MO615B - Implementação de Linguagens II
e
MC900A - Tópicos Especiais em Linguagem de Programação

Prof. Sandro Rigo

www.ic.unicamp.br/~sandro

Laços no CFG

Introdução

- Laços são estruturas no CFG que merecem atenção especial
  - Tipicamente, os programas gastam a maior parte do tempo de execução em laços.
- Otimizações que melhoram laços podem ter impacto significativo no desempenho.
- Vamos ver como detectar laços.

Dominadores

- Um nó *d* domina um nó *n* (*d* dom *n*) se
  - Todo caminho do nó inicial até *n* passa por *d*
- Propriedades
  - Reflexiva: todo nó domina ele mesmo
  - Transitiva: *a* dom *b* e *b* dom *c* => *a* dom *c*
  - Anti-simétrica: se *a* dom *b* e *b* dom *a* => *a* = *b*
- A entrada de um laço domina todos os nós do laço.

Dominadores

N = conjunto de nós

1 *dom* N

2 dom

3 dom

4 dom

5 dom

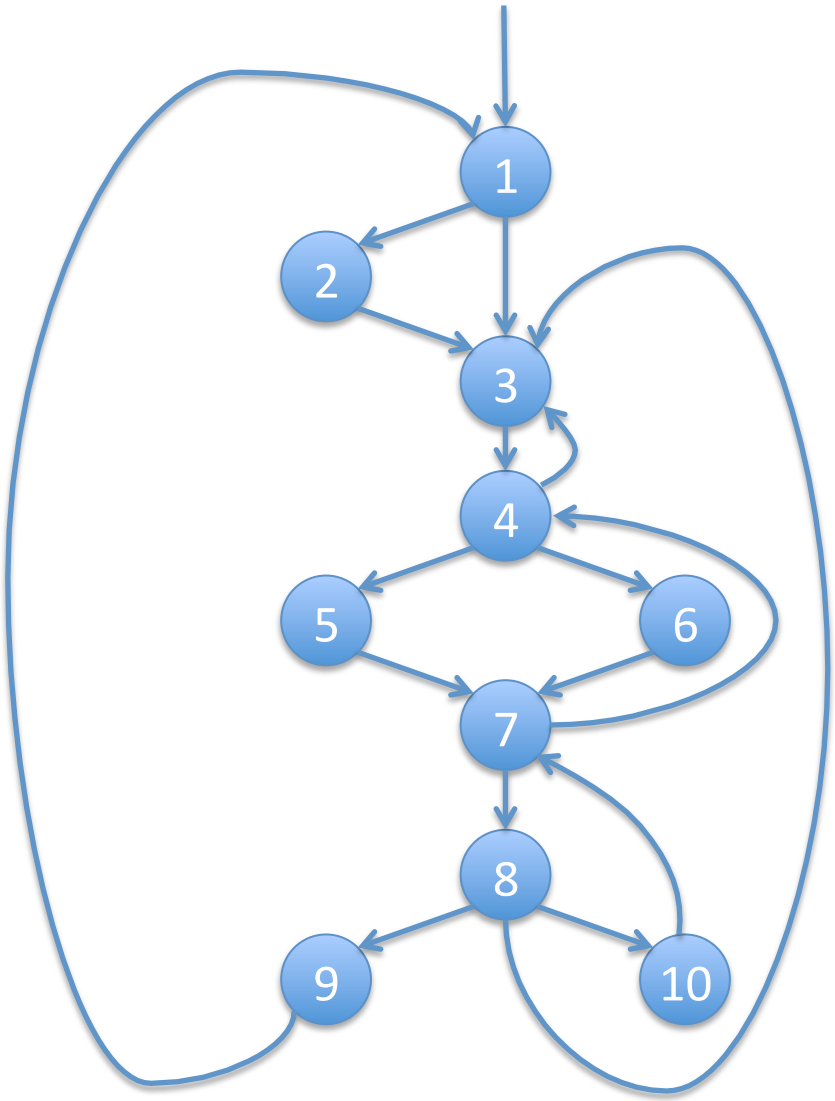
6 dom

7 dom

8 dom

9 dom

10 dom



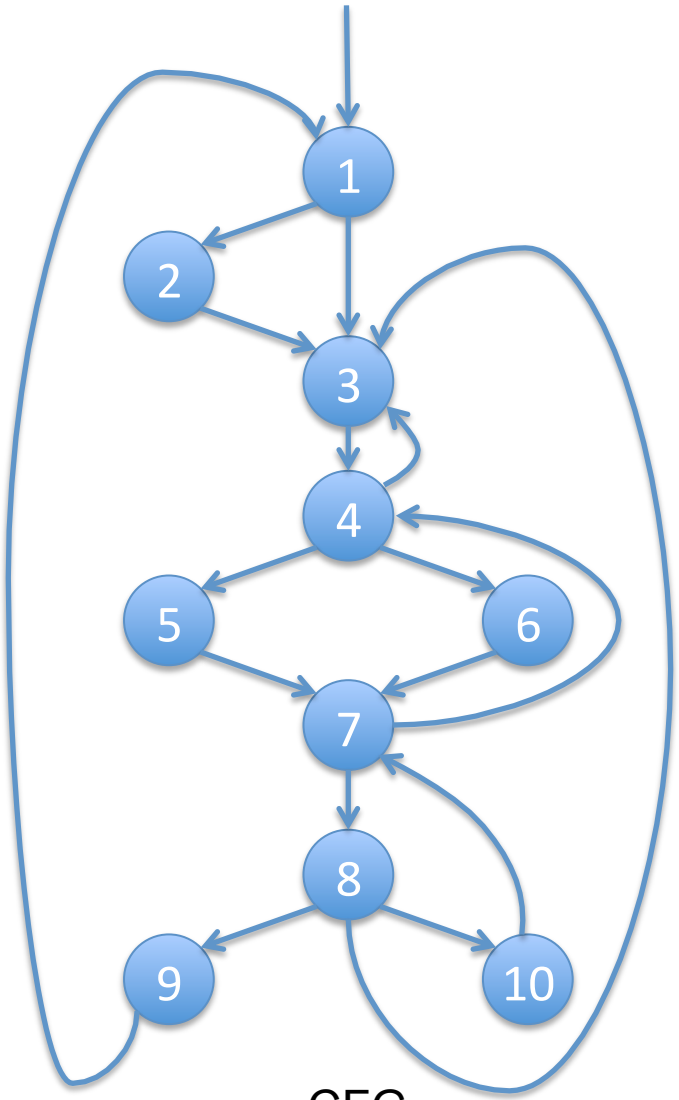
Dominador Imediato

- Para a ≠ b, *a idom b* se e somente se
  - a dom b
  - Não existe c tal que c ≠ a e c ≠ b para o qual
    - *a dom c* e *c dom b*
- Dominador imediato é único
- A informação de dominância imediata pode ser vista como uma árvore

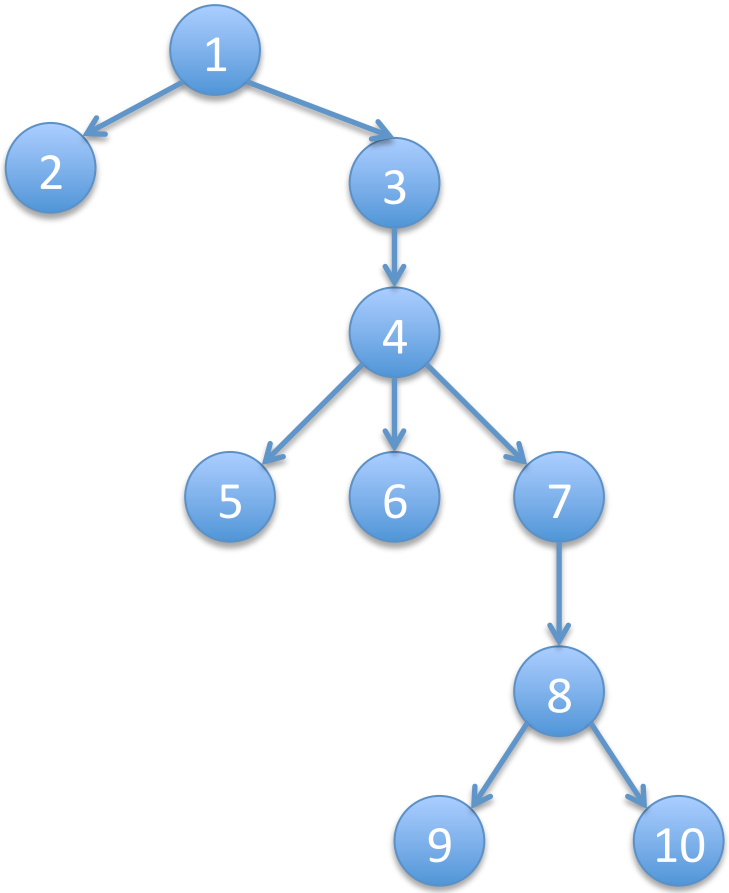
Árvore de Dominadores

- Útil para representar informação de dominância
- O nó inicial é a raiz
- Cada nó *d* domina seus descentes na árvore
- Cada nó é dominado imediatamente pelo seu pai
  - relação idom: arestas

Árvore de Dominadores



CFG



Árvore de Dominadores

Algoritmo para encontrar os dominadores

- Um nó a é dominado por ele mesmo
- Um nó a é dominado por b se todos os caminhos do início do programa até a passam por b
- Podemos usar uma DFA

Algoritmo para encontrar os dominadores

- DFA
  - Direção: *forwards*
  - Inicialização:
    - $OUT[ENTRY] = \{ENTRY\}$
    - $OUT[B] = \{B_1, B_2, ..., B_N\}$
  - Equações

$$OUT[B] = \{B\} \cup IN[B]$$

$$IN[B] = \bigcap_{P \in \Pr ed(B)} OUT[P]$$

Algoritmo para encontrar os dominadores

- Solução: O conjunto de nós que domina um nó B está em OUT[B].

Algoritmo – Dominância Imediata

- Tmp(i) = Domin(i) – i
- Para cada i
  - Verifica todo elemento de Tmp(i)
    - Possui um denominador fora ele mesmo?
      - Se sim, remove de Tmp(i)

Algoritmo – Dominância Imediata

```
procedure Idom_Comp(N,Domin,r) returns Node ⟶ Node
   N: in set of Node
   Domin: in Node ⟶ set of Node
   r: in Node
begin
   n, s, t: Node
   Tmp: Node ⟶ set of Node
   Idom: Node ⟶ Node
   for each n ∈ N do
      Tmp(n) := Domin(n) - {n}
   od
*  for each n ∈ N - {r} do
      for each s ∈ Tmp(n) do
         for each t ∈ Tmp(n) - {s} do
            if t ∈ Tmp(s) then
               Tmp(n) -= {t}
            fi
         od
      od
   od
   for each n ∈ N - {r} do
      Idom(n) := ◆Tmp(n)
   od
   return Idom
end    || Idom_Comp
```

Fonte:
Muchnick

Laços Naturais

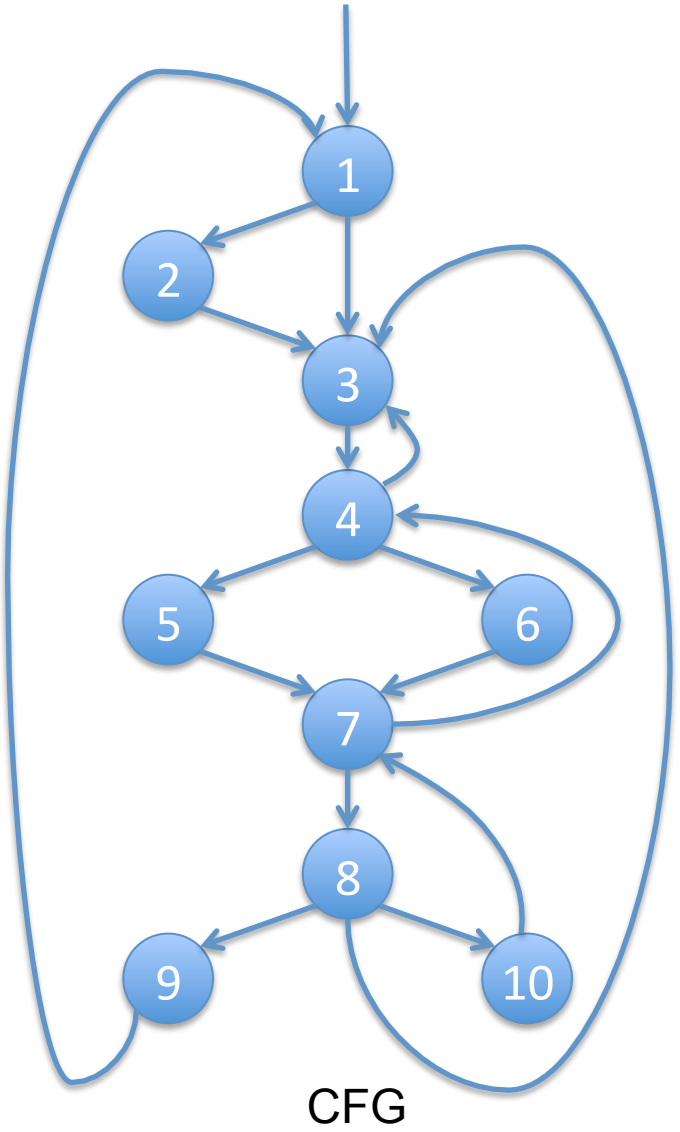
- Propriedades
  - Ter um único ponto de entrada
    - *Header* – (Domina todos os outros nós)
  - Ter pelo menos um caminho para iterar de volta ao *header*
- Busca por laços
  - Identificar as *back edges*
    - Arestas cuja cabeça domina sua cauda
      - a → b
        - » a é a cauda
        - » b é a cabeça

Laços Naturais

- Definição
  - Dada uma *back edge* $n \rightarrow d$
    - Seu laço natural é:
      - d mais o conjunto de nós que podem alcançar n sem passar por d
      - d é o *header* do laço

Laços Naturais

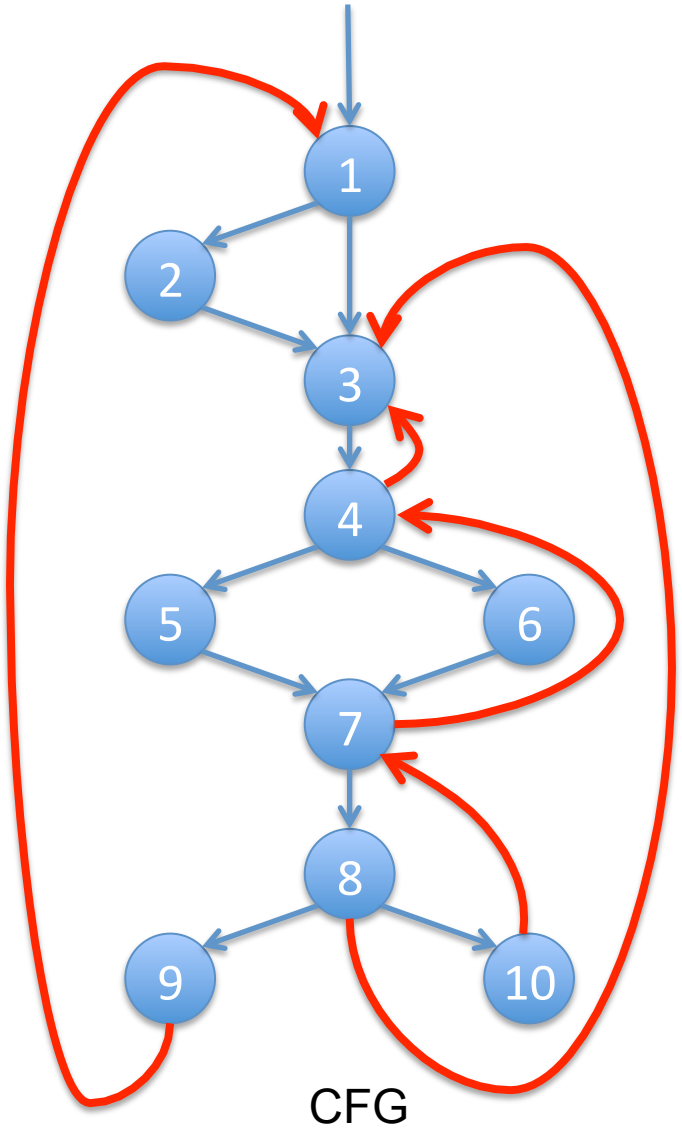
- Quais são as back edges do CFG?



CFG

Laços Naturais

- Quais os laços naturais das arestas abaixo?
  - $10 \rightarrow 7$
  - $9 \rightarrow 1$
- {4,5,6,7} é um laço natural?

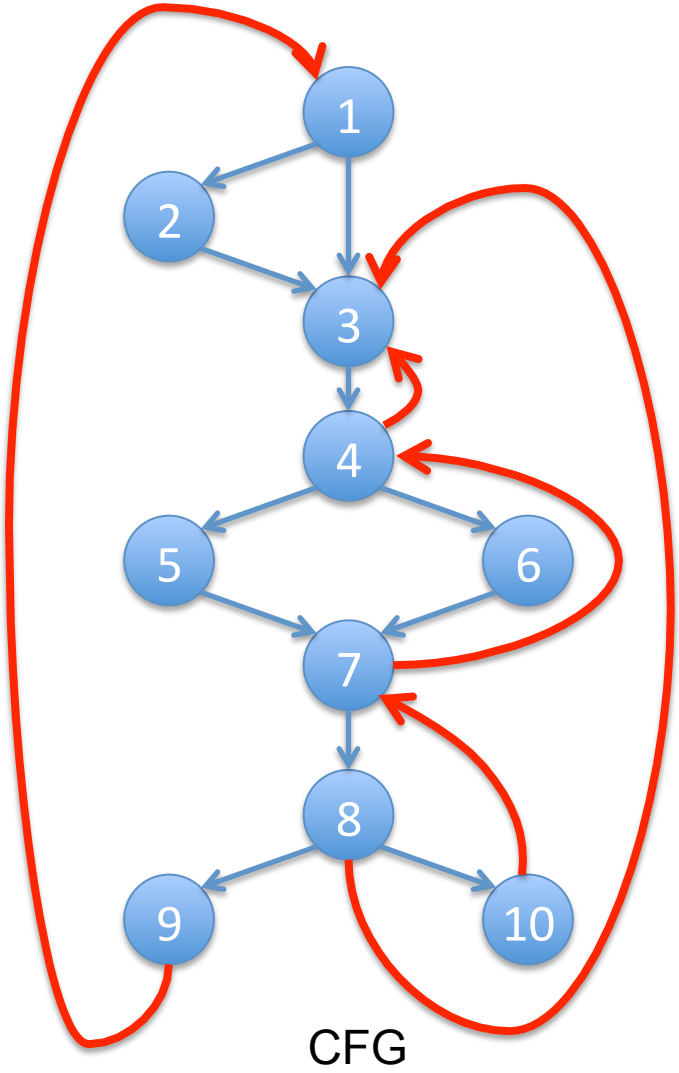


Construindo Laços Naturais

- Encontrar o laço natural da aresta n -> d
  - Laço := {n, d}
  - Anote d como visitado
  - Faça uma busca reversa no CFG a partir de n, adicionando todos os nós visitados no Laço.
- Este processo encontra todos os nós que alcançam *n* sem passar por *d*.

Laços Naturais

- Quais os laços do CFG?



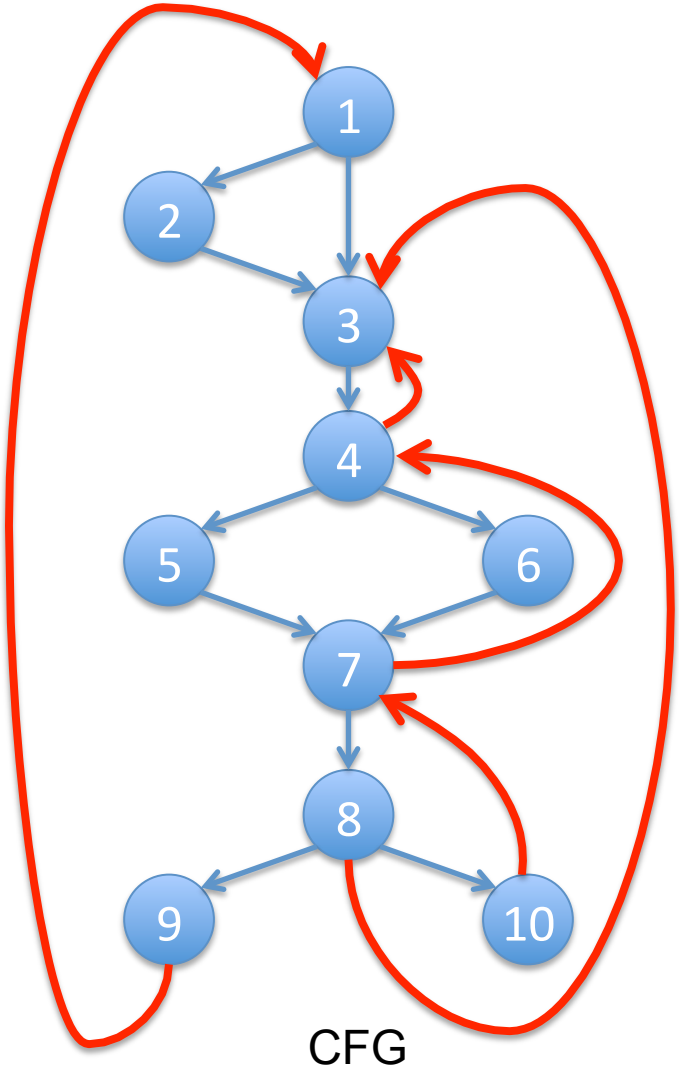
CFG

Laços Internos

- Adotaremos os laços naturais como os únicos laços que trataremos
  - Dois laços que não têm o mesmo header
    - são disjuntos ou
    - um é inteiramente contido no outro
- Se ignorarmos laços com o mesmo *header*
  - Inner loops são aqueles que não contém outros laços

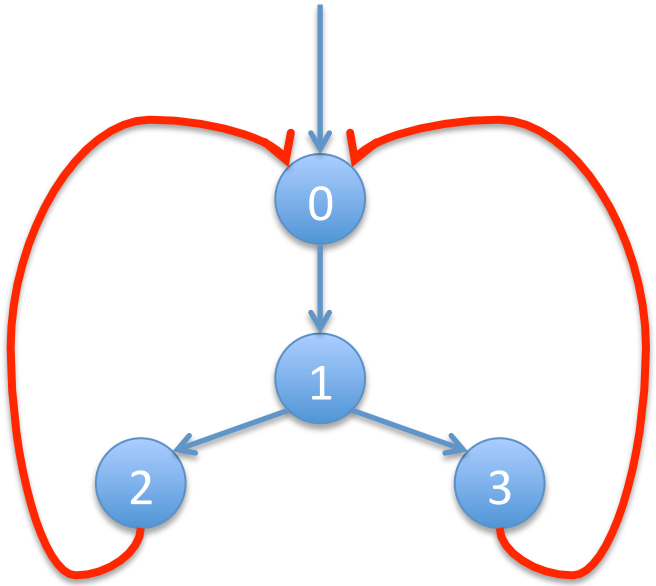
Laços Naturais

- Os laços mais internos são chamados *inner loops*.
- Quais os *inner loops do CFG*?



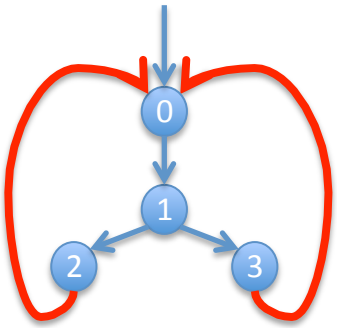
Laços com o mesmo *header*

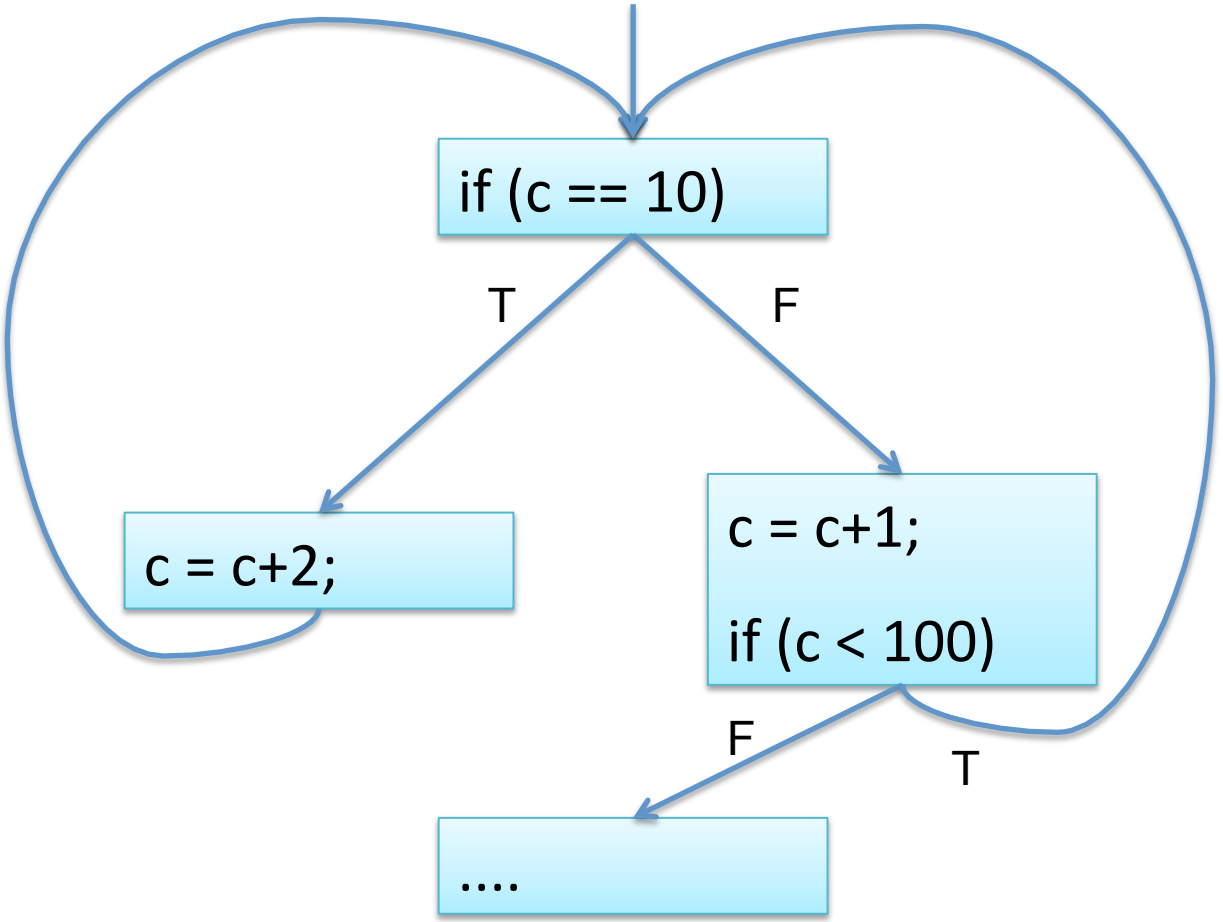
- Temos 1 laço ou 2 laços?



Laços com o mesmo *header*

```
do {
  if (c == 10) {
     c = c+2;
     continue;
  }
  c = c+1;
} while(c < 100);
```





Laços com o mesmo *header*

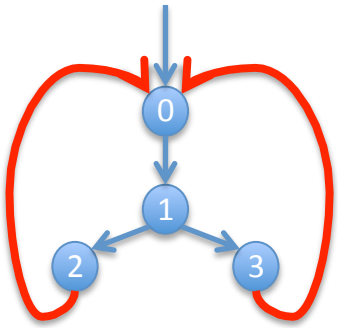
```
do {
  do {
     if(c != 10)
        break;
     c = c+2;
  } while(1);
  c = c+1;
} while(c < 100);
```

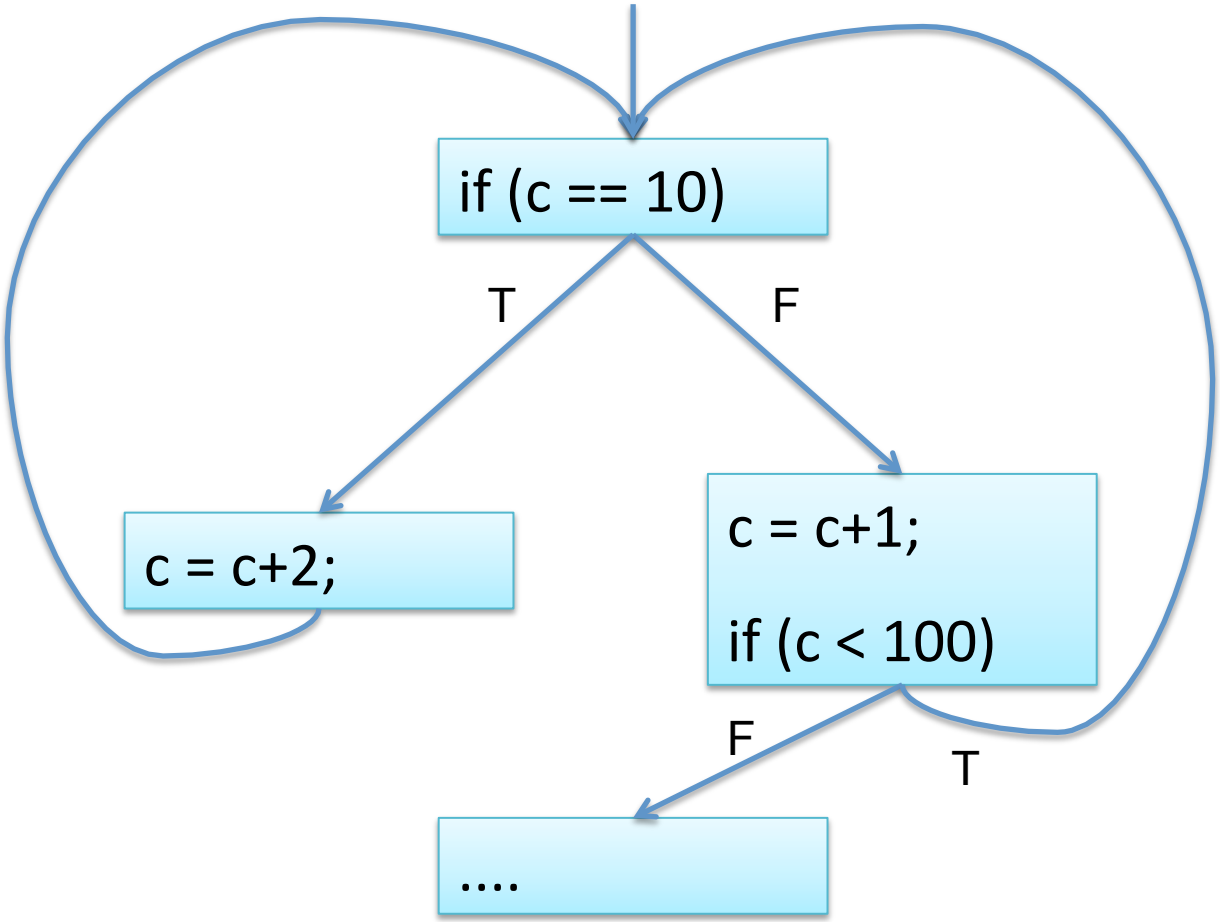
São iguais?

```
do {
  if (c == 10) {
     c = c+2;
     continue;
  }
  c = c+1;
} while(c < 100);
```

Laços com o mesmo *header*

```
do {
  do {
     if(c != 10)
        break;
     c = c+2;
  } while(1);
  c = c+1;
} while(c < 100);
```

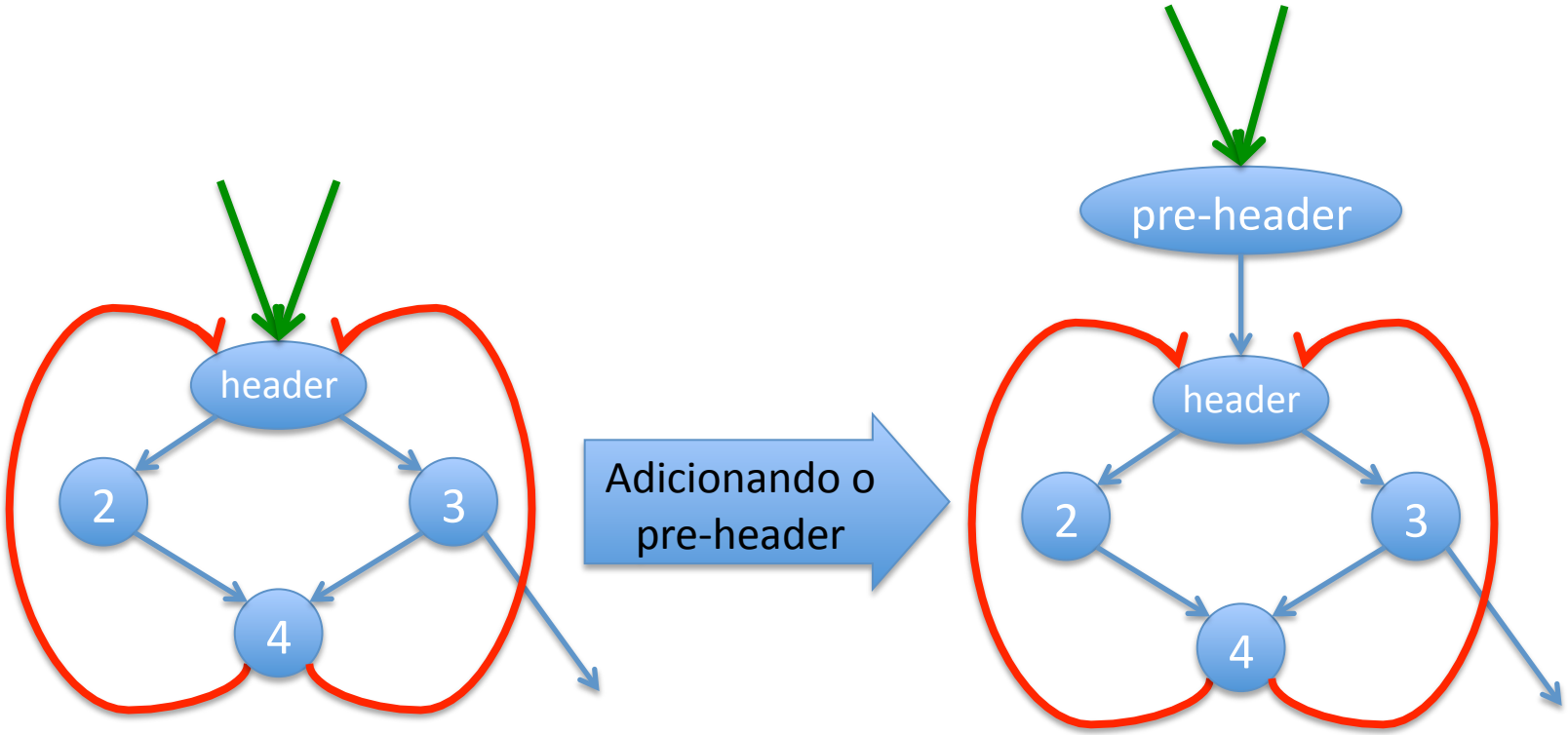




Pre-Headers

- Várias transformações necessitam mover sentenças para fora do laço
  - Antes do *header*
- Começamos o tratamento do laço L criando um nó especial
  - *Pre-header*
    - Tem somente o *header* como sucessor
    - Todas as arestas externas de L para o *header* são transferidas para o pre-header
    - As internas permanecem inalteradas

Pre-Headers



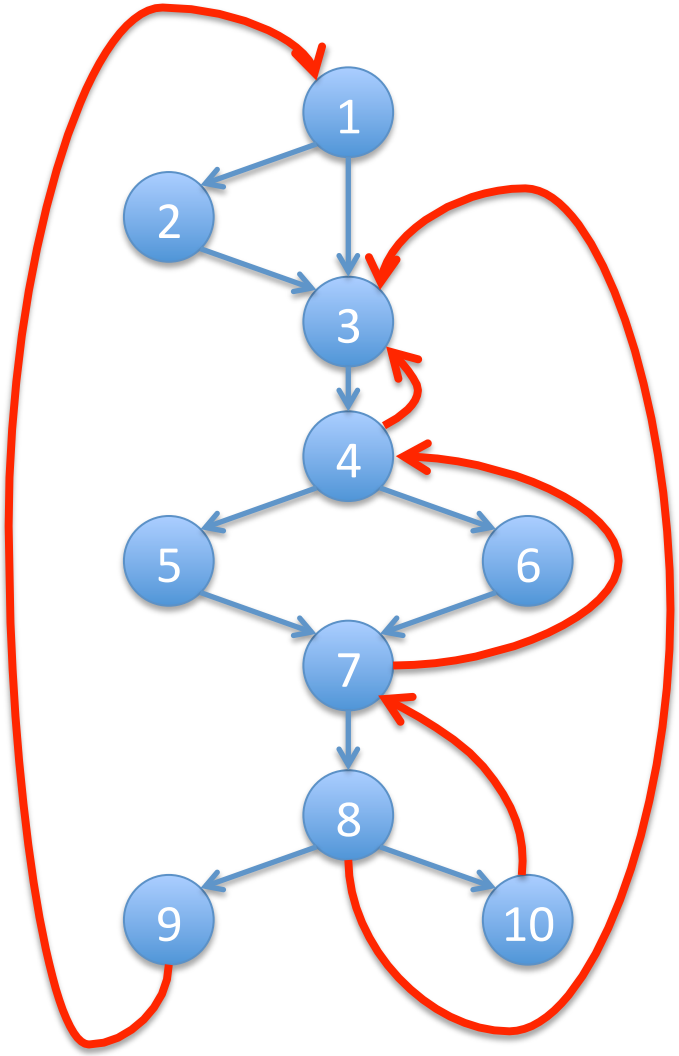
CFG Redutíveis

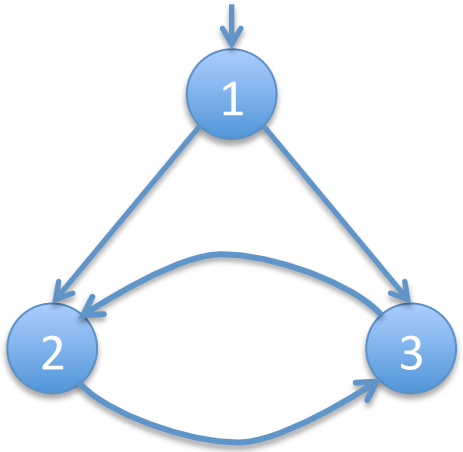
- Propriedade
  - Não existem saltos para dentro de laços vindo de nós externos ao laço
    - Garante que a entrada de laços é única
- Definição
  - G é redutível ⇔ as arestas podem ser particionadas em dois conjuntos disjuntos
    - *forward* e *back edges*
- Maioria dos CFGs na prática caem nessa classe
  - Dado que temos um programa estruturado
  - Mesmo programas com goto tendem a cair nesta classe.

CFG Redutíveis

- Propriedade
  1. as arestas *forward* formam um grafo acíclico no qual qualquer nó pode ser alcançado a partir do nó inicial de G (busca em profundidade).
  2. todas as *back edges* tem suas cabeças dominando suas caudas.
- Verificando
  – Verificamos a propriedade 1 com uma busca DFS e informações de dominância. (Dragão 9.6.3)

CFG Redutíveis





CFG Redutíveis

- Importância
  - Precisamos apenas verificar laços naturais de *back edges* para termos todos os laços do CFG.
- CFG não redutíveis, como o da figura anterior, tornam transformações não diretamente aplicáveis.

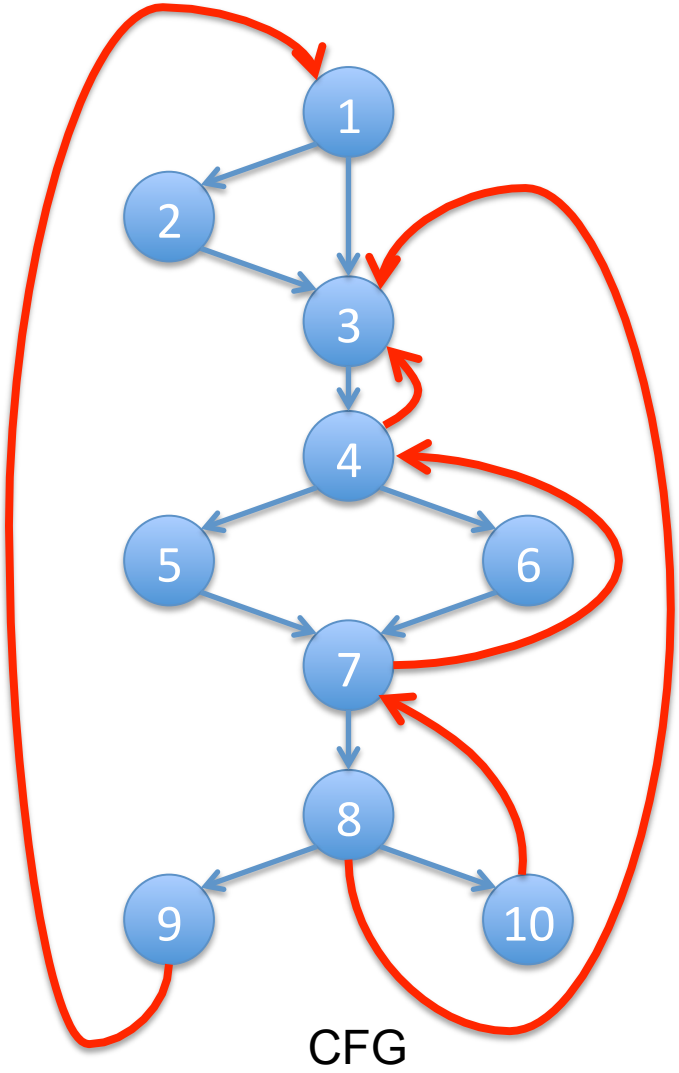
Laços Aninhados

- Seja A e B laços com *headers* a e b
- a ≠ b e b está em A
  - Nós em B são um subconjunto próprio dos nós em A
  - B está aninhado em A
- Aninhamento pode ser representado como uma árvore de laços no programa
- A raiz é todo o corpo do procedimento considerado com um pseudo-laço

Laços Aninhados

- Método no livro do Appel:
  - Calcule os dominadores no CFG
  - Construa a árvore de dominadores
  - Encontre os laços naturais e seus nós *header*
  - Para cada *header* h
    - Una todos os laços naturais de h em um único laço[h]
  - Construa a árvore de headers dos laços, tal que h1 está acima de h2 se h2 está em laço[h1]
- As folhas serão os laços mais aninhados

Árvore de Laços



CFG