Análise de longevidade

Alexsandro Santos Soares
prof.asoares@gmail.com

Sumário

Sumário

Sumário

Alocação e atribuição de registradores

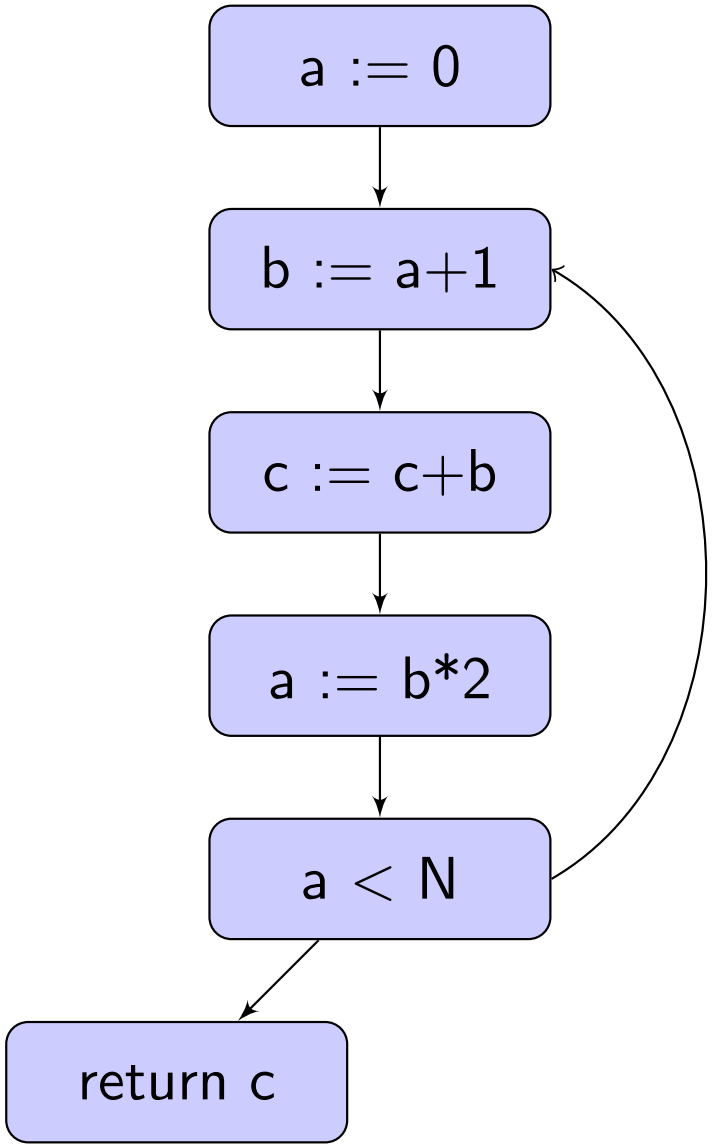
- O uso de registradores frequentemente é subdivido em dois problemas:
  - Decidir o conjunto de variáveis que residirão nos registradores em cada ponto do programa;
  - Determinar um registrador específico em que uma variável residirá.
- Um passo útil para a alocação de registradores é a análise da longevidade das variáveis.

Análise da longevidade

- Máquinas reais tem um número finito de registradores.
- Dois valores temporários podem ocupar o mesmo registrador se não estão *em uso* ao mesmo tempo.
  - Muitos temporários podem caber em poucos registradores;
  - Os que não couberem vão para a memória.
- Uma variável está *viva* se o seu conteúdo pode ser usado no futuro.
- A tarefa de determinar quais variáveis estão simultaneamente vivas é chamada da *análise de longevidade*.

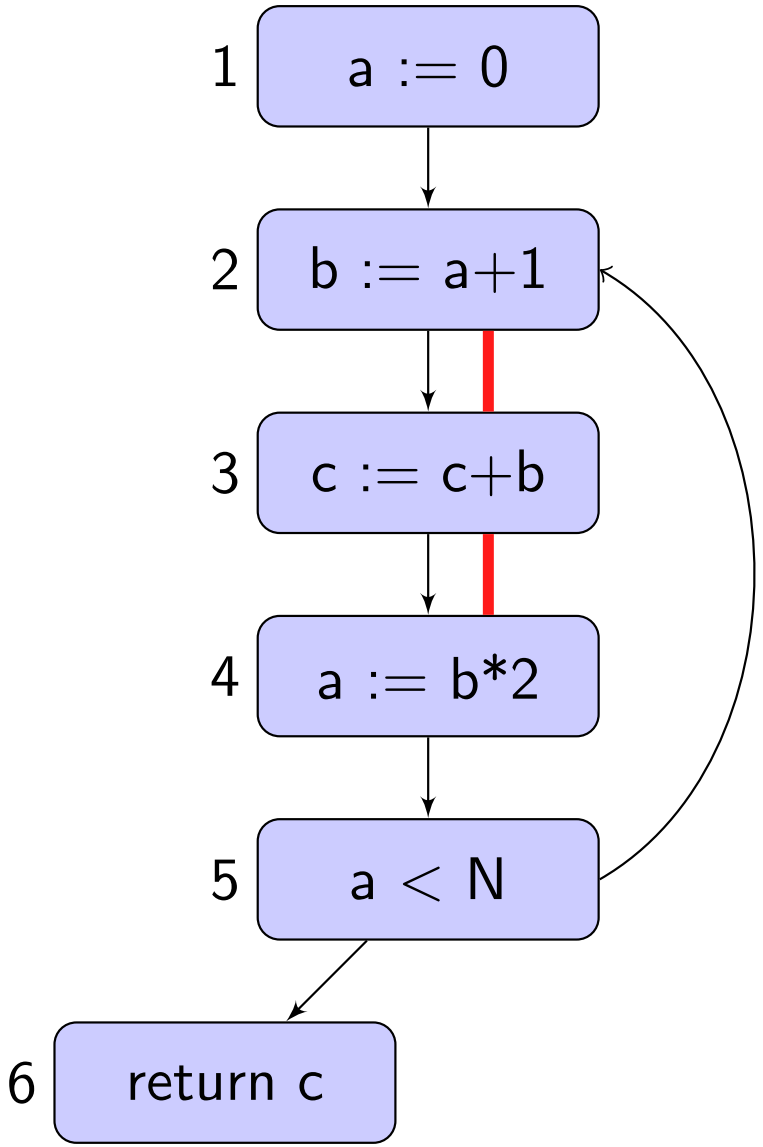
Grafo de fluxo de Controle (CFG)

$$
\begin{array}{ll}
 & a \leftarrow 0 \\
L_1 : & b \leftarrow a + 1 \\
 & c \leftarrow c + b \\
 & a \leftarrow b * 2 \\
 & \text{if } a < N \text{ goto } L_1 \\
 & \text{return } c
\end{array}
$$



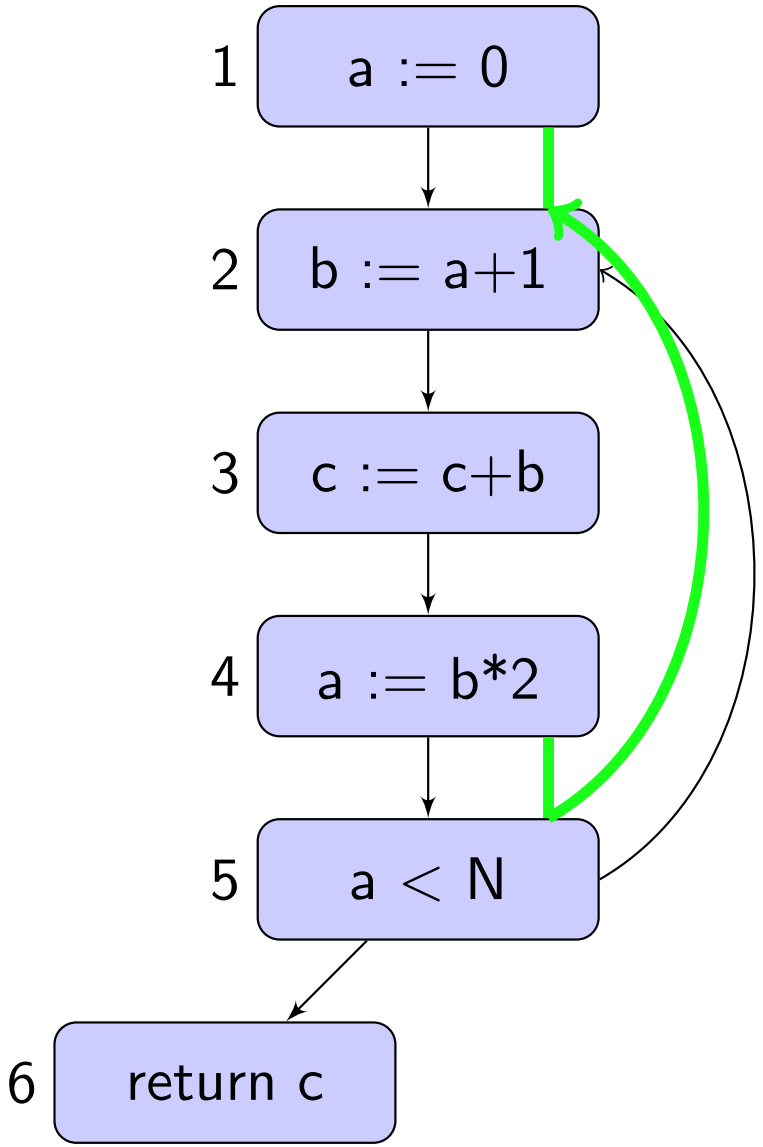
Exemplo de análise de longevidade: b

- *b* é usada em 4
  - precisa estar viva na aresta $3 \rightarrow 4$
- *b* não é definida (recebe um valor) no nó 3
  - Logo, deve estar viva viva na aresta $2 \rightarrow 3$
- *b* é definida no nó 2
  - Logo, *b* está morta na aresta $1 \rightarrow 2$
  - Seu valor nesse ponto não será mais útil
- Tempo de vida de *b*:
  - $\{\ 2 \rightarrow 3,\ 3 \rightarrow 4\ \}$



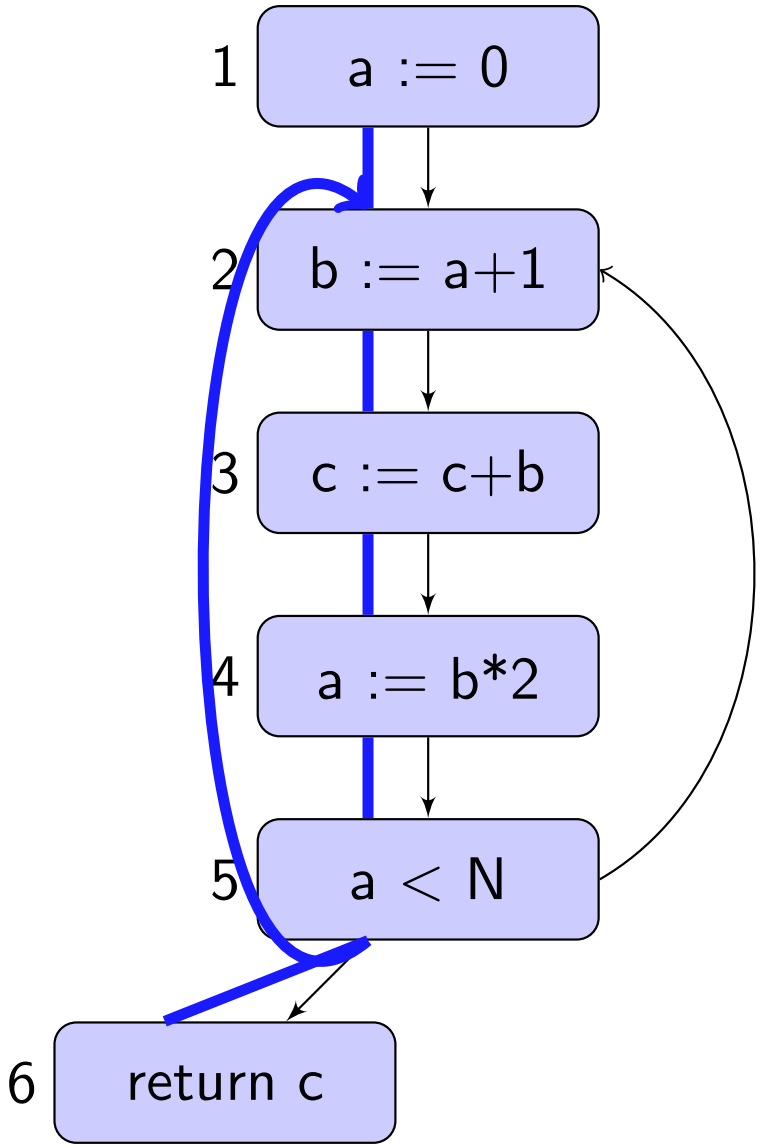
Exemplo de análise de longevidade: a

- Qual o tempo de vida de *a*?
  - $\{ 1 \rightarrow 2, 4 \rightarrow 5, 5 \rightarrow 2 \}$



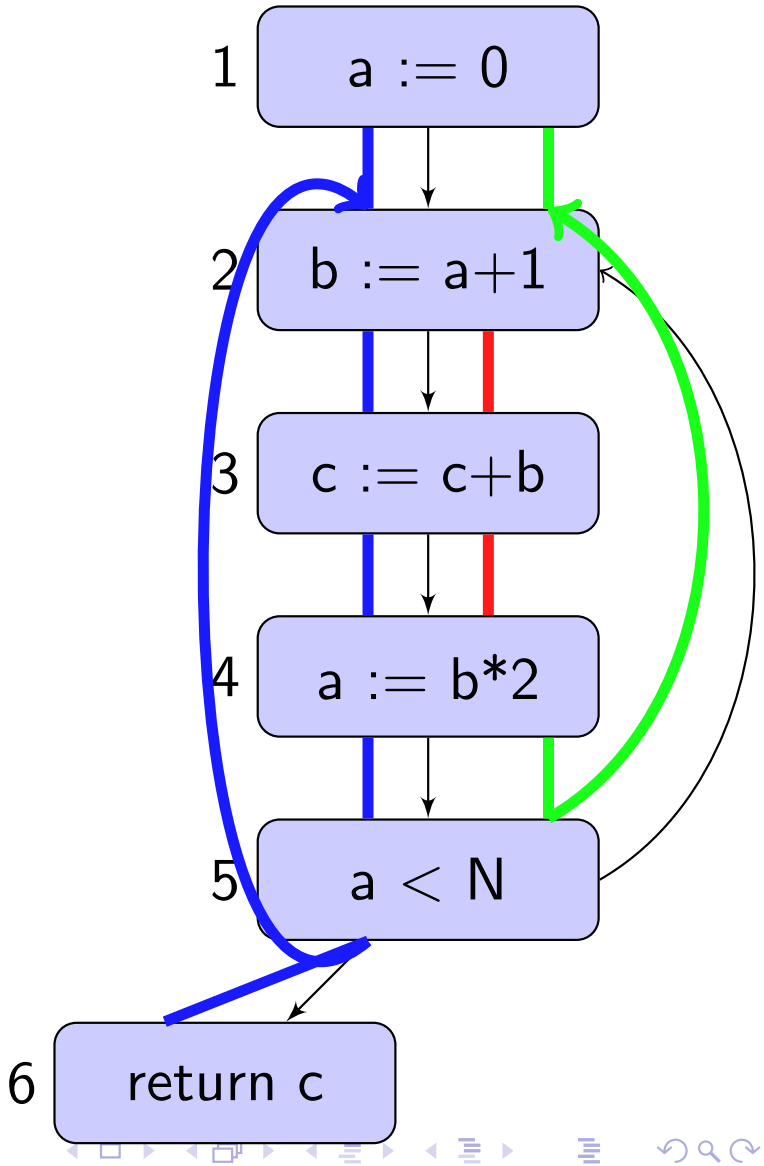
Exemplo de análise de longevidade: c

- Qual o tempo de vida de c?
  - $\{ 1 \to 2, 2 \to 3, 3 \to 4, 4 \to 5, 5 \to 6, 5 \to 2 \}$



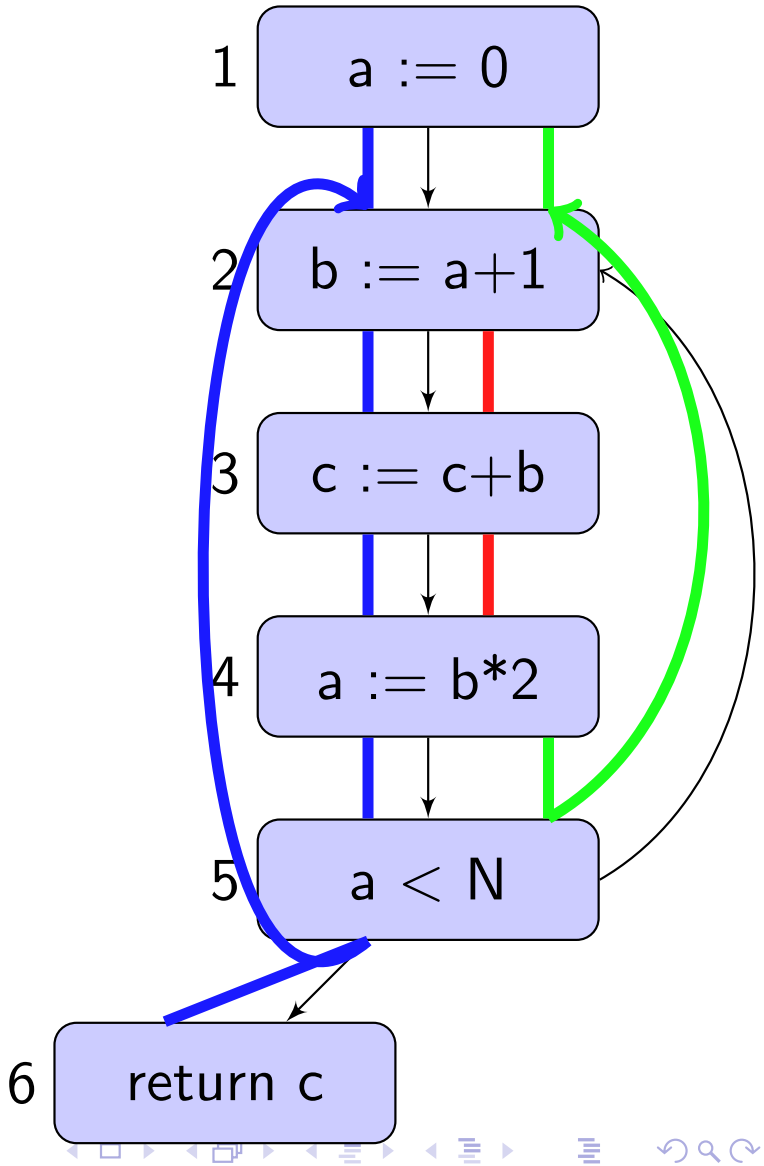
Exemplo de análise de longevidade: quantidade de registradores

- Qual é o número mínimo de registradores para manter as variáveis a, b e c?



Exemplo de análise de longevidade: quantidade de registradores

- Qual é o número mínimo de registradores para manter as variáveis *a*, *b* e *c*?
  - Somente dois, pois *a* e *b* nunca estão vivas simultaneamente.



Análise de longevidade: Terminologia

- Succ[n]: conjunto de nós sucessores a n
- Pred[n]: conjunto de predecessores de n
- Out-edges: saem para os sucessores
- In-edges: chegam dos predecessores
- Uma atribuição a uma variável *define* a mesma
- Uma ocorrência do lado direito de uma expressão é um *uso* da variável
- *Def de uma variável* é o conjunto de nós do grafo que a definem
- *Def de um nó* é o conjunto de variáveis que ele define
- Analogamente para *use*
- Uma variável v está *viva* em uma aresta se existe uma caminho dirigido desta aresta até um uso de v, que não passa por alguma definição de v
- Live-in: v é live-in em um nó n se v está viva em alguma in-edge de n
- Live-out: v é live-out em n se v está viva em alguma out-edge de n

Cálculo da longevidade

1. Se v está em use[n], então v é live-in em n.
2. Se v é live-in no nó n, então ela é live- out para todo m em pred[n].
3. Se v é live-out no nó n, e não está em def[n], então v é também live-in em n.

$$
\begin{aligned}
in[n] &= use[n] \cup (out[n] - def[n]) \\
out[n] &= \bigcup_{s \in succ[n]} in[s]
\end{aligned}
$$

Algoritmo

```
para cada n faça
  in[n]  := {}
  out[n] := {}
  repita
    para cada n faça
      in'[n]  := in[n]
      out'[n] := out[n]
      in[n]   := use[n] U (out[n] - def[n])
      out[n]  := U(s em succ[n]) in[s]
    fim-para
  até que in'[n] = in[n] e out'[n] = out[n] para todo n
fim-para
```

Observações sobre o algoritmo

- O fluxo da análise deve seguir o fluxo da longevidade: de trás para frente
- Complexidade:
  - Pior caso: $O(N^4)$
  - Usando-se uma ordenação dos nós via uma busca em profundidade, na prática executa tipicamente entre $O(N)$ e $O(N^2)$
- É conservador:
  - Se uma variável pode estar viva em algum nó n, ela estará no out[n]
  - Pode haver alguma variável em out[n] que na verdade não seja realmente usada adiante
- Deve ser dessa maneira para prevenir o compilador de tornar o programa errado!

Blocos Básicos

- Nós com apenas um predecessor e um sucessor podem ser unidos
- Teremos um grafo com menos nós
- Cada nó é um bloco básico
- Os algoritmos funcionam mais rapidamente

Grafos de interferência

- A informação de longevidade é usada para otimização da alocação de registradores.
- Uma *interferência* ocorre quando a e b não podem ocupar o mesmo registrador.
  - tempo de vida com sobreposição.
- A informação sobre interferência pode ser expressa como uma matriz ou por um grafo com um vértice para cada variável e arestas conectando variáveis que interferem uma com a outra.

Referências