**ESERCIZIO 1 – VERY BUSY EXPRESSION**

Seguendo la definizione di ‘very busy expression’, cioè:

* Un’espressione viene definita ‘very busy’ in un punto P se, in ogni percorso da P ad EXIT, l’espressione viene utilizzata prima di una ridefinizione.

Formalizziamo il primo problema usando la DataFlow Analysis tramite il seguente schema:

|  |  |
| --- | --- |
|  | VERY BUSY EXPRESSION |
| DOMAIN | {(a-b), (b-a)} |
| DIRECTION | Backward  in[BB] = f(out(BB))  out[BB] = ∧ in(succ[BB]) |
| TRANSFER FUNCTION | f(x) = Use[B] ∪ (x - Def[B]) |
| MEET OPERAND (∧) | Intersezione ∩ |
| BOUNDARY CONDITION | Out[Exit] = ∅ |
| INITIAL INTERIOR POINTS | Out[BB] = universal set |

Per quanto riguarda le iterazioni, si propone la seguente tabella, con una sola iterazione causa la mancanza di cicli nel CFG proposto dall’esercizio:

|  |  |  |
| --- | --- | --- |
|  | **ITERAZIONE 1** | |
|  | **IN** | **OUT** |
| **BB8 (exit)** | ∅ | ∅ |
| **BB7** | {a - b} ∪ (∅ - {x}) = {a - b} | ∅ |
| **BB6** | ∅ ∪ ({a - b} - {a}) = ∅ | = in[BB7] = {a - b} |
| **BB5** | {b - a} ∪ (∅ - {y}) = {b - a} | = in[BB6] = ∅ |
| **BB4** | {a - b} ∪ ({a - b} - {x}) = {a - b} | = in[BB7] = {a - b} |
| **BB3** | {b - a} ∪ ({a - b} - {x}) = {a - b, b - a} | = in[BB4] = {a - b} |
| **BB2** | ∅ ∪ ({b - a} - ∅) = {b - a} | in[BB3] ∩ in[BB5] = {a - b, b - a} ∩ {b - a} = {b – a} |
| **BB1 (entry)** | ∅ ∪ ({b - a} - ∅) = {b - a} | = in[BB2] = {b - a} |

**ESERCIZIO 2 – Dominators Analysis**

Def di “nodo d domina x”:

Il nodo “d” domina il nodo “x” se “d” appare in OGNI percorso del grafo che porta a “x”, ovvero non è possibile arrivare dall’entry a “x” senza essere passati per “d”.

Formalizziamo il secondo problema usando la DataFlow Analysis tramite il seguente schema:

|  |  |
| --- | --- |
|  | Dominators Analysis |
| DOMAIN | Sets of nodes (BB) |
| DIRECTION | Forward  Out[b] = fb(in[b])  in[BB] = ∧ out(pred[B]) |
| TRANSFER FUNCTION | f(x) = gen[B] ∪ (x - kill[B]) |
| MEET OPERAND (∧) | Intersezione ∩ |
| BOUNDARY CONDITION | Out[entry] = {entry} |
| INITIAL INTERIOR POINTS | Out[BB] = U (universal set) |

Specifichiamo la funzione di trasferimento nel seguente modo:

* Gen[B] : genera se stesso (per definizione il nodo si auto domina)
* Kill[B] : sempre vuoto, in quanto non esiste modo di “killare” un dominatore

Pertanto la funzione è semplificabile come:

* f(x) = gen[B] ∪ x

Anche in questa caso essendo il grafo aciclico sarà presente una sola iterazione, visto l’assenza di una retroazione.

|  |  |  |
| --- | --- | --- |
|  | **ITERAZIONE 1** | |
|  | **IN** | **OUT** |
| **BB1 (A)** | ∅ | {A} |
| **BB2 (B)** | Out[A] = {A} | B U A = {A,B} |
| **BB3 (C)** | Out[A] = {A} | C U A = {A,C} |
| **BB4 (D)** | Out[C] = {A,C} | D U {A,C} = {A,C,D} |
| **BB5 (E)** | Out[C] = {A,C} | E U {A,C} = {A,C,E} |
| **BB6 (F)** | Out[D] ∩ Out[E] = {A,C} | F U {A,C} = {A,C,F} |
| **BB7 (G)** | Out[B] ∩ Out[F] = {A} | G U A = {G,A} |

**ESERCIZIO 2 – Constant Propagation**

Formalizziamo il terzo problema tramite il framework di DataFlow Analysis con il seguente schema:

|  |  |
| --- | --- |
|  | Constant Propagation |
| DOMAIN | Coppie <x,c> (x variabile, c costante) |
| DIRECTION | Forward  Out[b] = fb(in[b])  in[BB] = ∧ out(pred[B]) |
| TRANSFER FUNCTION | f(x) = gen[B] ∪ (x - kill[B]) |
| MEET OPERAND (∧) | Intersezione ∩ |
| BOUNDARY CONDITION | Out[entry] = ∅ |
| INITIAL INTERIOR POINTS | Out[BB] = U (universal set) |

Descriviamo funzione di trasferimento, significato dell’intersezione e Universal set sul dominio definito: