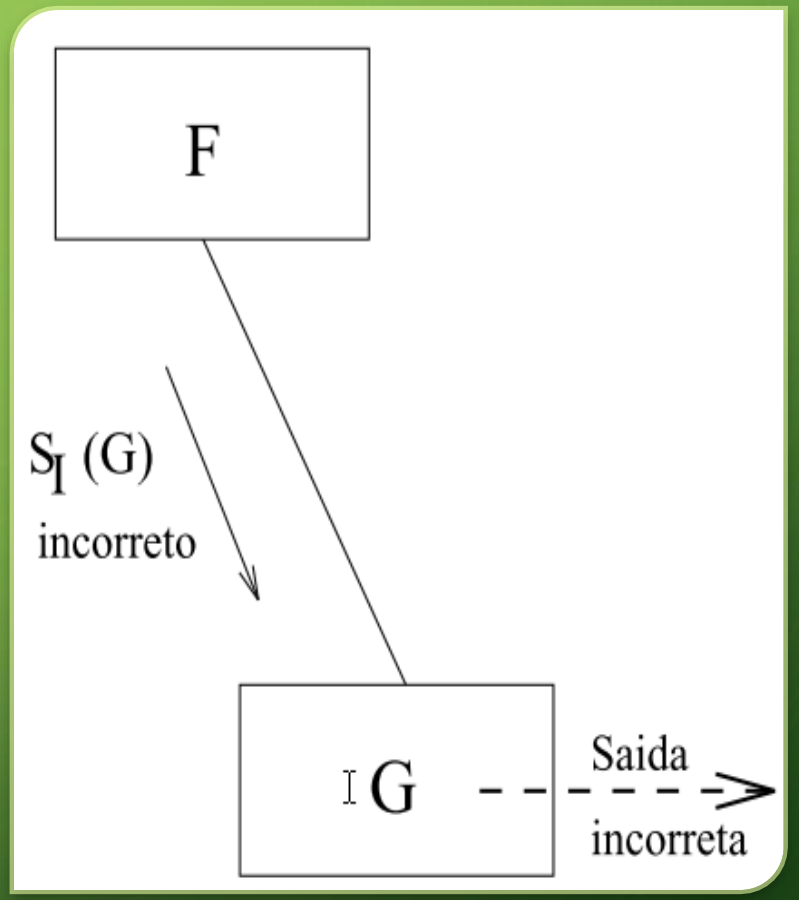
- Os erros de integração podem ser classificados em três categorias.

- Considere um programa **P** e um caso de teste **T** para **P**.
  - Suponha que em **P** existem unidades **F** e **G** e que **F** faz chamadas para **G**.
  - Considere $\mathbf{S_1(G)}$ a n-tupla (conjunto) de valores passados para G e $\mathbf{S_0(G)}$ a n-tupla de valores retornados de **G**.
  - Quando se executa P com o caso de teste **T**, um erro de integração é identificado na chamada de **F** para **G** quando:

CLASSIFICAÇÃO (TIPO 1)

Entrando em **G**, S_1**(G)** não têm os valores esperados e estes valores causam uma falha antes de retornar de **G**.

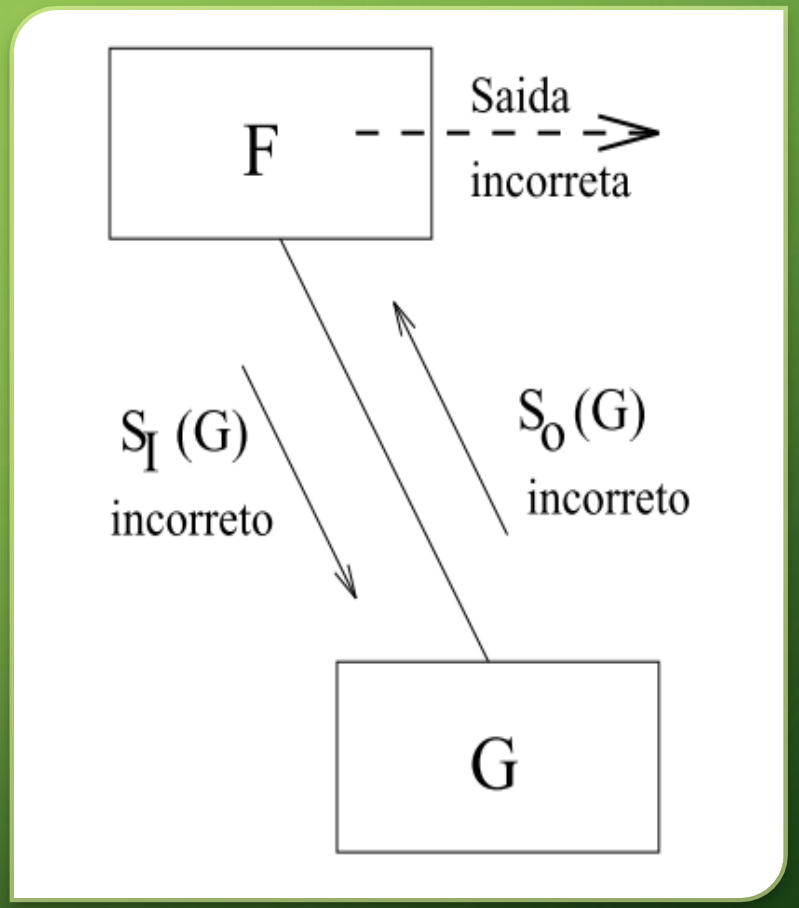
Exemplo: quando uma função é chamada com parâmetros incorretos fazendo com que a função chamada produza uma saída incorreta.



CLASSIFICAÇÃO (TIPO 2)

Entrando em **G**, $S_1(G)$ não têm os valores esperados e estes valores levam a um $S_0(G)$ incorreto, que por sua vez causa uma falha depois de retornarem de **G**.

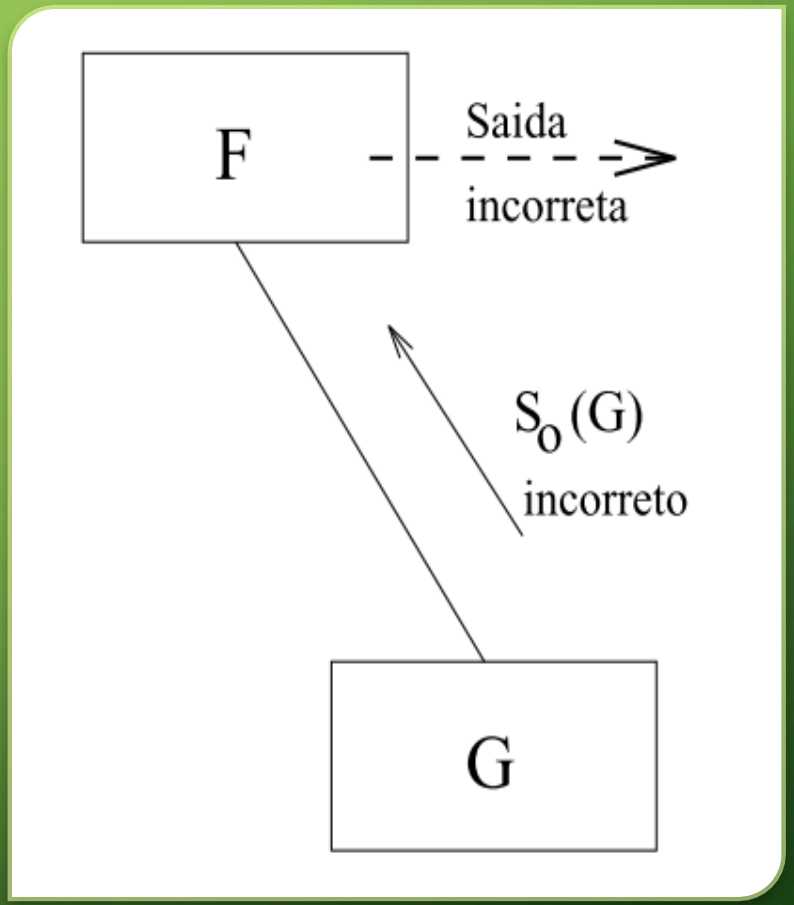
Exemplo: quando um parâmetro incorreto passado para a função é utilizado para calcular o valor de retorno.



CLASSIFICAÇÃO (TIPO 3)

Entrando em **G**, $S_1(G)$ têm os valores esperados, mas valores incorretos em $S_0(G)$ são produzidos dentro de **G** e estes valores incorretos causam uma falha após retornarem de **G**.

Esse tipo de erro pode ocorrer se uma função é chamada com todos os parâmetros corretos, mas internamente ela realiza um cálculo incorreto produzindo um valor de retorno não esperado que, posteriormente, leva a um resultado incorreto.



CLASSIFICAÇÃO (CONT.)

- Os tipos de erros podem ser de domínio ou computacional.
  - Domínio: quando um caminho incorreto é executado;
  - Computacional: quando o caminho correto é executado, mas o valor computado é incorreto.

- Dada uma função **F** que chama **G**, o primeiro ocorre quando um erro de domínio em **G** causa uma saída incorreta em **F**. O segundo ocorre quando um erro computacional em **G** produz um valor incorreto que é passado para **F** que, por sua vez, produz uma saída incorreta.

CLASSIFICAÇÃO (CONT.)

- A classificação dos tipos de erros é abrangente e não especifica o local do defeito que causa o erro. Ela simplesmente considera a existência de um valor incorreto entrando ou saindo de uma função chamada.
- Isso exclui, por exemplo, o caso em que $\mathbf{S_1(G)}$ tem os valores esperados mas um erro dentro de **G** produz uma saída incorreta antes do retorno de **G**. Neste caso, não existe nenhuma propagação de erro através da conexão **F-G** e esse tipo de erro deveria ser detectado no teste de unidade.

TROCA DE DADOS

- Em linguagens convencionais (C, Pascal, Fortran, etc.), unidades são conectadas por meio de chamadas de subprogramas (funções, procedimentos ou sub-rotinas).
- Em uma chamada da unidade **F** para a unidade **G**, existem quatro maneiras, não mutuamente exclusivas, de se trocarem dados entre as unidades.

TROCA DE DADOS (CONT.)

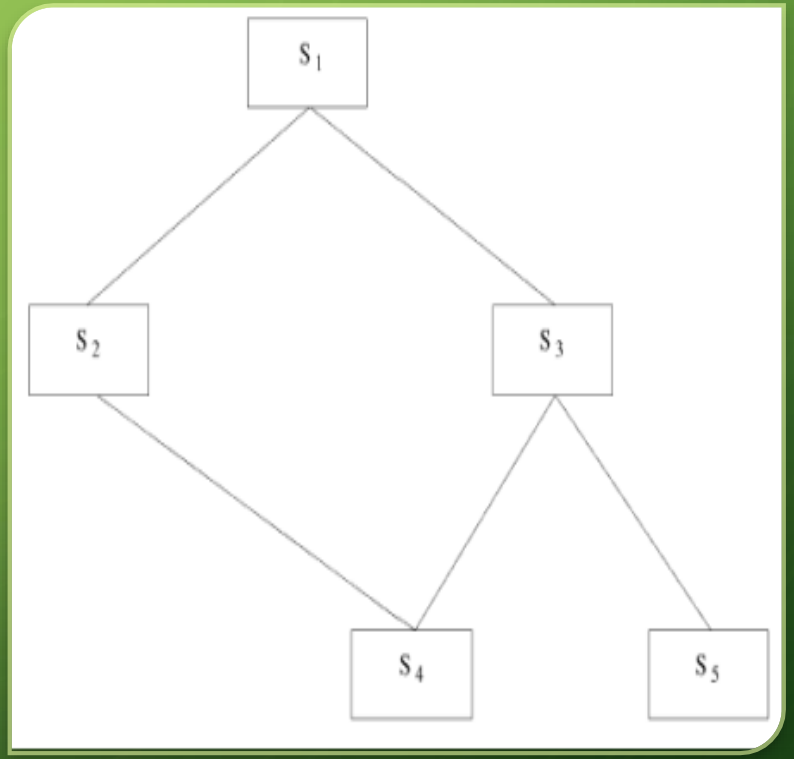
1. Dados são passados para **G** por meio de parâmetros de entrada (passagem por parâmetro);
2. Dados são passados para **G** e/ou retornados para **F** por meio de parâmetros de entrada-saída (passagem por referência);
3. Dados são passados para **G** e/ou retornados para **F** por meio de variáveis globais;
4. Dados são retornados para **F** por meio de comandos de retorno (como ***return*** em C).

Aplicação de mutação de interface ocorre somente nos pontos de interface (ou conexões) entre unidades

EXEMPLO

Para testar uma conexão **S_1-S_2**, introduzindo defeitos simples que levem a erros de integração, a Mutação de Interface aplica mutações em:

1. Pontos nos quais a unidade **S_1** chama a unidade **S_2**, portanto antes da execução de **S_2**;
2. Pontos dentro de **S_2** relacionados com sua interface.



CONDIÇÃO DE ALCANÇABILIDADE PARA MUTAÇÃO DE INTERFACE

- A execução de um mutante **M**, gerado pela conexão **S_1- S_2**, com um caso de teste **T** deve fazer com que o ponto no qual a mutação foi introduzida seja alcançado por meio de uma chamada de **S_1** para **S_2**.

Um operador de Mutação de Interface é focado no teste de interação entre duas unidades e não no teste de uma unidade

EXEMPLO

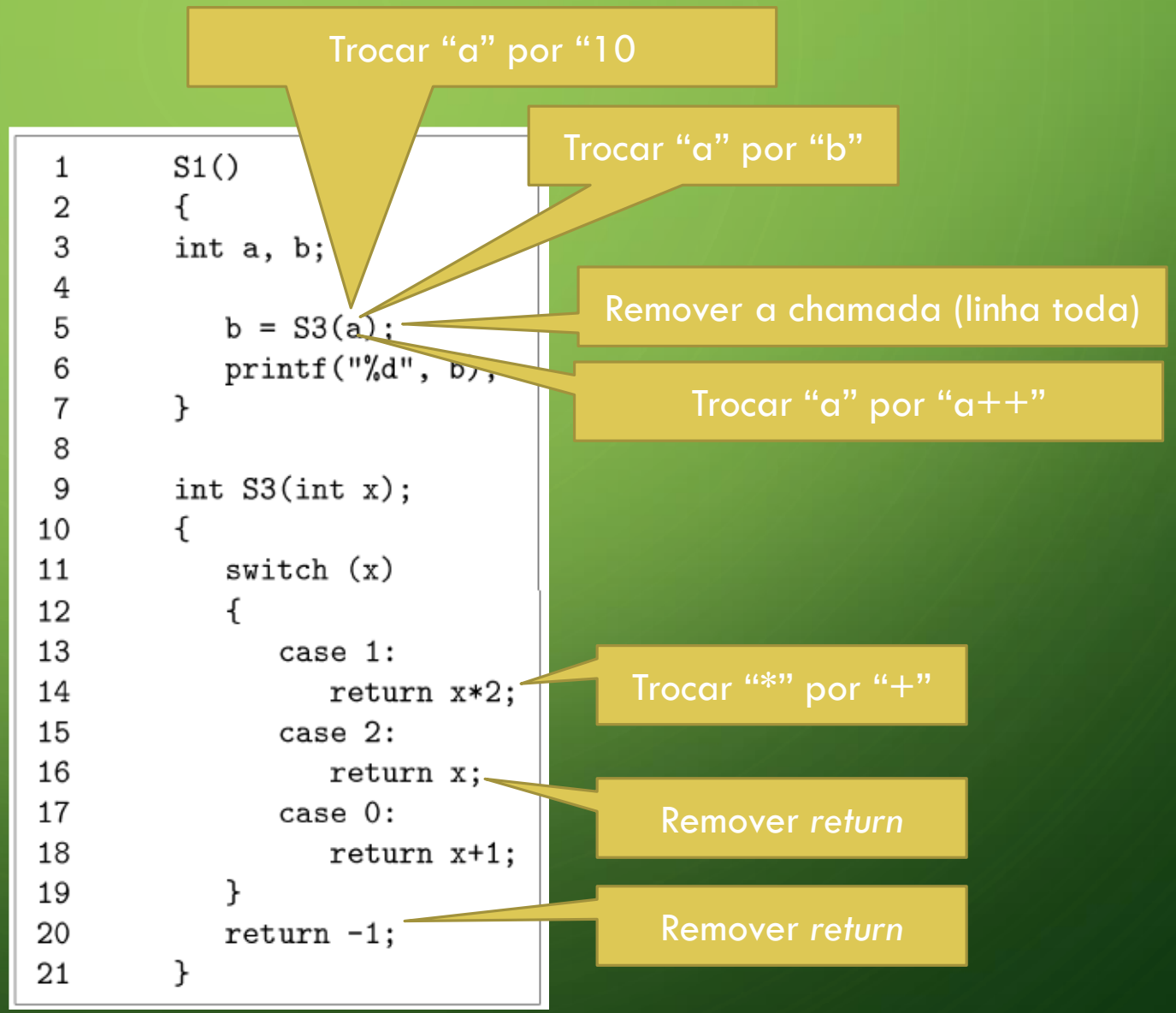
- Considere um operador de Mutação de Interface **OP(v)** que perturba uma variável **v**.
- O testador pode escolher, por exemplo, a conexão S_1-S_3 para testar, neste caso **OP** é aplicado sobre variáveis desta conexão.
- Digamos que será introduzido os erros do tipo 1 e 2, para isso **OP** pode mudra a entrada da unidade S_3. Esta entrada pode ser usada dentro de S_3 e causar um retorno incorreto para a unidade S_1.

EXEMPLO (CONT.)

OP pode ser aplicado de duas maneiras:

1. Primeiro, na passagem de parâmetros por valor, **OP** pode ser aplicado na chamada da unidade S_3 que ocorre dentro de S_1;
2. Segundo, na passagem de parâmetros por referência, **OP** pode ser aplicado nas referências que serão retornadas, dentro de S_3.

EXEMPLO (CONT.)




```
1       S1()
2       {
3       int a, b;
4
5          b = S3(a);
6          printf("%d", b);
7       }
8
9       int S3(int x);
10      {
11         switch (x)
12         {
13            case 1:
14               return x*2;
15            case 2:
16               return x;
17            case 0:
18               return x+1;
19         }
20         return -1;
21      }
```

OPERADORES DE MI PARA LINGUAGEM C

Dois grupos de operadores de MI são definidos considerando uma conexão entre duas unidades.

- Grupo I – Operadores aplicados dentro da função chamada (dentro do corpo da função **G**)
- Grupo II – Operadores aplicados dentro da função chamada que está fazendo a chamada (chamada para **G** dentro de **F**)

OPERADORES DE MI PARA LINGUAGEM C
GRUPO I

Estes operadores causam mudanças nos valores de entrado e/ou saída de funções.

Para a aplicação destes operadores é necessário identificar o lugar de onde a função foi chamada.

Mecanismo que habilite ou desabilite a mutação dependendo ou não de onde G é chamada de F (PROTEUM/IM)

OPERADORES DE MI PARA LINGUAGEM C
GRUPO I

Quando a conexão A-B irá ser testada:

- P(B): Conjunto que inclui os parâmetros formais de B.
- G(B): Conj. Variáveis globais acessadas por B.
- L(B): Conj. Variáveis declaradas em B (locais).
- E(B): Conj. Variáveis globais não acessadas em B.
- C(B): Conj. Constantes usadas em B.

OPERADORES DE MI PARA LINGUAGEM C
GRUPO I

- Operadores de Substituição Direta e Indireta de Variáveis
- Operadores de Variáveis de Incremento e Decremento
- Inserção de Operadores Unários
- Operadores de Retorno
- Operadores de Cobertura

GROUP I	
Name	Meaning
DirVarRepPar	Replaces interface variable by each element of P
DirVarRepGlo	Replaces interface variable by each element of G
DirVarRepLoc	Replaces interface variable by each element of L
DirVarRepExt	Replaces interface variable by each element of E
DirVarRepCon	Replaces interface variable by each element of C
DirVarRepReq	Replaces interface variable by each element of R
IndVarRepPar	Replaces non interface variable by each element of P
IndVarRepGlo	Replaces non interface variable by each element of G
IndVarRepLoc	Replaces non interface variable by each element of L
IndVarRepExt	Replaces non interface variable by each element of E
IndVarRepCon	Replaces non interface variable by each element of C
IndVarRepReq	Replaces non interface variable by each element of R
DirVarIncDec	Inserts/removes increment and decrement operations at interface variable uses
IndVarIncDec	Inserts/removes increment and decrement operations at non interface variable uses
DirVarAriNeg	Inserts arithmetic negation at interface variable uses
IndVarAriNeg	Inserts arithmetic negation at non interface variable uses
DirVarLogNeg	Inserts logical negation at interface variable uses
IndVarLogNeg	Inserts logical negation at non interface variable uses
DirVarBitNeg	Inserts bit negation at interface variable uses
IndVarBitNeg	Inserts bit negation at non interface variable uses
RetStaDel	Deletes `return` statement
RetStaRep	Replaces `return` statement
CovAllNod	Coverage of all nodes
CovAllEdg	Coverage of all edges

OPERADORES DE MI PARA LINGUAGEM C
GRUPO II

Dado uma conexão A-B os operadores são aplicados na instrução onde A chama B e sobre estes argumentos da chamada.

Quando um argumento é uma expressão os operadores são aplicados sobre a expressão inteira e não sobre os objetos da expressão como variáveis ou constants.

OPERADORES DE MI PARA LINGUAGEM C
GRUPO II

- Substituição de Argumentos
- Troca de Argumentos
- Eliminação de Argumentos
- Inserção de Operadores Unários
- Exclusão de Chamada de Função

GROUP II	
Name	Meaning
ArgRepReq	Replaces arguments by each element of R
ArgStcAli	Switches arguments of compatible type
ArgStcDif	Switches arguments of non compatible type
ArgDel	Remove argument
ArgAriNeg	Inserts arithmetic negation at arguments
ArgLogNeg	Inserts logical negation at arguments
ArgBitNeg	Inserts bit negation at arguments
ArgIncDec	Argument Increment and Decrement
FunCalDel	Removes function call

REFERÊNCIAS

- M. E. Delamaro, J. C. Maldonado, e M. Jino. *Introdução ao teste de software*, páginas 108-113. Elsevier, Rio de Janeiro, RJ, Brasil, 2007.

MATERIAL

- www.inf.ufpr.br/japlima