SSC5877 – Validação Verificação e Teste de Software

Teste Estrutural – Fluxo de Dados

Prof. Marcio E. Delamaro

delamaro@icmc.usp.br

Critérios Baseados em Fluxo de Dados

- Critérios pertencentes à Técnica de Teste Caixa Branca.
- Complementares aos critérios baseados em fluxo de controle.
- Busca testar o uso das variáveis em um programa, ou seja, como os dados são usados nas computações.

Motivação

- Teste de fluxo de dados é uma ferramenta poderosa para o uso incorreto de valores resultante de erros de codificação.
- Tornou-se popular com a publicação do trabalho de Rapps e Weyuker (1982):

  “"It is our belief that, just as one would not feel confident about a program without executing every statement in it as part of some test, one should not feel confident about a program without having seen the effect of using the value produced by each and every computation."

Ciclo de vida das variáveis

- Variáveis são criadas, usadas e destruídas.
- Em algumas linguagens de programação (BASIC e FORTRAN, por exemplo) a criação e destruição são automáticas.
- Em outras (C, C++ e Java, por exemplo), a criação deve ser explícita.
- Exemplos de declaração:

```
1 int x;       // x é criada como um inteiro
2 String y; // y é criada como uma String
```

Ciclo de vida das variáveis

- Declarações, em geral, ocorrem dentro de um bloco.
  - Variáveis são criadas quando a definição das mesmas são executadas.
  - Variáveis são destruídas no final do bloco (conceito de escopo da variável).

```
1  {            // início do bloco 1
2     int x;    // x é definida no bloco 1
3     ...;      // x pode ser usada aqui
4     {         // início do blodo 2
5        int y;// y é definida no bloco 2
6        ...;  // x e y podem ser usadas aqui
7     }         // y é automaticamente destruída no final do bloco 2
8     ...;      // x ainda pode ser usada daqui em diante
9  }            // x é automaticamente destruída no final do bloco 1
```

utilização de variáveis

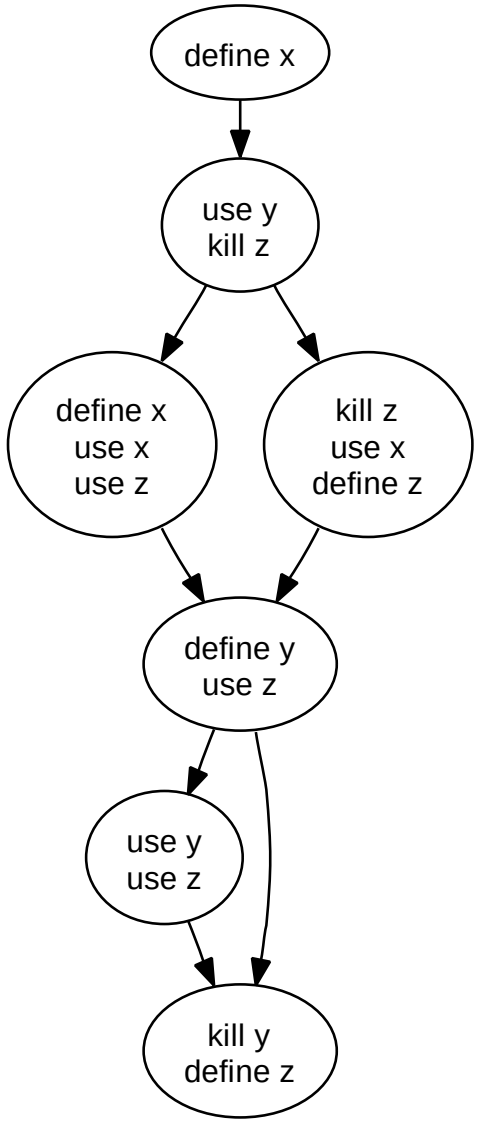
- No ciclode vida de uma variável podemos ainda distinguir dois tipos de utilização de uma variável
  - Quando o valor da variável é modificado. Por exemplo, via comando de atribuição: `a = 10` e `b = 5`.
  - Quando o valor da variável é usado mas seu valor não é modificado Por exemplo: `a = b * 1`.

Grafo def-uso

- Para avaliar os diferentes estados das variável no programa é utilizado um grafo denominado **Grafo Definição-Uso** ou Grado Def-Uso Rapps e Weyuker (1982).
- Semelhante ao Grafo de Fluxo de Controle.
- Inclui ainda variáveis definidas, usadas e destruídas em cada nó.
- Análise estática e dinâmica do Grafo Def-Uso.
  - Estático: examinar o grafo procurando por problemas, simulando mentalmente o comportamento.
  - Dinâmico: executar o programa com casos de testes e avaliar o resultado.

Grafo def-uso

- Exemplo de GFC com anotações de fluxo de dados.



Teste de fluxo de dados

- Assumir que o fluxo de controle do módulo está correto.
- Criar casos de testes de modo que:
  - Cada definição de variável é rastreada até seus usos.
  - Cada uso é rastreado a partir de sua definição correspondente.
- Para fazer isso:
  - Enumerar os caminhos no GDU usando a mesma abordagem do teste de fluxo de controle.
  - Criar casos de testes que cubram cada par “definição-uso” (associações de fluxo de dados) entre as variáveis.

Conceitos

- A ocorrência de variáveis em um programa pode ser classificada em:
  - **Definição** (*def* ou d): ocorre quando uma variável recebe um valor.

```
a = 1
```

  - **Uso**: ocorre quando a variável é referenciada e tem o seu valor consultado. Um uso pode ser:
    - **Computacional** (*c-uso* ou uc): a variável é utilizada em uma computação.

```
b = a * 2
```

    - **Predicativo** (*p-uso* ou up): a variável é utilizada em uma condição.

```
if (a > 0)
```

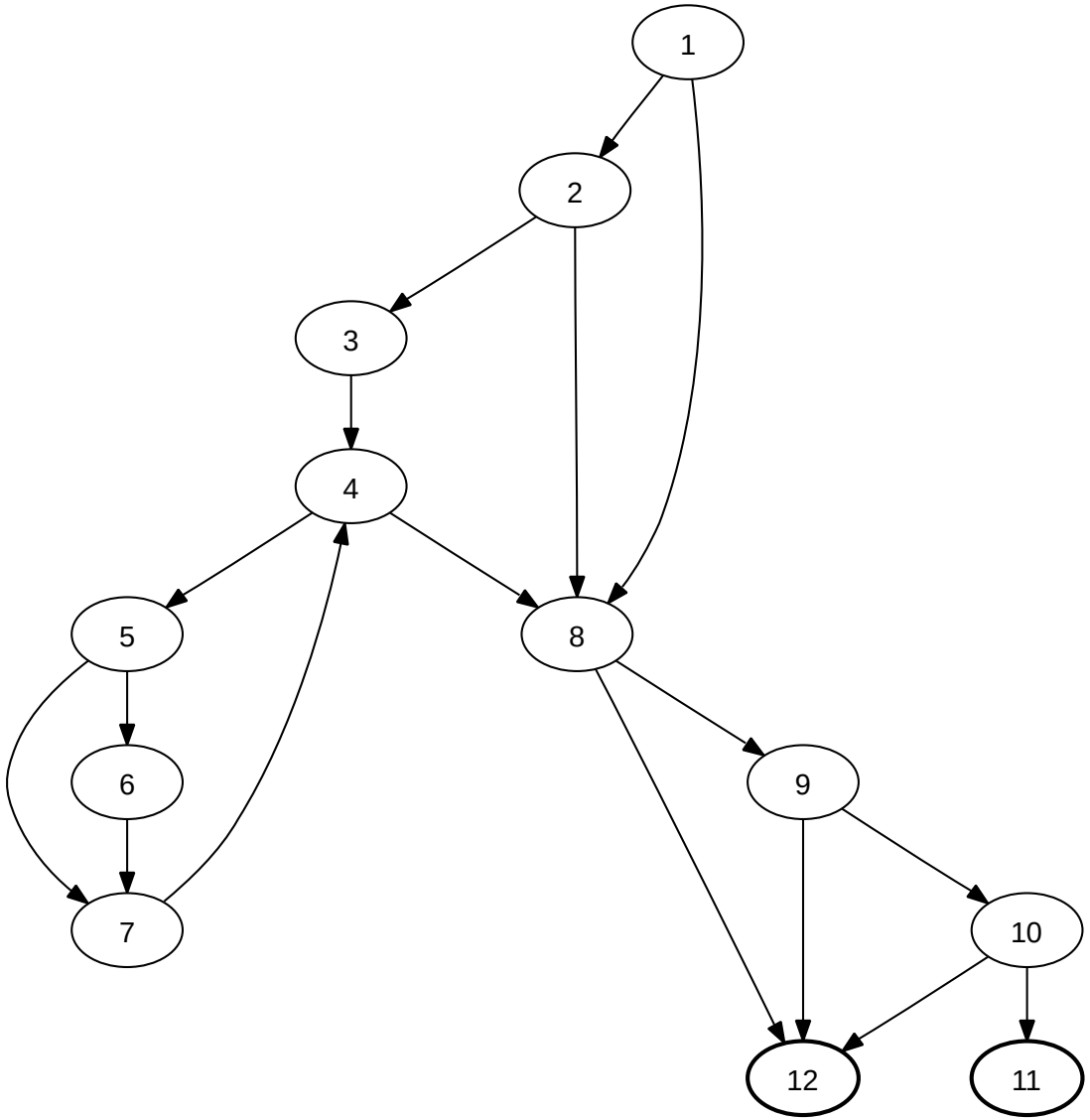
Conceitos

- **c-uso global** quando não existe *def* da variável no bloco em que ocorre o *c-uso*.
- **caminho livre de definição** em relação a uma variável x (c.d.l.(x)): caminho entre nós A e B, sendo que x pode ser definida em A, mas não existe nenhuma outra definição de x entre A e B.
- **def global**: quando a *def* de uma variável x em um bloco A é usada em um bloco B (ou em um predicado)
- um **du-caminho** no grafo def-uso em relação a uma variável x do nó i ao nó j é um caminho simples (sem repetição de nós exceto possivelmente o primeiro e o último), livre de definição em relação a x que possui uma definição de x no nó i e um uso no nó j.

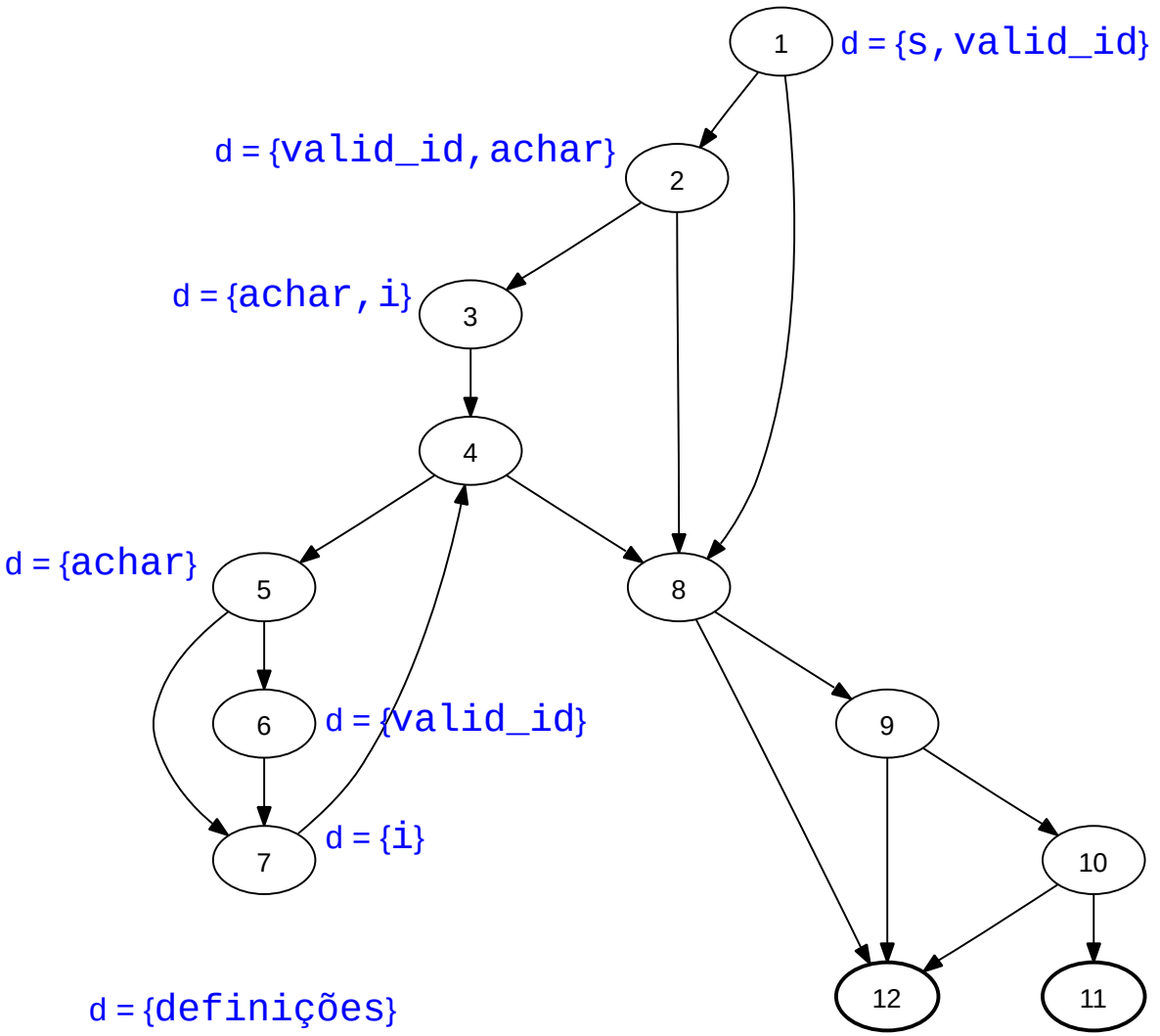
Exemplo – Identifier

```
public boolean validateIdentifier(String s) {
       char achar;
/*01*/ boolean valid_id = false;
/*01*/ if (s.length() > 0) {
/*02*/    achar = s.charAt(0);
/*02*/    valid_id = valid_s(achar);
/*02*/    if (s.length() > 1) {
/*03*/       achar = s.charAt(1);
/*03*/       int i = 1;
/*04*/       while (i < s.length() - 1) {
/*05*/          achar = s.charAt(i);
/*05*/          if (!valid_f(achar))
/*06*/             valid_id = false;
/*07*/          i++;
              }
          }
       }      /*08*/          /*09*/              /*10*/
       if (valid_id && (s.length() >= 1) && (s.length() < 6))
/*11*/    return true;
       else
/*12*/    return false;
}
```

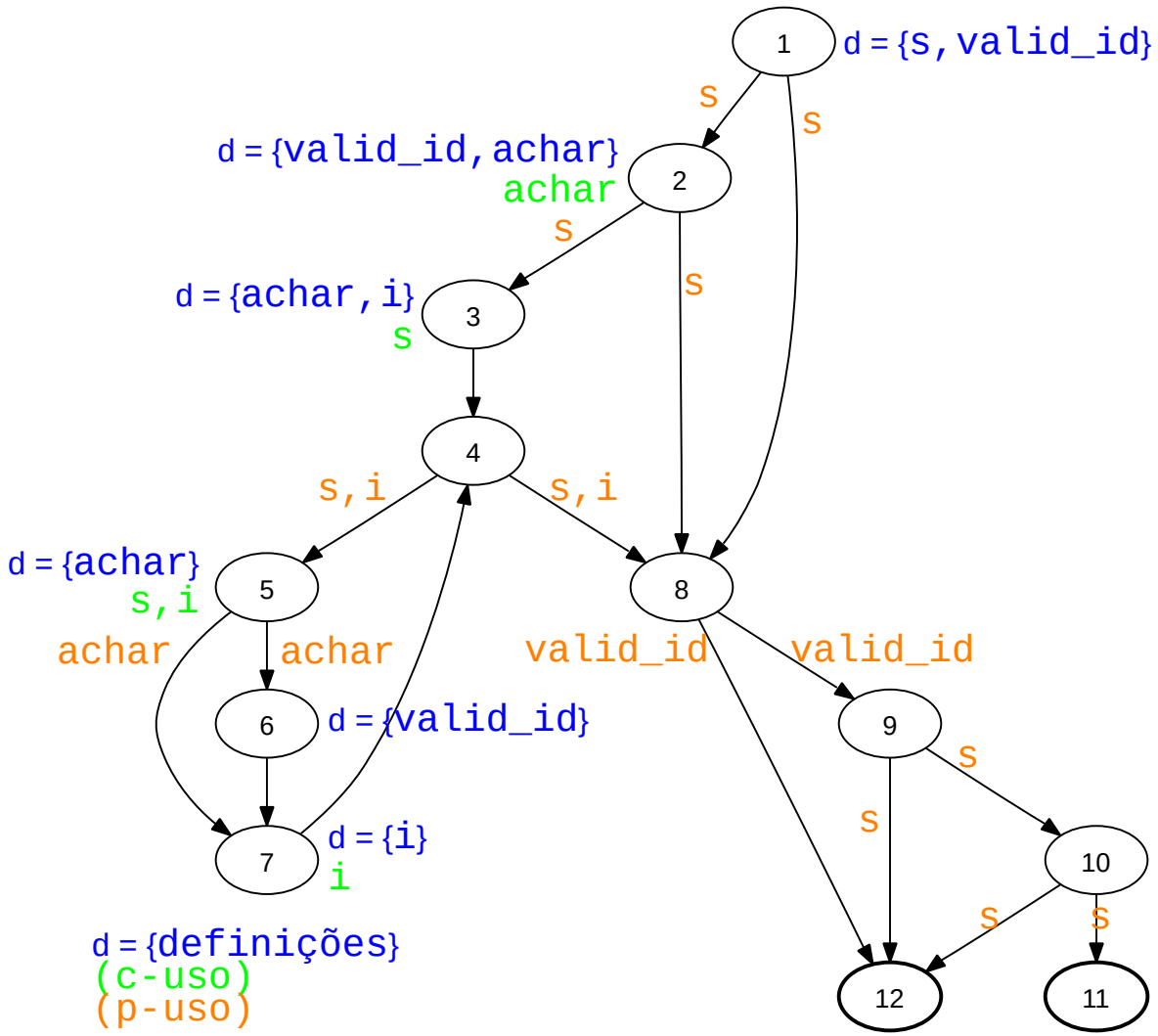
Identifier – GFC



Identifier – GFC + defs



Identifier – GFC + defs + usos



Critérios de Rapps e Weyuker (1982)

- Basseiam-se no Grafo Def-Uso para derivar os requisitos de testes.
- Objetivos: exercitar caminhos ligando definições globais a usos globais de variáveis do programa.
- Tipos:
  - todas as definições.
  - todos os p-usos.
  - todos os p-usos e alguns c-usos.
  - todos os c-usos e alguns p-usos.
  - todos os usos.

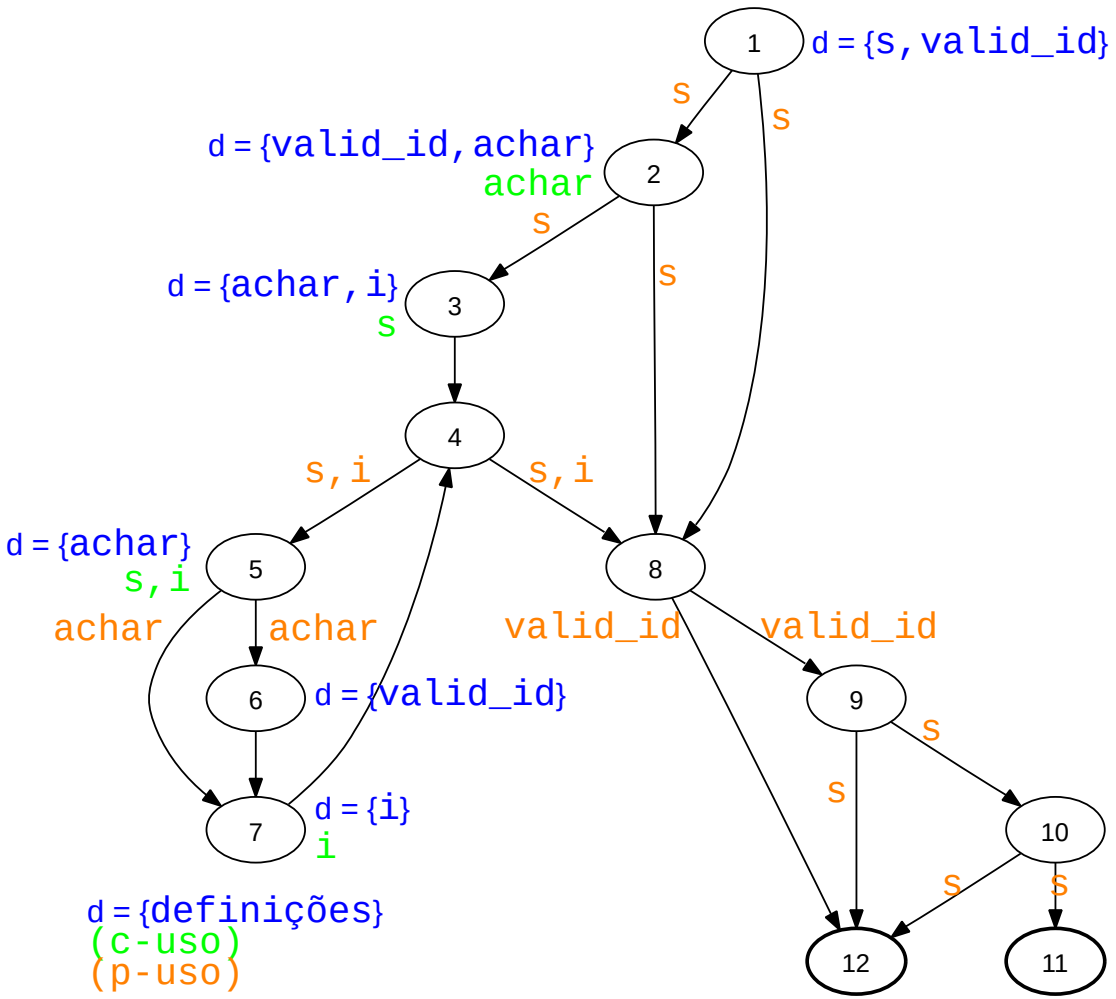
Critérios de Rapps e Weyuker (1982)

Critério	Descrição
todas-defs	para todas as definições de variáveis deve ser exercitado um caminho para um de seus usos.
todos-p-usos	para todas as definições de variáveis deve ser exercitado um caminho para todos os seus p-usos.
todos-p-usos-alguns-c-usos	para todas as definições de variáveis deve ser exercitado um caminho para todos os seus p-usos e alguns c-usos.
todos-c-usos-alguns-p-usos	para todas as definições de variáveis deve ser exercitado um caminho para todos os seus c-usos e para alguns p-usos.
todos-usos	para todas as definições de variáveis deve ser exercitado um caminho para todos os seus c-usos e para todos os seus p-usos.

Requisitos de teste

- Requisitos gerados pelos critérios de fluxo de dados:
- **Associações Definição-Uso**
  - Em geral uma tripla: $\langle$ *var*, *def*, *uso*$\rangle$, sendo:
    - *var* - variável para a qual a associação definição-uso foi estabelecida.
    - *def* - nó contendo a definição da variável *var*.
    - *uso* - nó/arco com uso computacional/predicativo de *var*.

Requisitos de teste



- Critérios de Fluxo de Dados:
  - Todas-Defs
  - Todas-Usos
    - $\langle$i, 3, 5$\rangle$
    - $\langle$i, 3, 7$\rangle$
    - $\langle$valid_id, 1, $(8, 9)$ $\rangle$
    - $\langle$valid_id, 1, $(8, 12)$ $\rangle$

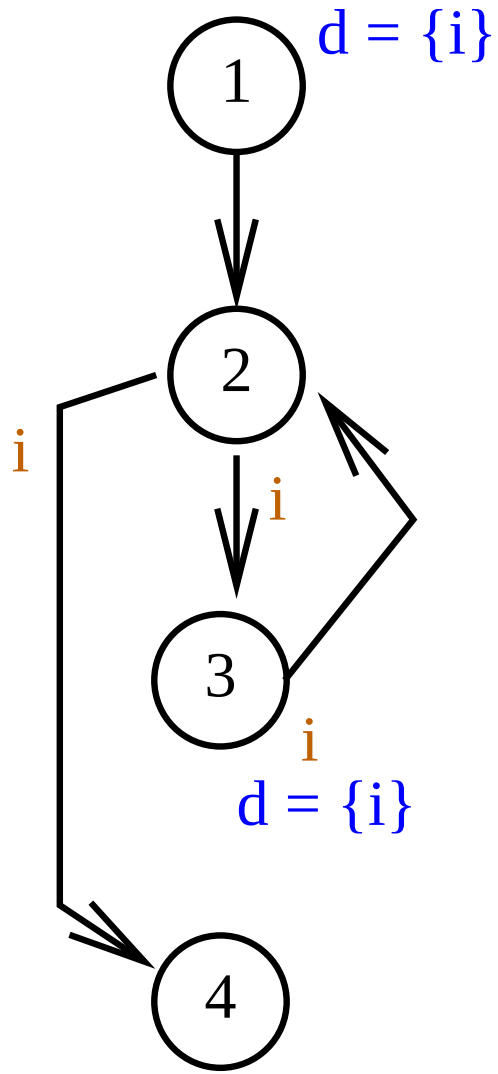
    ⋮

Comentários

- Complementares aos critérios de fluxo de controle.
- Mais comum no teste de unidade e de integração.
- Também requerem conhecimento do programa para serem aplicados.
- Exigem a análise de associações definição-uso quando a sua executabilidade.

Executabilidade

```
for (i = 0; i < 10; i++)
{
   printf("\%d", i);
}
return;
```



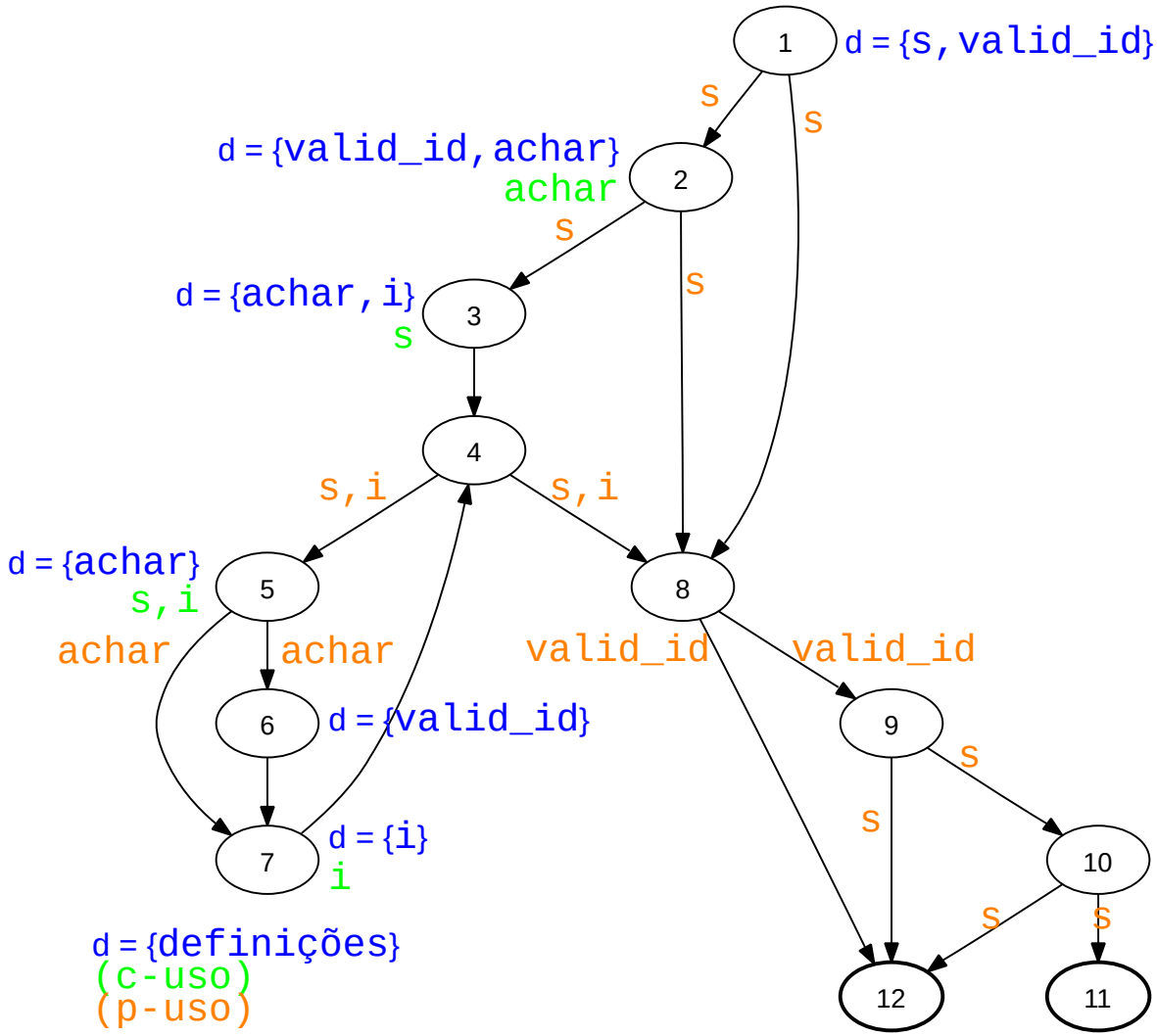
<i, 1, (2,4)>

A ferramenta JaBUTi

- `http://incubadora.fapesp.br/projects/jabuti/`
- `Jabuti-bin.zip` possui tudo que precisa para executar a ferramenta
- `java -cp Jabuti-bin.zip br.jabuti.gui.JabutiGUI`
- Programa “dot” (GraphViz – `http://www.graphviz.org/`) deve estar instalado no seu diretório padrão.

Exercício

Compute todas as associações definição-uso para o programa Identifier



Referências

RAPPS, S.; WEYUKER, E. J. Data flow analysis techniques for program test data selection. In: *6th International Conference on Software Engineering*, Tokio, Japan, 1982, p. 272–278.