

Data Flow Analysis

Assignment 2

Iacopo Ruzzier, Daniele Fassetta, Anna Semeraro

13 aprile 2025

Indice

1	Very Busy Expressions	2
1.1	Definizione del problema	2
1.2	Esempio	2
2	Dominator Analysis	3
2.1	Definizione del problema	3
2.2	Esempio	3
3	Constant propagation	4
3.1	Definizione del problema	4
3.2	Esempio	4

1 Very Busy Expressions

1.1 Definizione del problema

- 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

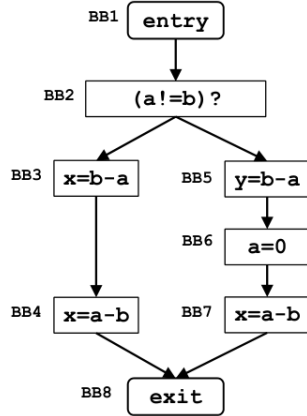
	Very Busy Expressions
Domain	Expressions
Direction	Backward $in[b] = f_b(out[b])$ $out[b] = \wedge in(succ[b])$
Transfer function	$f_b = Gen_b \cup (x - Kill_b)$
Meet Operation (\wedge)	\cap
Boundary Condition	$in[exit] = \emptyset$
Initial interior points	$in[b] = \mathcal{U}$

Tabella 1: Very busy expressions

Dove

- $Gen[b]$: espressioni valutate all'interno del Basic Block
- $Kill[b]$: un'espressione viene uccisa quando un suo operando viene ridefinito all'interno di b

1.2 Esempio



BB	Iterazione 1	
	$OUT[B] = \wedge IN(succ[b])$	$IN[B] = Gen[B] \cup (OUT[B] - Kill[B])$
BB7	\emptyset (boundary condition)	$a - b \cup (\emptyset - \emptyset) = a - b$
BB6	$a - b$	$\emptyset \cup (\{a - b\} - \{a - b, b - a\}) = \emptyset$
BB5	\emptyset	$b - a \cup (\emptyset - \emptyset) = b - a$
BB4	\emptyset	$a - b \cup (\emptyset - \emptyset) = a - b$
BB3	$a - b$	$b - a \cup (\{a - b\} - \emptyset) = \{b - a, a - b\}$
BB2	$\{b - a\} \cap \{a - a, a - b\} = b - a$	$\emptyset \cup (\{b - a\} - \emptyset) = b - a$
BB1	$b - a$	/

2 Dominator Analysis

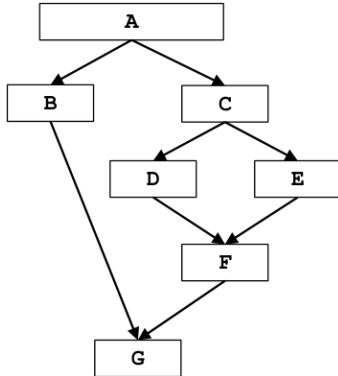
2.1 Definizione del problema

- 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 B_i con un insieme $Dom[B_i]$:
 - $B_i \in Dom[B_j] \iff B_i$ domina B_j
 - $B_i \in Dom[B_i]$: per definizione un nodo domina se stesso

	Dominator Analysis
Domain	Basic Blocks
Direction	Forward $out[b] = f_b(in[b])$ $in[b] = \wedge out(pred[b])$
Transfer function	$f_b = Dom[b] = b \cup x$
Meet Operation (\wedge)	\cap
Boundary Condition	$in[entry] = \emptyset$
Initial interior points	$in[b] = \mathcal{U}$

Tabella 2: Dominator analysis

2.2 Esempio



BB	Iterazione 1	
	$IN[B] = \wedge OUT(pred[b])$	$DOM[B] = B \cup IN[B]$
A	\emptyset (boundary condition)	$A \cup \emptyset = A$
B	A	$B \cup A = \{A, B\}$
C	A	$C \cup A = \{A, C\}$
D	$\{A, C\}$	$D \cup \{A, C\} = \{A, C, D\}$
E	$\{A, C\}$	$E \cup \{A, C\} = \{A, C, E\}$
F	$\{A, C, D\} \cap \{A, C, E\} = \{A, C\}$	$F \cup \{A, C\} = \{A, C, F\}$
G	$\{A, B\} \cap \{A, C, F\} = A$	$G \cup A = \{A, G\}$

3 Constant propagation

3.1 Definizione del problema

- 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 del CFG è un insieme di coppie del tipo $\langle \text{variabile}, \text{valore costante} \rangle$.
- Se abbiamo la coppia $\langle x, c \rangle$ al nodo n , significa che x è garantito avere il valore c ogni volta che n viene raggiunto durante l'esecuzione del programma.

	Constant Propagation
Domain	pairs (v, c)
Direction	Forward $out[b] = f_b(in[b])$ $in[b] = \bigwedge out(pred[b])$
Transfer function	$f_b = Gen[b] \cup (x - Kill[b])$
Meet Operation (\wedge)	\cap
Boundary Condition	$in[entry] = \emptyset$
Initial interior points	$in[b] = \mathcal{U}$

Tabella 3: Constant propagation

- $Gen[b]$ rappresenta l'insieme di nuovi assegnamenti (all'interno di b) con valore costante. I casi di interesse sono:
 - $x = c$ con c costante: $Gen[b] = Gen[b] \cup (x, c)$
 - $x = c1 \oplus c2$ con $c1, c2$ costanti: $Gen[b] = Gen[b] \cup (x, c1 \oplus c2)$ (calcolando il valore dell'espressione)
 - $x = c \oplus y$ o $x = y \oplus c$ con c costante:
 1. controlliamo che y sia presente nell'insieme in input: $\exists (y, v_y) \in In[b]$
 2. se è vero, $Gen[b] = Gen[b] \cup (x, e)$ con $e = c \oplus v_y$ (o $v_y \oplus c$)
 - $x = y \oplus z$, valutando y e z allo stesso modo del caso precedente:

$$\exists (y, v_y), (z, v_z) \in In[b] \implies Gen[b] = Gen[b] \cup (x, e) \text{ con } e = v_y \oplus v_z$$

Nel caso di definizioni multiple di x , riteniamo valida solamente l'ultima all'interno di b

- $Kill[b]$: ogni definizione $x = \text{expr}$ uccide le coppie $(x, c) \in \mathcal{D}$ (dominio)

3.2 Esempio

Data la convergenza, si ferma l'esecuzione dell'algoritmo all'iterazione 3.

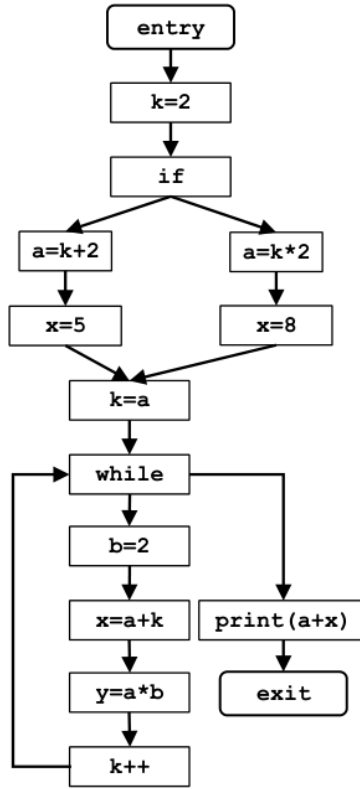


Figura 1: Esempio 3

	Iterazione 1	
	IN[B]	OUT[B]
BB1	\emptyset	\emptyset
BB2	\emptyset	$\{(k, 2)\}$
BB3	$\{(k, 2)\}$	$\{(k, 2)\}$
BB4	$\{(k, 2)\}$	$\{(k, 2), (a, 4)\}$
BB5	$\{(k, 2), (a, 4)\}$	$\{(k, 2), (a, 4), (x, 5)\}$
BB6	$\{(k, 2)\}$	$\{(k, 2), (a, 4)\}$
BB7	$\{(k, 2), (a, 4)\}$	$\{(k, 2), (a, 4), (x, 5)\}$
BB8	$\{(k, 2), (a, 4)\}$	$\{(k, 4), (a, 4)\}$
BB9	$\{(k, 4), (a, 4)\} \cap \mathcal{U} = \{(k, 4), (a, 4)\}$	$\{(k, 4), (a, 4)\}$
BB10	$\{(k, 4), (a, 4)\}$	$\{(k, 4), (a, 4), (b, 2)\}$
BB11	$\{(k, 4), (a, 4), (b, 2)\}$	$\{(k, 4), (a, 4), (b, 2), (x, 8)\}$
BB12	$\{(k, 4), (a, 4), (b, 2), (x, 8)\}$	$\{(k, 4), (a, 4), (b, 2), (x, 8), (y, 8)\}$
BB13	$\{(k, 4), (a, 4), (b, 2), (x, 8), (y, 8)\}$	$\{(k, 5), (a, 4), (b, 2), (x, 8), (y, 8)\}$
BB14	$\{(k, 4), (a, 4)\}$	$\{(k, 4), (a, 4)\}$
BB15	$\{(k, 4), (a, 4)\}$	$\{(k, 4), (a, 4)\}$

Tabella 4: Iterazione 1

	Iterazione 2	
	IN[B]	OUT[B]
BB1	\emptyset	\emptyset
BB2	\emptyset	$\{(k, 2)\}$
BB3	$\{(k, 2)\}$	$\{(k, 2)\}$
BB4	$\{(k, 2)\}$	$\{(k, 2), (a, 4)\}$
BB5	$\{(k, 2), (a, 4)\}$	$\{(k, 2), (a, 4), (x, 5)\}$
BB6	$\{(k, 2)\}$	$\{(k, 2), (a, 4)\}$
BB7	$\{(k, 2), (a, 4)\}$	$\{(k, 2), (a, 4), (x, 5)\}$
BB8	$\{(k, 2), (a, 4)\}$	$\{(k, 4), (a, 4)\}$
BB9	$\{(k, 4), (a, 4)\} \cap \{(y, 8), (k, 5), (a, 4), (b, 2)(x, 8)\} = \{(a, 4)\}$	$\{(a, 4)\}$
BB10	$\{(a, 4)\}$	$\{(a, 4), (b, 2)\}$
BB11	$\{(a, 4), (b, 2)\}$	$\{(a, 4), (b, 2)\}$
BB12	$\{(a, 4), (b, 2)\}$	$\{(a, 4), (b, 2), (y, 8)\}$
BB13	$\{(a, 4), (b, 2), (y, 8)\}$	$\{(a, 4), (b, 2), (y, 8)\}$
BB14	$\{(a, 4), (b, 2), (y, 8)\}$	$\{(a, 4), (b, 2), (y, 8)\}$
BB15	$\{(a, 4), (b, 2), (y, 8)\}$	$\{(a, 4), (b, 2), (y, 8)\}$

Tabella 5: Iterazione 2

	Iterazione 3	
	IN[B]	OUT[B]
BB1	\emptyset	\emptyset
BB2	\emptyset	$\{(k, 2)\}$
BB3	$\{(k, 2)\}$	$\{(k, 2)\}$
BB4	$\{(k, 2)\}$	$\{(k, 2), (a, 4)\}$
BB5	$\{(k, 2), (a, 4)\}$	$\{(k, 2), (a, 4), (x, 5)\}$
BB6	$\{(k, 2)\}$	$\{(k, 2), (a, 4)\}$
BB7	$\{(k, 2), (a, 4)\}$	$\{(k, 2), (a, 4), (x, 5)\}$
BB8	$\{(k, 2), (a, 4)\}$	$\{(k, 4), (a, 4)\}$
BB9	$\{(k, 4), (a, 4)\} \cap \{(y, 8), (a, 4), (b, 2)\} = \{(a, 4)\}$	$\{(a, 4)\}$
BB10	$\{(a, 4)\}$	$\{(a, 4), (b, 2)\}$
BB11	$\{(a, 4), (b, 2)\}$	$\{(a, 4), (b, 2)\}$
BB12	$\{(a, 4), (b, 2)\}$	$\{(a, 4), (b, 2), (y, 8)\}$
BB13	$\{(a, 4), (b, 2), (y, 8)\}$	$\{(a, 4), (b, 2), (y, 8)\}$
BB14	$\{(a, 4), (b, 2), (y, 8)\}$	$\{(a, 4), (b, 2), (y, 8)\}$
BB15	$\{(a, 4), (b, 2), (y, 8)\}$	$\{(a, 4), (b, 2), (y, 8)\}$

Tabella 6: Iterazione 3