Copy Propagation

- Dadas as instruções:

  i_1: **t** = z , onde z é variável

  i_2: y = **t** + x

- A variável t será "constante" em i_2 se:

  1. i_1 alcança i_2
  2. nenhuma outra definição de **t** alcança i_2
  3. não existe definição de **z** em qualquer caminho de i_1 a i_2, incluindo passagens sobre i_2 uma ou mais vezes

- Pode-se reescrever:

  i_2: y = **z** + x

Copy Propagation

- As Condições 1 e 2 podem ser checadas utilizando-se *ud-chains*

- Condição 3:
  - Nova *dataflow analysis*
    - c_in[B]: conjunto de cópias **s: x = y** tais que todo caminho do início até o nó B contém a sentença ***s***, e após a última ocorrência de ***s*** não há atribuições a ***y***
    - c_out[B]: idem para o final de B

Copy Propagation

- Condição 3
  - Nova *dataflow analysis*
    - c_gen[B]: ***s*** ocorre em B e não há atribuição a ***y*** após ***s***
    - c_kill[B]: ***s*** é morta em B se ***x*** ou ***y*** são atribuídos em B e ***s*** não está em B
      - Nota-se que diferentes atribuições ***x = y*** matam umas as outras
      - c_in[B] só pode conter uma sentença ***x = y*** com ***x*** à esquerda

Copy Propagation

- **Condição 3**
  - Equações: as mesmas de *available expressions*!
  - É chamada de Cópias Disponíveis

$$in[B] = \bigcap_{P \in Pred(B)} out[P]$$

$$in[B1] = \varnothing$$

$$out[B] = c\_gen[B] \cup (in[B] - c\_kill[B])$$

Copy Propagation

Algorithm 10.6. Copy propagation.

Input. A flow graph G, with ud-chains giving the definitions reaching block B, and with $c\_in[B]$ representing the solution to Equations 10.12, that is, the set of copies `x:=y` that reach block B along every path, with no assignment to

chains giving the uses of each definition.

Output. A revised flow graph.

Method. For each copy s: `x:=y` do the following.

1. Determine those uses of `x` that are reached by this definition of `x`, namely, s: `x:=y`.
2. Determine whether for every use of `x` found in (1), s is in $c\_in[B]$, where B is the block of this particular use, and moreover, no definitions of `x` or `y` occur prior to this use of `x` within B. Recall that if s is in $c\_in[B]$, then s is the only definition of `x` that reaches B.
3. If s meets the conditions of (2), then remove s and replace all uses of `x` found in (1) by `y`. □

Copy Propagation

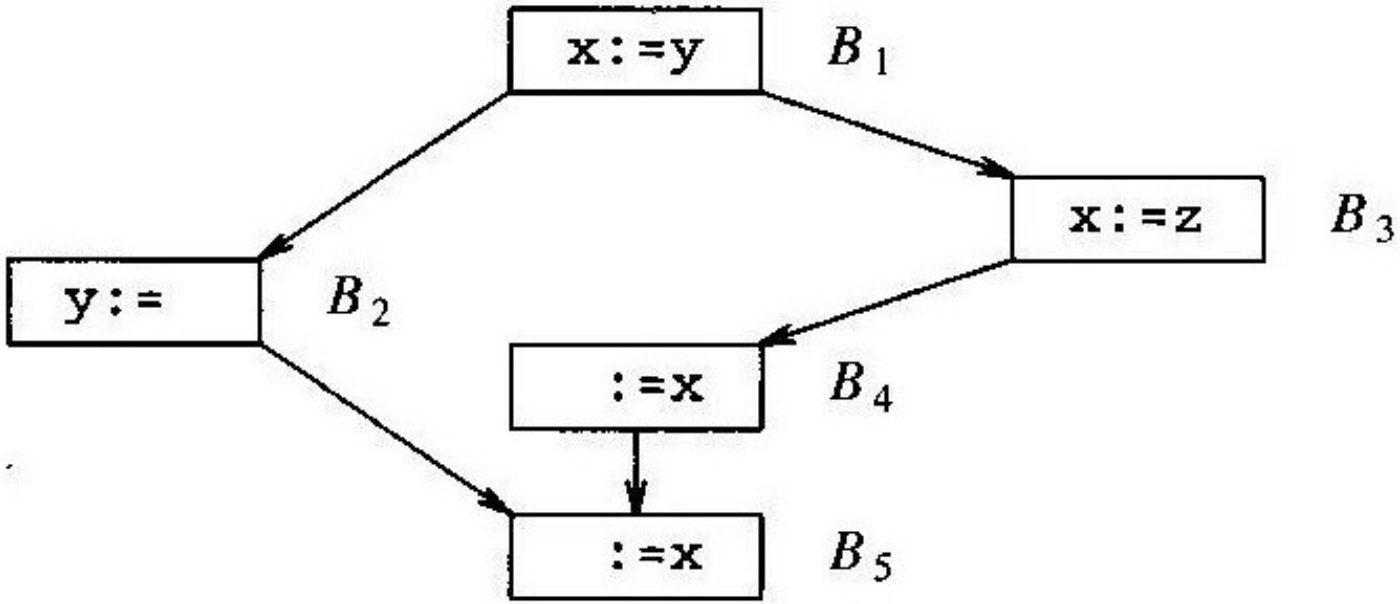


Fig. 10.35. Example flow graph.

Copy Propagation

c_gen/c_kill:

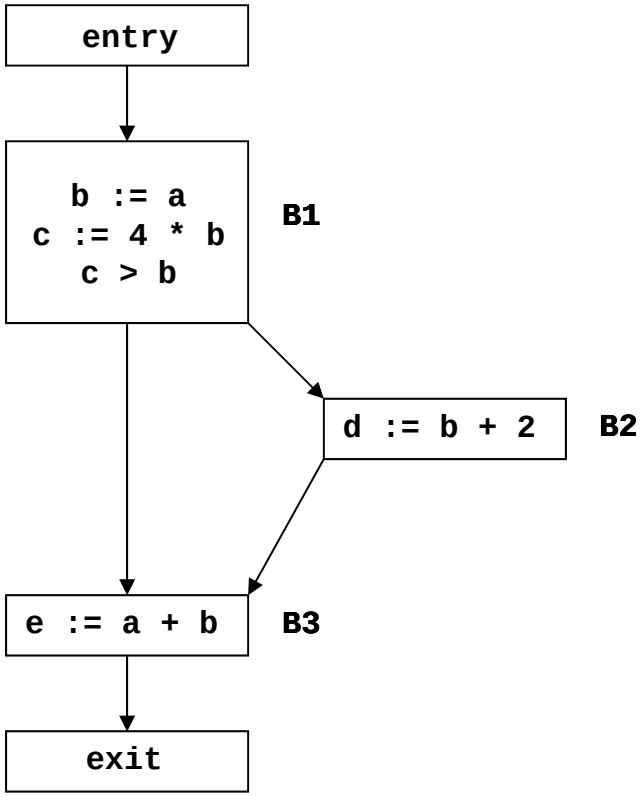
```
c_gen[B1] = {x=y}        c_kill[B1] = {x=z}
c_gen[B2] = {}           c_kill[B2] = {x=y}
c_gen[B3] = {x=z}        c_kill[B3] = {x=y}
```

- Os outros são vazios

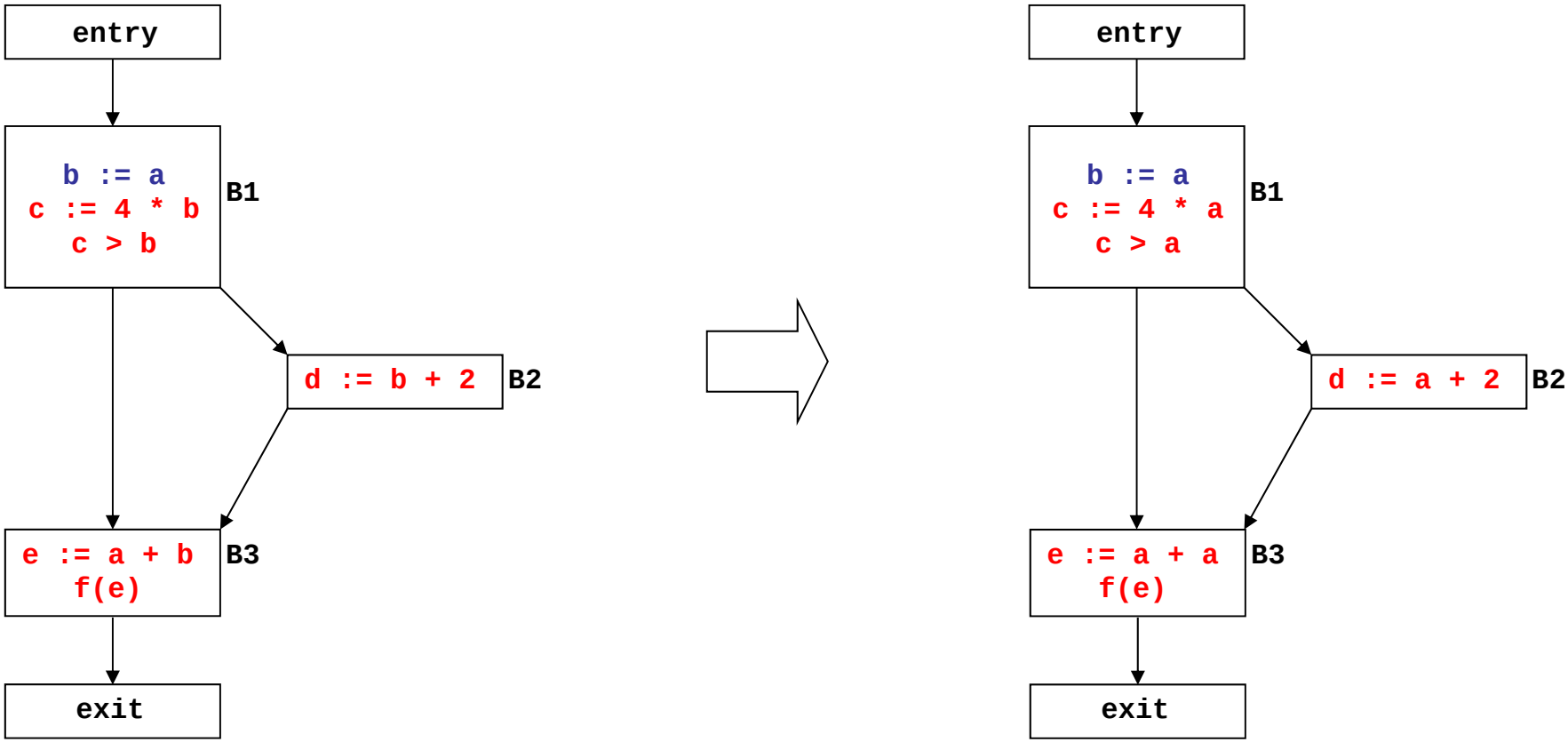
in/out:

```
in[B1] = {}              out[B1] = {x=y}
in[B2] = {x=y}           out[B2] = {}
in[B3] = {x=y}           out[B3] = {x=z}
in[B4] = {x=z}           out[B4] = {x=z}
in[B5] = {}              out[B5] = {}
```

Copy Propagation



Copy Propagation



Dead Code Elimination