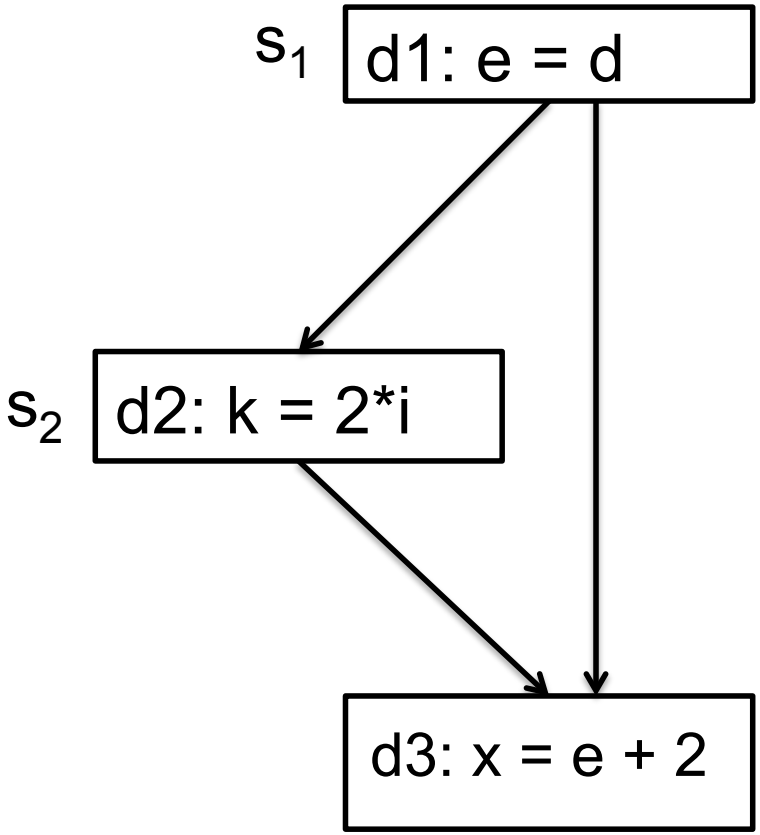
Otimizações

Guido Araújo
guido@ic.unicamp.br

Copy Propagation

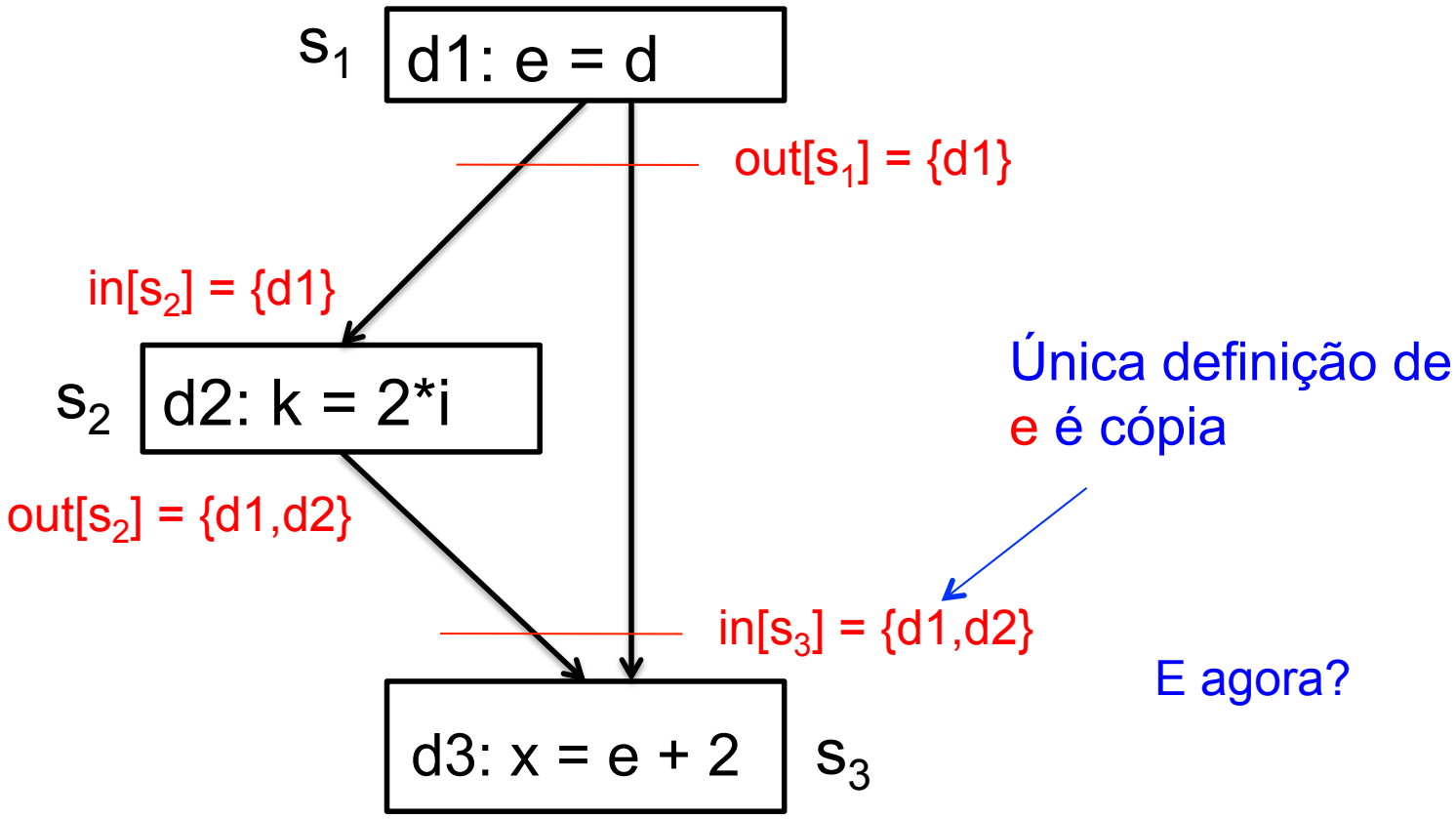
- Elimina cópias desnecessárias
- Seja d: t ← z
- Seja n: y ← t op x
- Quando t será uma cópia em n?
  - Neste caso, podemos reescrever n da forma
    - n: y ← z op x

Copy Propagation

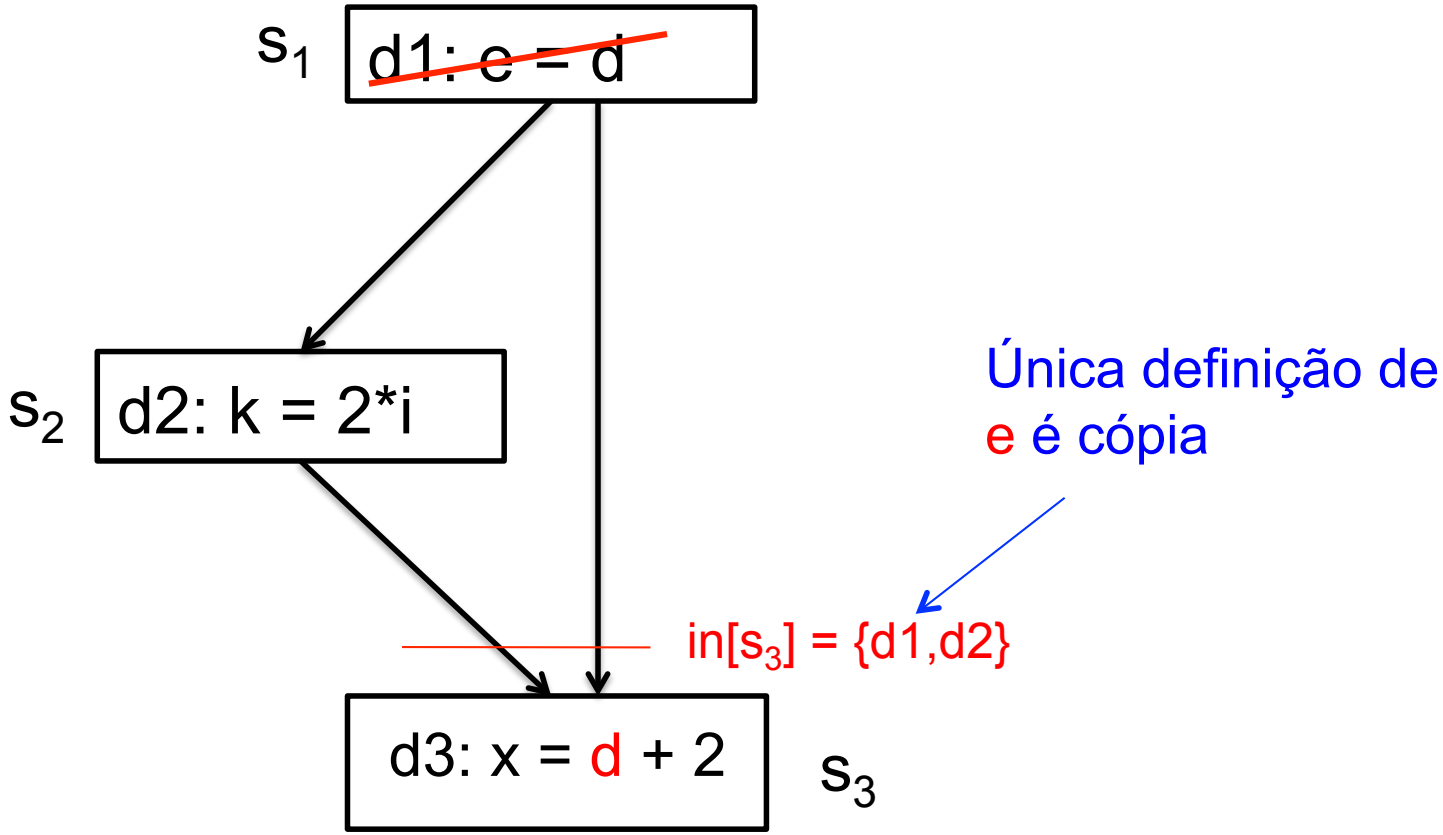


Que análise é necessária?

Reaching Definitions



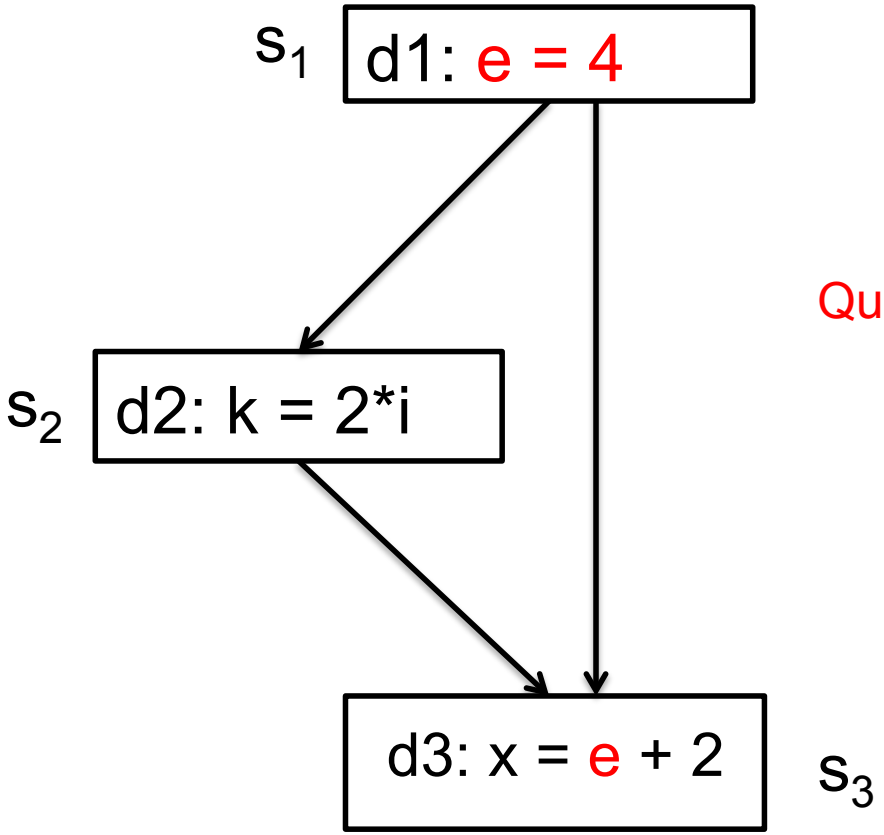
Copy Propagation



Constant Folding

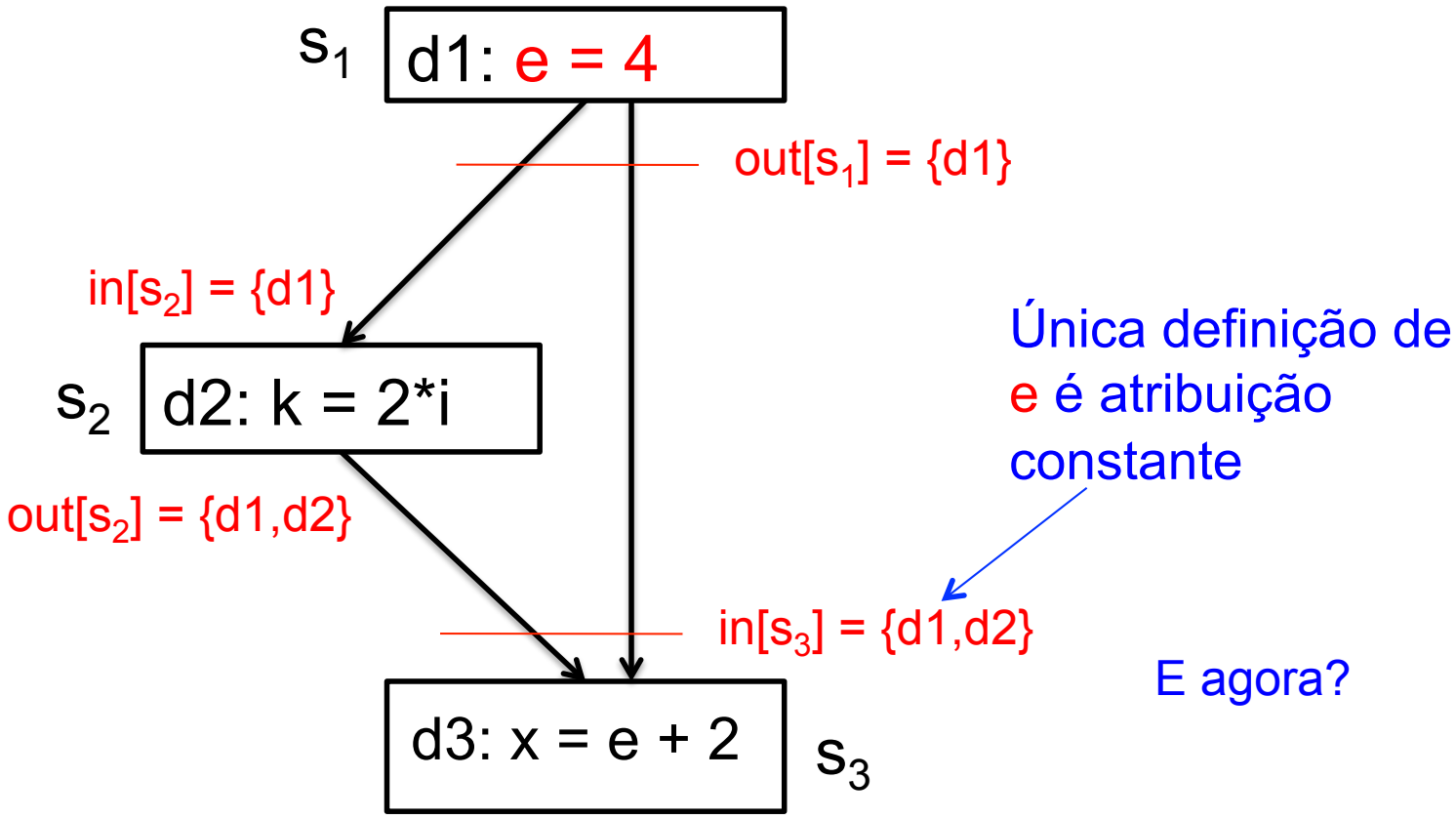
- Seja d: t ← c (constante)
- Seja n: y ← t op x
- Quando t será constante em n?
  - Neste caso, podemos reescrever n da forma
    - n: y ← c op x

Constant Folding

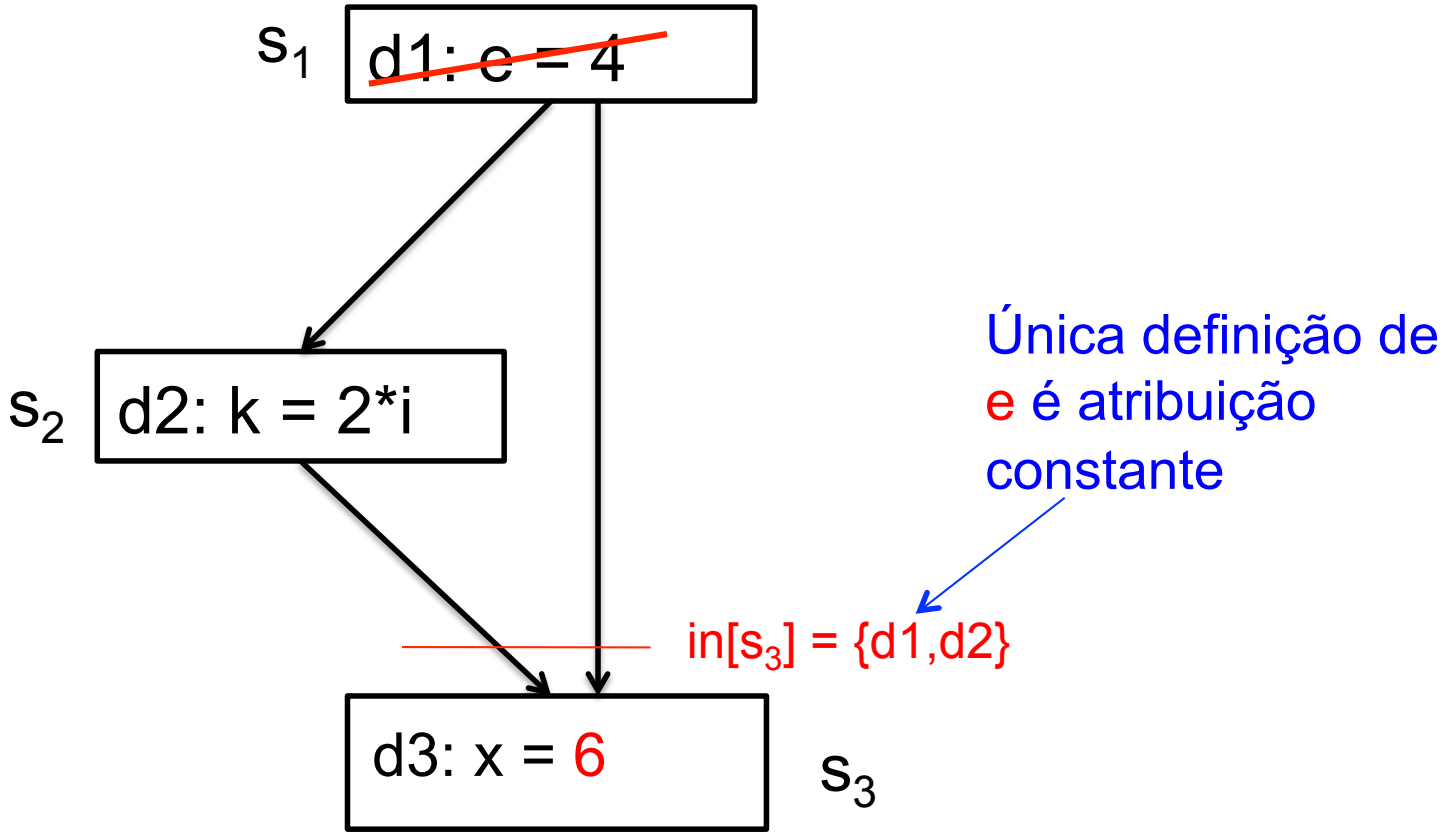


Que análise é necessária?

Reaching Definitions



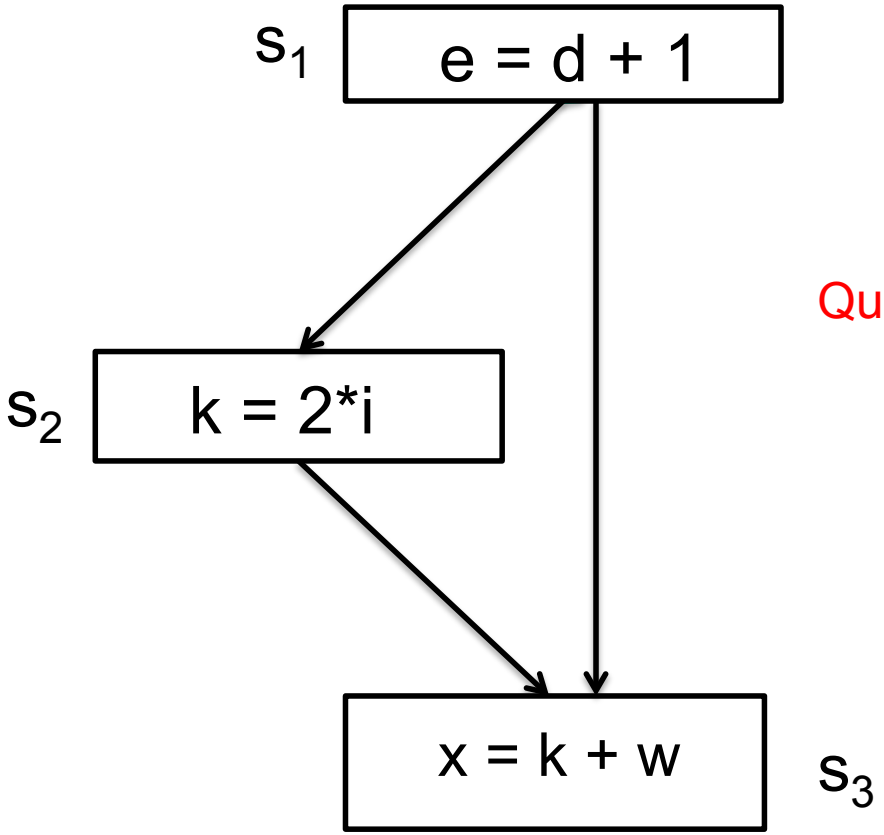
Constant Folding



Dead Code Elimination

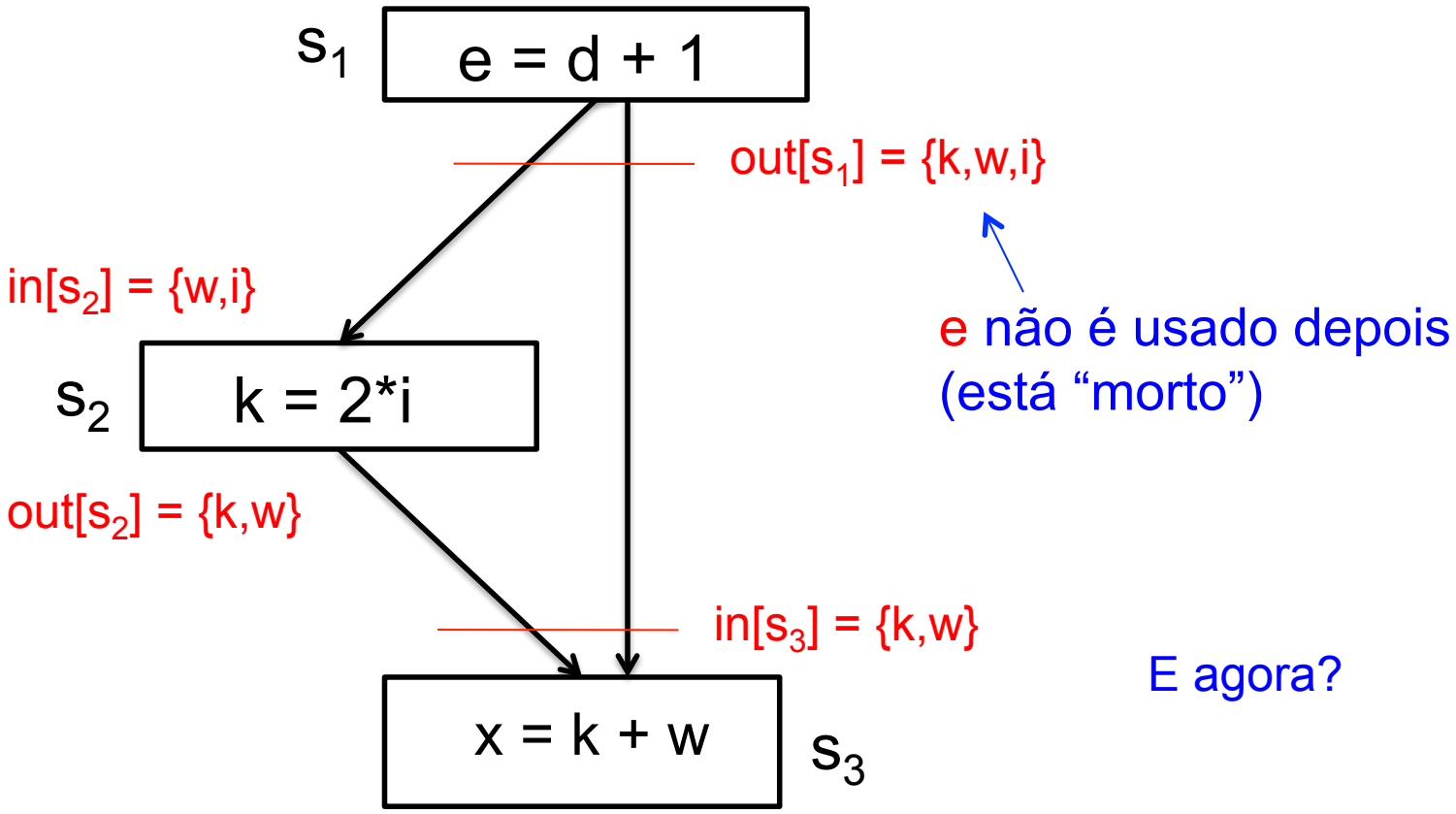
- Se a não está viva em out[s] em:
  - s: a ← t op x
  - s: a ← M[x]
- Podemos apagar s
- Qual análise é necessária?
- Tomar cuidado com efeitos colaterais

Dead Code Elimination

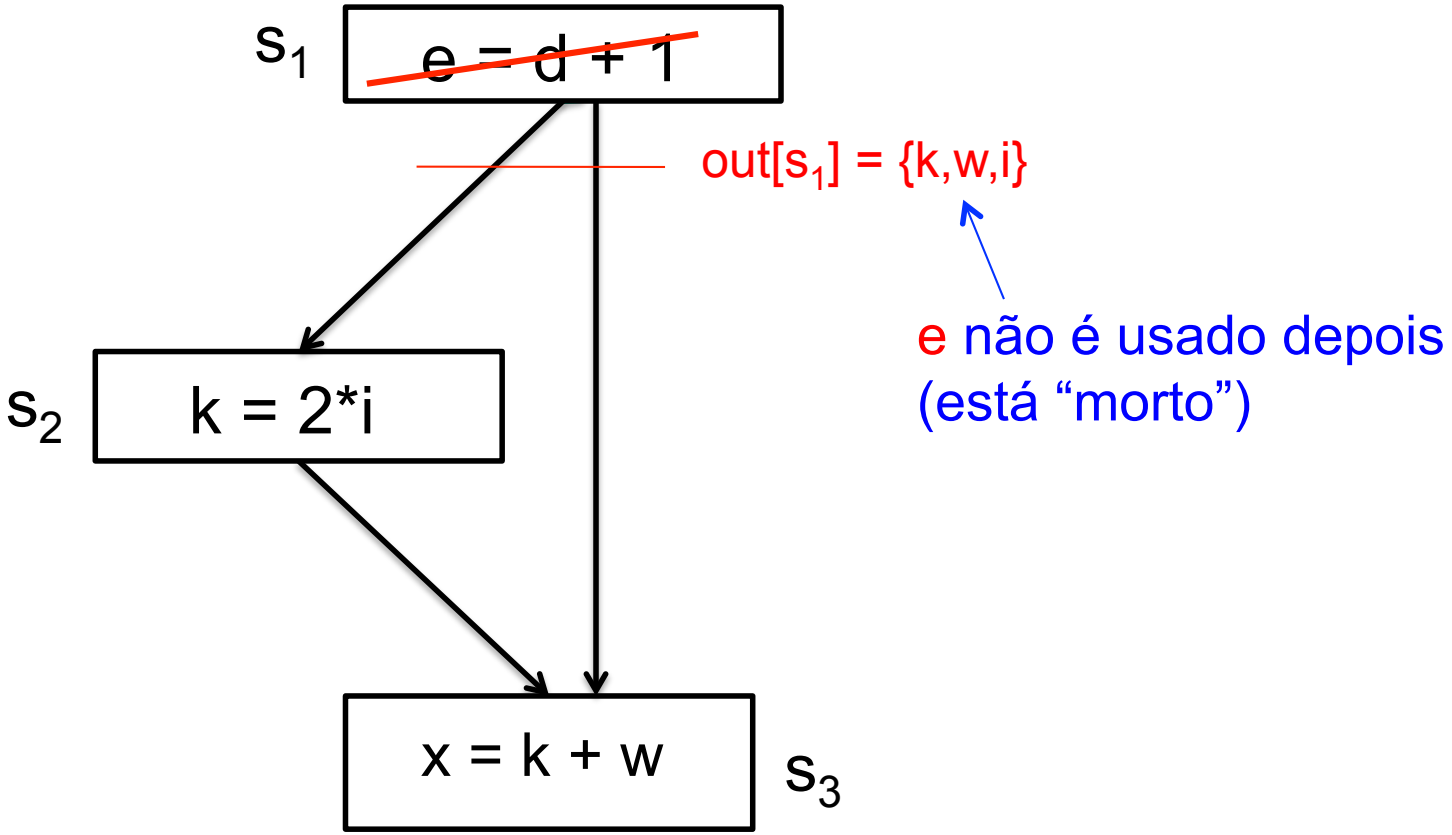


Que análise é necessária?

Liveness Analysis



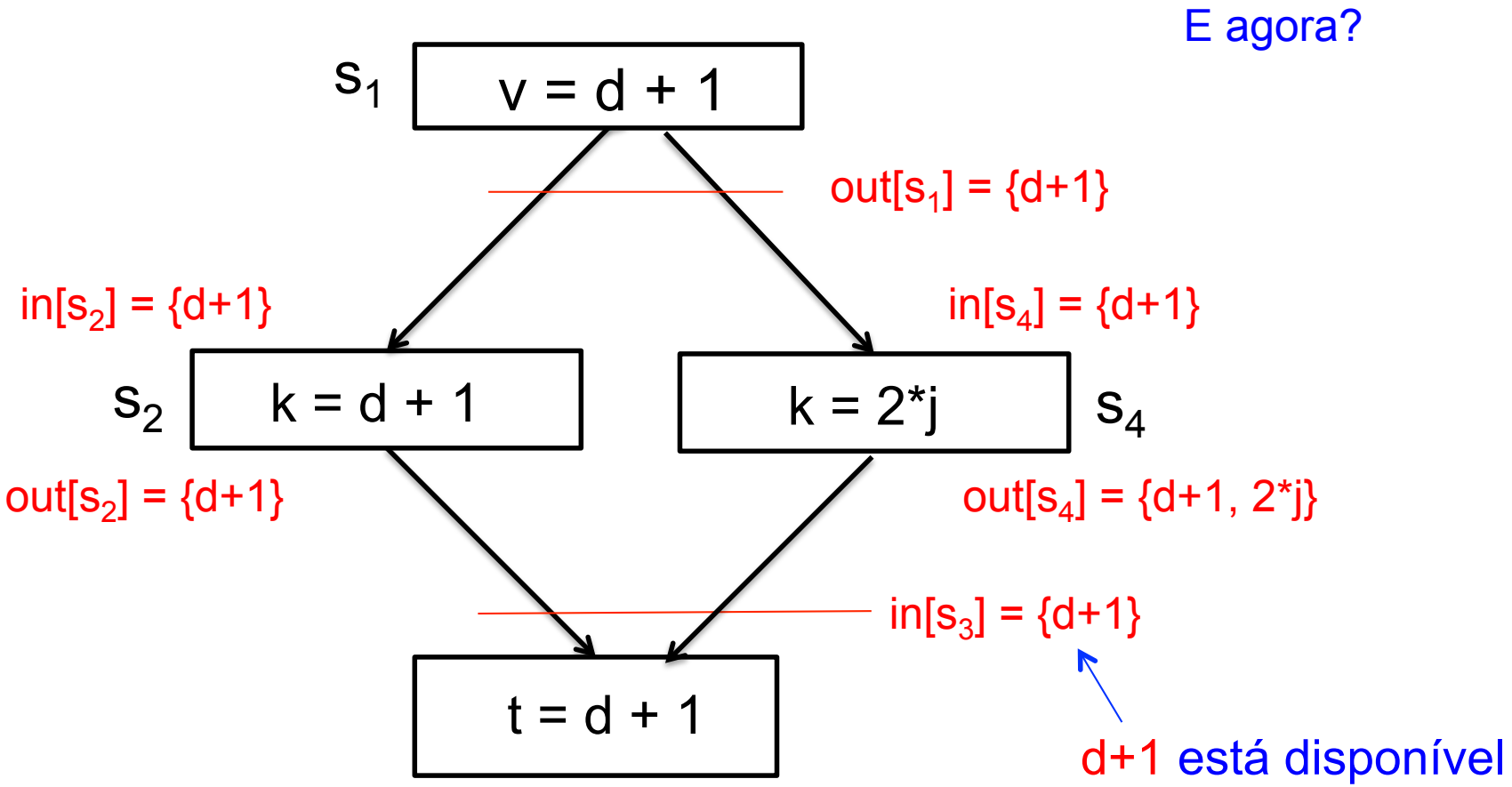
Deadcode Elimination



Common-subexpression Elimination

- Seja s: t ← x op y
- Se x op y está disponível em s
  - Elimine o cálculo de x op y de s
- Algoritmo
  - Usa informação das expressões disponíveis em s
  - Compute reaching expressions, encontrando expressões da forma n: v ← x op y que alcançam s
  - Crie um novo temporário w e reescreva n da forma
    - n: w ← x op y
    - n': v ← w
  - Modifique s para:
    - s: t ← w

Avaliable Expression



Common Sub-expression Elimination