Instalando e Testando com a Poke-Tool

Instalando a Poke-Tool

Para a instalação da Poke-Tool siga os passos abaixo.

1º passo: Faça o download do fonte compilado para teu diretório de usuário:

```
$ wget http://poketool.iv.org.br/portal/downloads/poke-tool-build-for-fedora-3-i686/poke-i686-2007-06-06.tgz
```

Obs.: Para saber o caminho do teu diretório de usuário rode este comando no terminal:

```
$ echo $HOME
```

2º passo: crie um diretório para a instalação da ferramenta e copie o arquivo baixado para o diretório criado. Como exemplo instalei no meu diretório $HOME:

```
$ mkdir poketool
$ cd poketool
$ cp $HOME/poke-i686-2007-06-06.tgz $HOME/poketool/poke-i686-2007-06-06.tgz
```

3º passo: descompacte o arquivo:

```
$ tar xvf poke-i686-2007-06-06.tgz
```

e saia do diretório:

```
$ cd
```

4º passo: O diretório /bin da Poketool deve ser incluído agora no PATH do sistema e as variáveis de ambiente NEWLITABS e NEWPOKETABS devem ser ajustadas. Para isso execute os camandos abaixo:

```
$ export PATH=$PATH:"$HOME/poketool/bin/"
$ NEWLITABS="$HOME/poketool/bin/litabs"
$ NEWPOKETABS="$HOME/poketool/bin/poketabs"
$ export NEWLITABS NEWPOKETABS
```

Para que esta modificação seja permanente você deve alterar o arquivo de configuração de usuário *shell.* Abra o arquivo .bashrc da tua pasta de usuário (se a optar por abrir arquivo navegando na pasta com a interface gráfica – no Ubuntu, Nautilus – o arquivo .bashrc é oculto, portanto deverá pressionar Ctrl+H para visualizar o arquivo):

```
$ gedit $HOME/.bashrc
```

e adicione os 4 comandos anteriores(alteração do PATH e variáveis de ambiente) no final do arquivo.

5º passo: faça logout e divirta-se com a ferramenta.

Uma outra maneira de instalar a Poke-Tool é criar um script de instalação. Se você já baixou e descompactou a ferramenta, tem agora um diretório com nome “poke-i686-2007-06-06”. Se o nome do arquivo não for esse o diretório deve ser renomeado e a instalação da Poke-Tool será da seguinte maneira:

1º passo: Abra o gedit

```
$ gedit &
```

e digite as seguintes linhas:

```
#!/bin/sh
#########################################################################
#                                                                       #
#     Script de instalação e configuração da Poke Tool no Ubuntu  #
#                                                                       #
#     Thiago Furtado de Mendonça – thiagofurtado17@gmail.com      #
#           21/09/2010                                                  #
#                                                                       #
#########################################################################

if ! test –d /opt/poketool;
then echo 'Criando diretório de instalação...';
mkdir /opt/poketool;
fi;

echo 'Copiando arquivos...'
cp –r ../poke–i686–2007–06–06/bin /opt/poketool

echo 'Configurando permissões'
chmod –R 777 /opt/poketool/

echo '\n# Configuração de variáveis de ambiente para Poke–Tool\n' >>
$HOME/.bashrc

echo export PATH=$PATH:"/opt/poketool/bin/" >> $HOME/.bashrc
echo NEWLITABS="/opt/poketool/bin/litabs" >> $HOME/.bashrc
echo NEWPOKETABS="/opt/poketool/bin/poketabs" >> $HOME/.bashrc
echo export NEWLITABS NEWPOKETABS >> $HOME/.bashrc
```

e salve o arquivo dentro da pasta “poke-i686-2007-06-06” como install.sh. Agora clique com o botão direito no arquivo, abra a *tab* 'Permissões' e dê permissão de execução ao arquivo.

3º passo: execute o script como super usuário:

```
$ sudo ./install.sh
```

Faça logout no sistema e a Poke-Tool já estará funcionando.

Testando com a Poke-Tool

Para exemplificação neste tutorial o programa teste.c a seguir será utilizado

```c
#include <stdio.h>

int f()
{
    int x, y;
    printf("Enter x: \n");
    scanf("%d", &x);
    printf("x value = %d\n", x);
    printf("Enter y: \n");
    scanf("%d", &y);
    printf("x value = %d\n", y);
    while(x > 10)
    {
        x = x - 10;
        if(x == 10)
            break;
    }
    if((y < 20) && (x%2==0))
        y = y + 20;
    else
        y = y - 20;
    return 2 * x + y;
}

int main()
{
    printf("output = %d\n", f());
    return 0;
}
```

teste.c

Poketool

Antes de começar a tentar fazer os testes, certifique-se que o arquivo de código fonte, no nosso caso o arquivo *teste.c* está no diretório /bin da ferramenta Poketool. Se você optou pelo primeiro modo de instalação é o caminho $HOME/poketoo/bin. Se você optou por rodar o script pronto, é o caminho /opt/poketool/bin (neste caso, verifique se teu usuário tem permissões nesse diretório – é interessante que saiba a utilidade do comando *chown*).

A primeira coisa a se fazer é gerar os arquivos para a análise do programa. Para isso utilize a ferramenta *newpoketool* do pacote para a geração destes arquivo da seguinte forma:

```
$ cd "/diretório/bin/da/poketool"
```

troque "/diretório/bin/da/poketool" por $HOME/poketool/bin ou /opt/poketool/bin dependendo

da instalação que usou.

Agora rode efetivamente a ferramenta. Rodando o comando *newpoketool* sem parâmetros, um pequeno *how* to será exibido na tela explicando como deve ser executado o comando:

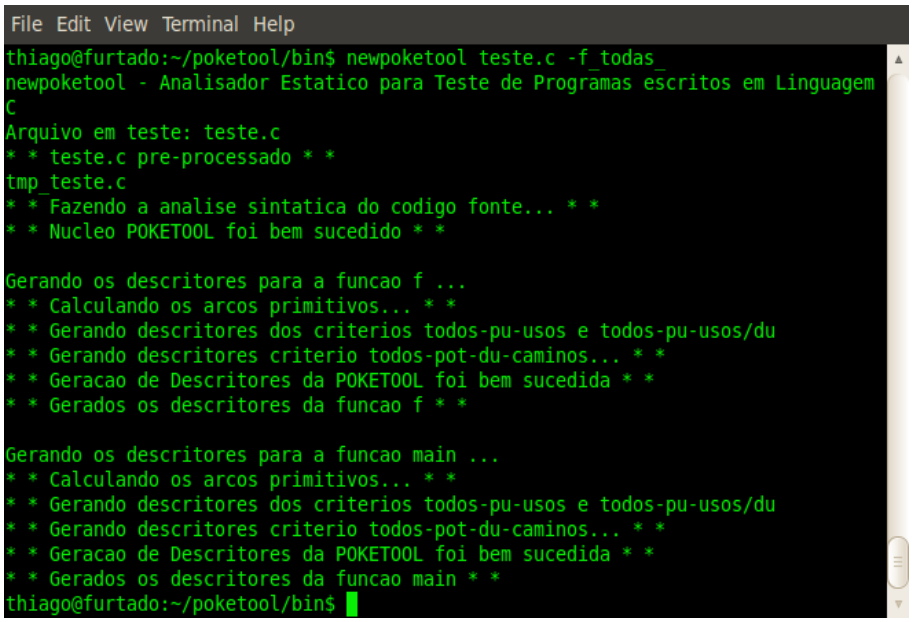
```
$ newpoketool <nome do arquivo> [-L<nivel modelo de dados>] [-c<selecao de criterios>] [-I<Diretorios de Include>] [-D<defines>]  -f<funcao> [-f<funcao>]
```

Para os parâmetros de execução deverão ser usados os seguintes valores:

- Nome do arquivo: arquivo de código com extenção .c
- Nível modelo de dados: valor entre 0 e 5, que indica o nível de modelo de dados usado. A omissão deste parâmetro implica na execução do teste com nível 2.
- Seleção de critérios: As opções abaixo são aceitas
  - gfc: todos nos e todos arcos;
  - df: todos nos, todos arcos, todos p-usos e todos usos;
  - dup: todos nos, todos arcos e todos du-caminhos;
  - pu: todos nos, todos arcos, todos pot-usos e todos pot-usos/du;
  - pdu: todos nos, todos arcos e todos pot-du-caminhos;
  - df-pu:todos nos, todos arcos, todos p-usos, todos usos, todos pot-usos e todos pot-usos/du;
  - parâmetro omitido: todos critérios são levados em consideração;
- Diretórios de include: parâmetros para indicação dos diretórios incluídos nos *#include.*
- Defines: todas as macros definidas com *#define.*
- Funcõo: funções que deseja testar.

No nosso caso aqui gerei os arquivos com o comando:

```
$ newpoketool teste.c -f_todas_
```



Execução do comando newpoketool sobre teste.c.

O parâmetro -f usado com a opção _todas_ executa a ferramenta para teste de todas as funções declaradas no arquivo com o fonte.

A execução do comando vai gerar arquivos e diretórios referentes aos casos de teste. O arquivo *poketool.h* tem declarado a função ponta_de_prova() usado para marcação de pontos no código modificado. O código modificado pode ser verificado no arquivo gerado com nome *testeprog.c*. Este arquivo contém comentários para visualização dos blocos (nós) que farão parte do grafo de fluxo de controle. Arquivos com extensão *.li* e *.nli* são arquivos contendo uma linguagem intermediária utilizada pela Poke-Tool. Um arquivo temporário com nome *tmp_teste.c* é gerado com o programa reescrito incluíndo as funções e definições das bibliotecas inclúidas no projeto.

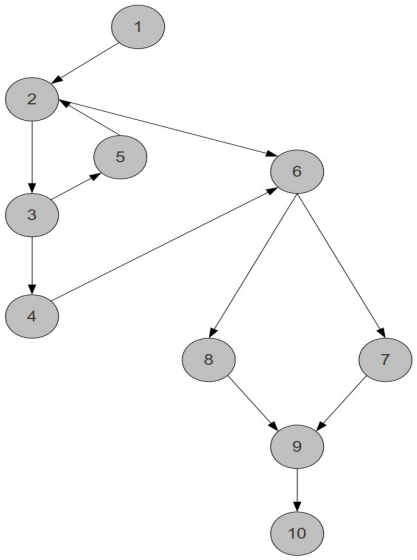
Para cada função um diretório com o mesmo nome é criado. Neste diretório estão os arquivos que efetivamente contém informações sobre os testes que devem ser aplicados sobre o código a ser testado. São arquivos com descritores dos testes.

O arquivo f.gfc contém o grafo de fluxo de controle da função f. Pode ser verificado no arquivo testeprog.c os comandos referentes a cada nó do grafo.

```
329  int f()
330  /* 1 */ {
331      FILE *path = fopen("f/path.tes","a");
332      static int printed_nodes = 0;
333  /* 1 */ int x, y;
334      ponta_de_prova(1);
335  /* 1 */ printf("Enter x: \n");
336  /* 1 */ scanf("%d", &x);
337  /* 1 */ printf("x value = %d\n", x);
338  /* 1 */ printf("Enter y: \n");
339  /* 1 */ scanf("%d", &y);
340  /* 1 */ printf("x value = %d\n", y);
341  /* 2 */ while(x > 10)
342  /* 3 */ {
343          ponta_de_prova(2);
344          ponta_de_prova(3);
345  /* 3 */     x = x - 10;
346  /* 3 */     if(x == 10)
347  /* 4 */     {
348              ponta_de_prova(4);
349  /* 4 */         goto out0_break;
350  /* 4 */     }
351          ponta_de_prova(5);
352  /* 5 */ }
353      ponta_de_prova(2);
354      out0_break:;
355      ponta_de_prova(6);
356  /* 6 */ if((y < 20) && (x%2==0))
357  /* 7 */ {
358          ponta_de_prova(7);
359  /* 7 */     y = y + 20;
360  /* 7 */ }
361  /* 8 */ else
362  /* 8 */ {
363          ponta_de_prova(8);
364  /* 8 */     y = y - 20;
365  /* 8 */ }
366      ponta_de_prova(9);
367      ponta_de_prova(10);
368      fclose(path);
369  /* 9 */ return 2 * x + y;
370  /* 10 */ }
```

Fonte com comentários dos nós do GFC.

O grafo gerado pela Poketool é escrito no arquivo com extensão .gfc. Nesse arquivo a primeira linha equivale ao número de nós. Logo abaixo no arquivo é descrito para cada nó os nós que tem ligação. Abaixo está a visualização do grafo de fluxo de controle.



Grafo de fluxo de controle.

No arquivo grafodef.tes é possível ainda ver as definições, p-usos e c-usos para cada nó do grafo.

O próximo passo é compilar normalmente o **programa modificado** gerado pela Poke-Tool. O arquivo gerado para nosso teste é o arquivo *testeprog.c*.

```
$ gcc –o testeprog testeprog.c
```

Poxeexec

Agora você poderá executar os casos de testes. Para este passo o executável *pokeexec* é utilizado passando o executável gerado pela compilação anterior como parâmetro. O pokeexec passa a execução para o programa em teste para a execução dos casos de testes. A ferramenta é utilizada da seguinte forma:

```
$ pokeexec <executavel> –+lk –rnum –f<function name> [–f<function name>]
```

Para os parâmetros de execução deverão ser usados os seguintes valores:

- +l: programa aceita parametros por linha de comando;
- -l: programa não aceita parametros por linha de comando;
- +k: programa aceita parametros do teclado;
- -k: programa nao aceita parametros do teclado;
- -rnum: começa a execução do teste pelo caso 'num'.

Aqui executei os casos de testes sobre a função *f*.

`$ pokeexec testeprog –ff` - (a omissão dos parâmetros l e k fazem com que a pokeexec pergunte se terá ou não entrada de teclado e linha de comando).

Escolha dos testes

Analisando a complexidade, 4 testes são necessários mas visualizando o código e o grafo percebemos que para que todos os critérios sejam executados, precisamos de mais casos de testes.

Para os valores de x e y, neste tutorial escolhi valores próximo aos valores limites que as decisões nos laços condicionais testam. Os valores usados foram:

	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º
x	9	10	11	9	10	11	9	10	11	30	30	30
y	19	19	19	20	20	20	21	21	21	19	20	21

A saída da execução da *pokeexec* neste exemplo foi a seguinte:

```
thiago@furtado:~/poketool/bin$ pokeexec testeprog -ff
========================
pokeexec: Starting test session to program "testeprog" ...

pokeexec: Deleting all files of the directory "output"
pokeexec: Accepts parameters in command line ? (y/n) [y]
n
pokeexec: Accepts keyboard input? (y/n) [y]
y
pokeexec: Deleting all files of the directory "keyboard"
========================
pokeexec:  Executing test case 1.
pokeexec: Keyboard input:
9 19
Enter x:
x value = 9
Enter y:
x value = 19
output = 17

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 2.
pokeexec: Keyboard input:
10 19
Enter x:
x value = 10
Enter y:
x value = 19
output = 59

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
```

```
y
pokeexec:  Executing test case 3.
pokeexec: Keyboard input:
11 19
Enter x:
x value = 11
Enter y:
x value = 19
output = 1

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 4.
pokeexec: Keyboard input:
9 20
Enter x:
x value = 9
Enter y:
x value = 20
output = 18

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 5.
pokeexec: Keyboard input:
10 20
Enter x:
x value = 10
Enter y:
x value = 20
output = 20

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 6.
pokeexec: Keyboard input:
11 20
Enter x:
x value = 11
Enter y:
x value = 20
output = 2

========================
```

```
pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 7.
pokeexec: Keyboard input:
9 21
Enter x:
x value = 9
Enter y:
x value = 21
output = 19

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 8.
pokeexec: Keyboard input:
10 21
Enter x:
x value = 10
Enter y:
x value = 21
output = 21

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 9.
pokeexec: Keyboard input:
11 21
Enter x:
x value = 11
Enter y:
x value = 21
output = 3

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 10.
pokeexec: Keyboard input:
30 19
Enter x:
x value = 30
Enter y:
x value = 19
output = 59

========================
```

```
pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 11.
pokeexec: Keyboard input:
30 20
Enter x:
x value = 30
Enter y:
x value = 20
output = 20

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
y
pokeexec:  Executing test case 12.
pokeexec: Keyboard input:
30 21
Enter x:
x value = 30
Enter y:
x value = 21
output = 21

========================

pokeexec: Do you want to continue to execute test cases? (y/n)[n]
n
pokeexec: OK
========================

Type <RETURN> to finish test case submission...
```

Os diretórios abaixo são criados:

- keyboard: contém arquivos das entradas do teclado digitadas para execução do programa em teste;
- output: contém arquivos com saída para cada teste executado.
- Input: contém arquivos com entrada por linha de comando.

No diretório da função testada, arquivos *pathN.tes* são criados contendo o caminho percorrido na execução no teste N.

Executado os testes você deverá analisar a cobertura dos teus testes. Para isso outra ferramenta disponibilizada pela Poke-Tool será utilizada.

Pokeaval

Com os casos de testes rodados, você terá o auxílio do comando *newpokeaval* que vai analisar os arquivos *pathN.tes* para uma análise de cobertura.

O comando é usado usado como segue:

$ newpokeaval -d<dir. funcao> [-h|-f] -pdu|-pu|-pudu|-nos|-arcs|-puses|-uses|-du [-ne|-e] x to y

Para os parâmetros na execução deverão ser usados os seguintes valores:

- -h: mantém o arquivo histórico das outras execuções (opcional) ;
- -f : gera apenas arquivos de frequência sem gerar arquivos de saída (opcional) ;
- dir. funcao: diretório da funçao a ser avaliada;
- Critérios de **avaliação**:
  - pdu: todos potenciais du-caminhos ;
  - pu: todos potenciais usos ;
  - pudu: todos potenciais usos/du ;
  - nos: todos nos ;
  - arcs: todos arcos ;
  - puses: todos usos predicativos ;
  - uses: todos usos;
  - du: todos du-caminhos ;
- [-ne|-e] : apresenta associações não executadas ou executadas (opcional) ;
- x to y : inicia execução do caso de teste x e termina no caso y (obrigatorio) .

Aqui executei a *newpokeaval* assim:

```
$ newpokeaval –df –uses –puses –arcs –nos 1 to 12
```

E a saída foi a seguinte:

```
thiago@furtado:~/poketool/bin$ newpokeaval –df –uses –puses –arcs –nos
1 to 12
newpokeaval – Avaliador de Casos de Teste da POKE–TOOL

            **** Avaliando Caso de Teste Numero 1 ****

* * Realizando a avaliacao do caso de teste * *

* * Avaliacao do caso de teste foi bem sucedida * *

            **** Avaliando Caso de Teste Numero 2 ****

* * Realizando a avaliacao do caso de teste * *

* * Avaliacao do caso de teste foi bem sucedida * *

            **** Avaliando Caso de Teste Numero 3 ****
```

* * Realizando a avaliacao do caso de teste * *

* * Avaliacao do caso de teste foi bem sucedida * *

**** Avaliando Caso de Teste Numero 4 ****

* * Realizando a avaliacao do caso de teste * *

* * Avaliacao do caso de teste foi bem sucedida * *
newpokeaval: Msg: Caso de teste 4 NAO matou nenhuma associacao do criterio todos usos!
newpokeaval: Msg: Caso de teste 4 NAO matou nenhuma associacao do criterio todos p-usos!
newpokeaval: Msg: Caso de teste 4 NAO matou nenhum arco do criterio todos arcos!
newpokeaval: Msg: Caso de teste 4 NAO matou nenhum no' do criterio todos nos!

**** Avaliando Caso de Teste Numero 5 ****

* * Realizando a avaliacao do caso de teste * *

* * Avaliacao do caso de teste foi bem sucedida * *
newpokeaval: Msg: Caso de teste 5 NAO matou nenhuma associacao do criterio todos usos!
newpokeaval: Msg: Caso de teste 5 NAO matou nenhuma associacao do criterio todos p-usos!
newpokeaval: Msg: Caso de teste 5 NAO matou nenhum arco do criterio todos arcos!
newpokeaval: Msg: Caso de teste 5 NAO matou nenhum no' do criterio todos nos!

**** Avaliando Caso de Teste Numero 6 ****

* * Realizando a avaliacao do caso de teste * *

* * Avaliacao do caso de teste foi bem sucedida * *
newpokeaval: Msg: Caso de teste 6 NAO matou nenhuma associacao do criterio todos usos!
newpokeaval: Msg: Caso de teste 6 NAO matou nenhuma associacao do criterio todos p-usos!
newpokeaval: Msg: Caso de teste 6 NAO matou nenhum arco do criterio todos arcos!
newpokeaval: Msg: Caso de teste 6 NAO matou nenhum no' do criterio todos nos!

**** Avaliando Caso de Teste Numero 7 ****

* * Realizando a avaliacao do caso de teste * *

* * Avaliacao do caso de teste foi bem sucedida * *

```
            newpokeaval: Msg: Caso de teste 7 NAO matou nenhuma associacao do
criterio todos usos!
            newpokeaval: Msg: Caso de teste 7 NAO matou nenhuma associacao do
criterio todos p-usos!
            newpokeaval: Msg: Caso de teste 7 NAO matou nenhum arco do criterio
todos arcos!
            newpokeaval: Msg: Caso de teste 7 NAO matou nenhum no' do criterio
todos nos!

                        **** Avaliando Caso de Teste Numero 8 ****

            * * Realizando a avaliacao do caso de teste * *

            * * Avaliacao do caso de teste foi bem sucedida * *
            newpokeaval: Msg: Caso de teste 8 NAO matou nenhuma associacao do
criterio todos usos!
            newpokeaval: Msg: Caso de teste 8 NAO matou nenhuma associacao do
criterio todos p-usos!
            newpokeaval: Msg: Caso de teste 8 NAO matou nenhum arco do criterio
todos arcos!
            newpokeaval: Msg: Caso de teste 8 NAO matou nenhum no' do criterio
todos nos!

                        **** Avaliando Caso de Teste Numero 9 ****

            * * Realizando a avaliacao do caso de teste * *

            * * Avaliacao do caso de teste foi bem sucedida * *
            newpokeaval: Msg: Caso de teste 9 NAO matou nenhuma associacao do
criterio todos usos!
            newpokeaval: Msg: Caso de teste 9 NAO matou nenhuma associacao do
criterio todos p-usos!
            newpokeaval: Msg: Caso de teste 9 NAO matou nenhum arco do criterio
todos arcos!
            newpokeaval: Msg: Caso de teste 9 NAO matou nenhum no' do criterio
todos nos!

                        **** Avaliando Caso de Teste Numero 10 ****

            * * Realizando a avaliacao do caso de teste * *

            * * Avaliacao do caso de teste foi bem sucedida * *

                        **** Avaliando Caso de Teste Numero 11 ****

            * * Realizando a avaliacao do caso de teste * *

            * * Avaliacao do caso de teste foi bem sucedida * *
            newpokeaval: Msg: Caso de teste 11 NAO matou nenhuma associacao do
criterio todos usos!
            newpokeaval: Msg: Caso de teste 11 NAO matou nenhuma associacao do
```

criterio todos p-usos!
newpokeaval: Msg: Caso de teste 11 NAO matou nenhum arco do criterio todos arcos!
newpokeaval: Msg: Caso de teste 11 NAO matou nenhum no' do criterio todos nos!

**** Avaliando Caso de Teste Numero 12 ****

* * Realizando a avaliacao do caso de teste * *

* * Avaliacao do caso de teste foi bem sucedida * *
newpokeaval: Msg: Caso de teste 12 NAO matou nenhuma associacao do criterio todos usos!
newpokeaval: Msg: Caso de teste 12 NAO matou nenhuma associacao do criterio todos p-usos!
newpokeaval: Msg: Caso de teste 12 NAO matou nenhum arco do criterio todos arcos!
newpokeaval: Msg: Caso de teste 12 NAO matou nenhum no' do criterio todos nos!

ASSOCIACOES DO CRITERIO TODOS-USOS executadas:

<1,(2,3), x>
<1,(2,6), x>
<1,3, x>
<1,(3,4), x>
<1,(3,5), x>
<1,(6,7), x>
<1,(6,8), x>
<1,9, x>
<1,(6,7), y>
<1,(6,8), y>
<1,7, y>
<1,8, y>
<3,(3,4), x>
<3,(3,5), x>
<3,(6,7), x>
<3,(6,8), x>
<3,9, x>
<3,(2,3), x>
<3,(2,6), x>
<3,3, x>
<7,9, y>
<8,9, y>

Cobertura Total = 100.000000

ASSOCIACOES DO CRITERIO TODOS-USOS PREDICATIVOS executadas:

<1,(2,3), x>
<1,(2,6), x>
<1,(3,4), x>
<1,(3,5), x>
<1,(6,7), x>
<1,(6,8), x>
<1,(6,7), y>
<1,(6,8), y>
<3,(3,4), x>
<3,(3,5), x>
<3,(6,7), x>
<3,(6,8), x>
<3,(2,3), x>
<3,(2,6), x>

Cobertura Total = 100.000000

ARCOS DO CRITERIO TODOS-ARCOS executados:

1) arco (2, 6)
2) arco (3, 4)
3) arco (3, 5)
4) arco (6, 7)
5) arco (6, 8)

Cobertura Total = 100.000000

NOS DO CRITERIO TODOS-NOS executados:

  1   2   3   4   5   6   7   8   9 10

Cobertura Total = 100.000000