1 Alcançabilidade de Definições

Gen ...

$Kill$ ...

IN ...

OUT ...

	Gen	Kill	IN	OUT
B_1	000	000	000	000
B_2	100	100	000	100
B_3	010	010	000	010
B_4	001	001	110	111
B_5	000	000	100	100
B_6	000	000	010	010

Table 1: Alcançabilidade de Definições — $(a := b + c, e := 7, d := b + c)$

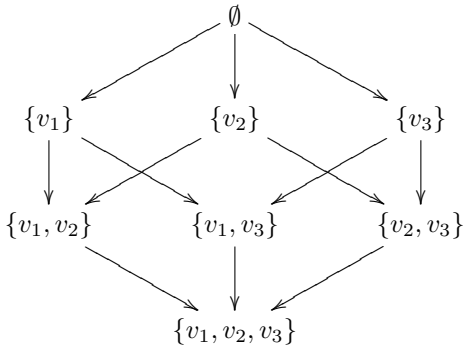
2 Vivacidade de Variáveis

Gen ...

$Kill$ ...

IN ...

OUT ...



	Gen	Kill	IN	OUT
B_1	00000	00000	01100	01100
B_2	01100	10000	01100	01100
B_3	00000	00001	01100	01100
B_4	01100	00010	01100	00000
B_5	00000	00000	01100	01100
B_6	00000	00000	01100	01100

Table 2: Vivacidade de Variáveis — (a, b, c, d, e)

3 Disponibilidade de Expressões

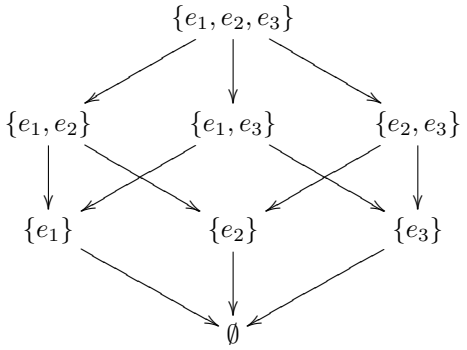
A análise é para frente (forward) e sua intenção é determinar em cada ponto do código, quais expressões estão disponíveis, isto é, foram seguramente executadas e, caso fossem executadas novamente (naquele ponto) produziriam o mesmo resultado.

Gen Indica quais expressões foram geradas dentro do bloco e que não foram "mortas" por redefinições de seus operandos dentro do mesmo bloco. É igual à entrada das expressões antecipáveis.

Kill Indica quais expressões (considerando o universo inteiro) foram mortas por redefinições (posteriores [1]) de seus operandos que ocorrem dentro do bloco.

IN Indica quais expressões estão disponíveis na entrada do bloco. É uma interseção das saídas dos blocos predecessores.

OUT Indica quais expressões estão disponíveis na saída do bloco. É igual ao último *Gen*.



	Gen	Kill	IN	OUT
B_1	0	0	0	0
B_2	1	0	0	1
B_3	0	0	0	0
B_4	1	0	0	1
B_5	0	0	1	1
B_6	0	0	0	0

Table 3: Disponibilidade de Expressões — ((+, b, c))

[1] Só faz sentido em análises internas ao bloco.

4 Disponibilidade de Expressões Anticipáveis

A análise é para trás (backward) e sua intenção é determinar em cada ponto do código, quais expressões podem ser movidas para o início do bloco (ou para blocos antecedentes).

Gen Indica quais expressões podem ser movidas para o início do bloco (ou para blocos antecedentes).

Kill Indica quais expressões (considerando o universo inteiro) foram mortas por redefinições (anteriores [2]) de seus operandos que ocorrem dentro do bloco.

IN Indica quais expressões podem ser movidas para blocos antecedentes.

OUT Indica quais expressões de blocos subsequêntes podem ser movidas para o final do bloco atual – estas expressões podem ou não serem antecipadas pelo bloco atual.

	Gen	Kill	IN	OUT
B_1	0	0	1	1
B_2	1	0	1	1
B_3	0	0	1	1
B_4	1	0	1	0
B_5	0	0	1	1
B_6	0	0	1	1

Table 4: Disponibilidade de Expressões Anticipáveis — ((+, b, c))

[2]Só faz sentido em análises internas ao bloco.

5 Disponibilidade Parcial de Expressões

Gen ...

$Kill$ ...

IN ...

OUT ...

	Gen	Kill	IN	OUT
B_1	0	0	0	0
B_2	1	0	0	1
B_3	0	0	0	0
B_4	1	0	1	1
B_5	0	0	1	1
B_6	0	0	0	0

Table 5: Disponibilidade Parcial de Expressões — ((+, b, c))

6 Mortalidade de Variáveis

Gen ...

$Kill$ ...

In ...

In ...

	Gen	Kill	IN	OUT
B_1	00000	00000	10011	10011
B_2	10000	01100	10011	10011
B_3	00001	00000	10011	10011
B_4	00010	01100	10011	11111
B_5	00000	00000	10011	10011
B_6	00000	00000	10011	10011

Table 6: Mortalidade de Variáveis — (a, b, c, d, e)

7 Alcançabilidade de Definições para Propagação de Cópias

Gen ...

$Kill$ ...

In ...

In ...

	Gen	Kill	IN	OUT
B_1	000	000	000	000
B_2	000	000	000	000
B_3	000	000	000	000
B_4	000	000	000	000
B_5	000	000	000	000
B_6	000	000	000	000

Table 7: Alcançabilidade de Definições para Propagação de Cópias — $(a := b + c, e := 7, d := b + c)$

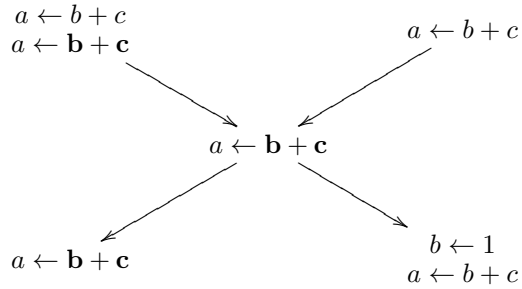
8 Eliminação de Redundâncias Parciais

8.1 Expressão Redundante

Uma expressão é *redundante* no ponto p se em cada caminho até p:

1. Ela é avaliada antes de alcançar p, e
2. Nenhum de seus operandos constituintes é redefinido antes de p.

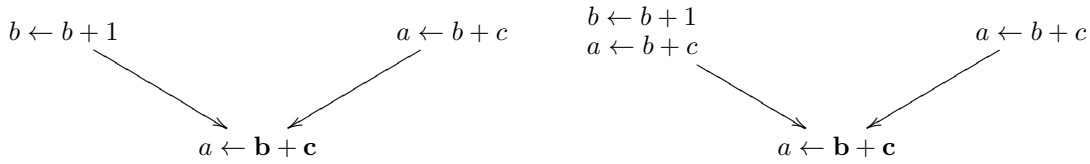
Por exemplo, na Equação 1, as ocorrências de expressões em negrito são redundantes.



(1)

Uma expressão é *parcialmente redundante* no ponto p se ela é redundante ao longo de alguns caminhos, mas não todos, até p.

Por exemplo, na Equação 2, a expressão $b + c$ em negrido no diagrama da esquerda é parcialmente redundante. A inserção de uma cópia de $b + c$ depois da definição de b pode tornar uma expressção parcialmente redundante em uma totalmente redundante como mostra o diagrama da direita.



(2)

	ENTRY	B_1	B_2	B_3	B_4	B_5	B_6	EXIT
e_gen	{0}	{0}	{1}	{0}	{1}	{0}	{0}	{0}
e_kill	{0}	{0}	{0}	{0}	{0}	{0}	{0}	{0}
anticipated_out	{1}	{1}	{1}	{1}	{0}	{1}	{1}	{0}
anticipated_in	{1}	{1}	{1}	{1}	{1}	{1}	{1}	{0}
available_in	{0}	{0}	{1}	{1}	{1}	{1}	{1}	{1}
available_out	{0}	{1}	{1}	{1}	{1}	{1}	{1}	{1}
earliest	{1}	{1}	{0}	{0}	{0}	{0}	{0}	{0}
postponable_in	{0}	{0}	{1}	{1}	{0}	{0}	{1}	{0}
postponable_out	{0}	{1}	{0}	{1}	{0}	{0}	{1}	{0}
latest	{0}	{0}	{1}	{0}	{0}	{0}	{1}	{0}
used_out	{0}	{0}	{1}	{0}	{0}	{1}	{1}	{0}
used_in	{0}	{0}	{0}	{0}	{1}	{1}	{0}	{0}
*cond*_1	{0}	{0}	{1}	{0}	{0}	{0}	{1}	{0}
*cond*_2	{0}	{0}	{1}	{0}	{1}	{0}	{0}	{0}

Table 8: Eliminação de Redundâncias Parciais — ((+, b, c))

9 Propagação de Constantes

Gen ...

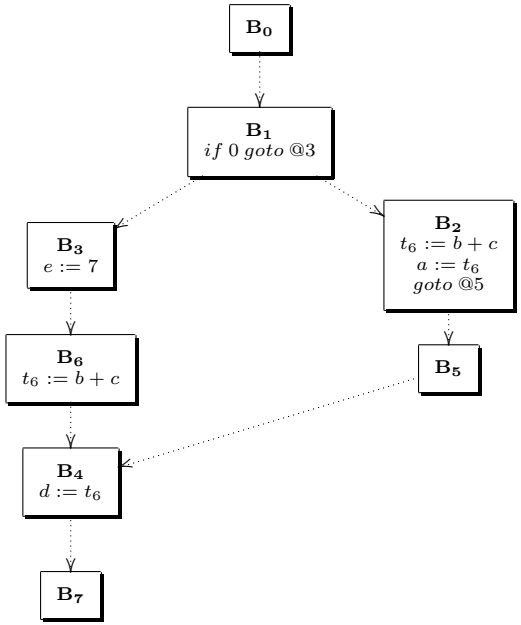
$Kill$ ...

In ...

In ...

	IN	OUT
B_1	$(\top, \top, \top, \top, \top, \top)$	$(\top, \top, \top, \top, \top, \top)$
B_2	$(\top, \top, \top, \top, \top, \top)$	$(\top, \top, \top, \top, \top, \top)$
B_3	$(\top, \top, \top, \top, \top, \top)$	$(\top, \top, \top, \top, 7, \top)$
B_4	$(\top, \top, \top, \top, 7, \top)$	$(\top, \top, \top, \top, 7, \top)$
B_5	$(\top, \top, \top, \top, \top, \top)$	$(\top, \top, \top, \top, \top, \top)$
B_6	$(\top, \top, \top, \top, 7, \top)$	$(\top, \top, \top, \top, 7, \top)$

Table 9: Propagação de Constantes — (a, b, c, d, e, t_6)


B0
B1
if 0 goto @3
B3
e := 7
B2
t6 := b + c
a := t6
goto @5
B6
t6 := b + c
B5
B4
d := t6
B7