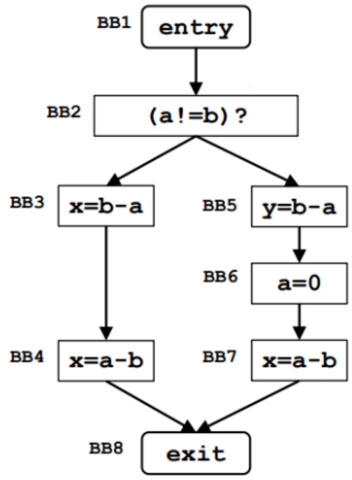
Very Busy Expressions

- Un'espressione è ***very busy*** in un punto *p* se, indipendentemente dal percorso preso da *p*, l'espressione viene usata prima che uno dei suoi operandi venga definito.
- Un'espressione *a+b* è **very busy** in un punto *p* se *a+b* è valutata in tutti i percorsi da *p* a EXIT e non c'è una definizione di a o b lungo tali percorsi
  - Ci interessa l'insieme di espressioni disponibili (available) all'inizio del blocco B
  - L'insieme dipende dai percorsi che cominciano al punto *p* prima di B

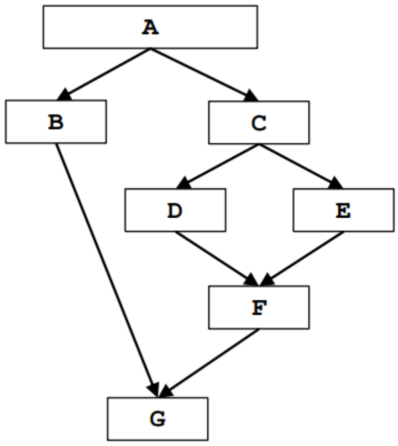


Framework	
Domain	Insieme delle espressioni: {b-a,a-b}
Direction	Backward: $In[b] = f_b [Out[b]]$ $Out[b] = \cap In [Suc[b]]$
Transfer function	$f_b(x) = Gen_b \cup (Out_b - Kill_b)$
Meet Operator	$\cap$
Boundary Condition	$In[exit] = \varnothing$
Initial interior points	$In[B] = u$

BB	**ITERAZIONE 1**	
	IN [B]	**OUT [B]**
BB1	b-a	b-a
BB2	b-a	(b-a,a-b) ∩ b-a = b-a
BB3	b-a ∪ (a-b \ ∅)= (b-a,a-b)	a-b
BB4	a-b ∪ (∅ \ ∅) = a-b	∅
BB5	b-a ∪ (∅ \ ∅) = b-a	∅
BB6	∅ ∪ ((a-b) \ (a-b,b-a)) = ∅	a-b
BB7	a-b	∅
BB8	∅	∅

Dominator Analysis

- In un CFG diciamo che un nodo *X* **domina** un altro nodo *Y* se il nodo *X* appare in ogni percorso del grafo che porta dal blocco ENTRY al blocco *Y*
- Annotiamo ogni *basic block* Bi con un insieme DOM[Bi]
  - Bi ∈ DOM[Bj] se e solo se Bi domina Bj
- Per definizione un nodo domina sé stesso
  - Bi ∈ DOM[Bi]



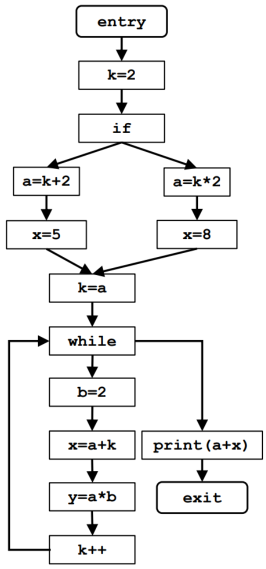
DOM[F] = {A,C,F}

Framework	
Domain	Insieme dei Basic Blocs: {A,B,C,D,E,F,G}
Direction	Forward: In[b] = ∩ Out [Pred[b]] Out[b] = f_b [In[b]]
Transfer function	f_b= Gen_b ∪ In_b
Meet Operator	∩
Boundary Condition	Out[A] = A
Initial interior points	Out[b] = *u*

BB	**ITERAZIONE 1**	
	IN [B]	**OUT [B]**
A	∅	A
B	A	B ∪ In[B] = B,A
C	A	C ∪ In[C] = C,A
D	C,A	D ∪ In[D] = D,C,A
E	C,A	E ∪ In[E] = E,C,A
F	Out[D] ∩ Out[E] = C,A	F ∪ In[F] = F,C,A
G	Out[B] ∩ Out[F] = A	G ∪ In[G] = G,A

Constant Propagation

- L’obiettivo della *constant propagation* è quello di determinare in quali punti del programma le variabili hanno un valore costante.
- L’informazione da calcolare per ogni nodo *n* del CFG è un insieme di ***coppie*** del tipo *<variabile, valore costante>*.
- Se abbiamo la coppia *<x, c>* al nodo *n*, significa che *x* è garantito avere il valore *c* ogni volta che *n* viene raggiunto durante l’esecuzione del programma.



Framework	
Domain	coppie del tipo <variabile, valore costante>
Direction	Forward: $In[b] = \cap\ Out\ [Pred[b]]$ $Out[b] = f_b\ [In[b]]$
Transfer function	$f_b = Gen_b \cup (In_b - Kill_b)$
Meet Operator	$\cap$
Boundary Condition	Out[entry] = ∅
Initial interior points	Out[b] = *u*

BB	**ITERAZIONE 1**		**ITERAZIONE 2**	
	IN [B]	**OUT [B]**	**IN [B]**	**OUT [B]**
entry	∅	∅	∅	∅
k=2	∅	<k,2>	∅	<k,2>
if	<k,2>	<k,2>	<k,2>	<k,2>
a=k+2	<k,2>	<k,2>,<a,4>	<k,2>	<k,2>,<a,4>
x=5	<k,2>,<a,4>	<k,2>,<a,4>,<x, 5>	<k,2>,<a,4>	<k,2>,<a,4>,<x, 5>
a=k*2	<k,2>	<k,2>,<a,4>	<k,2>	<k,2>,<a,4>

x=8	<k,2>,<a,4>	<k,2>,<a,4>,<x, 8>	<k,2>,<a,4>	<k,2>,<a,4>,<x, 8>
k=a	<k,2>,<a,4>,<x, 8> ∩ <k,2>,<a,4>,<x, 5> = <k,2>,<a,4>	<a,4>	<k,2>,<a,4>,<x, 8> ∩ <k,2>,<a,4>,<x, 5> = <k,2>,<a,4>	<a,4>
while	<a,4>	<a,4>	<a,4>,<b,2>,<y, 8> ∩ <a,4> = <a,4>	<a,4>
b=2	<a,4>	<a,4>,<b,2>	<a,4>	<a,4>,<b,2>
x=a+k	<a,4>,<b,2>	<a,4>,<b,2>	<a,4>,<b,2>	<a,4>,<b,2>
y=a*b	<a,4>,<b,2>	<a,4>,<b,2>,<y, 8>	<a,4>,<b,2>	<a,4>,<b,2>,<y, 8>
k++	<a,4>,<b,2>,<y, 8>	<a,4>,<b,2>,<y, 8>	<a,4>,<b,2>,<y, 8>	<a,4>,<b,2>,<y, 8>
print(a+x)	<a,4>	<a,4>	<a,4>	<a,4>
exit	<a,4>	<a,4>	<a,4>	<a,4>