**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 | Set of expression |
| DIRECTION | Backward  in[BB] = f(out(BB))  out[BB] = Ù in(succ[BB]) |
| TRANSFER FUNCTION | f(x) = Use[B] È (x - Def[B]) |
| MEET OPERAND (Ù) | Intersezione Ç |
| BOUNDARY CONDITION | in[Exit] = Æ |
| INITIAL INTERIOR POINTS | in[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)** | Æ | Æ |
| **BB7** | {a - b} ∪ (∅ - ∅) = {a - b} | Æ |
| **BB6** | ∅ ∪ {a - b} - {(a-b), (b-a)} = ∅ | = in[BB7] = {a - b} |
| **BB5** | {b - a} ∪ (∅ - ∅) = {b - a} | = in[BB6] = ∅ |
| **BB4** | {a - b} ∪ (∅ - ∅) = {a - b} | = in[BB8] = ∅ |
| **BB3** | {b - a} ∪ ({a - b} - ∅) = {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} |

Descriviamo la funzione di trasferimento:

* Use: l’espressione RHS
* Def: es.: ‘x = y op. Z’, vengono killate le espressioni che hanno come operando ‘x’

**ESERCIZIO 2 – Dominators Analysis**

Definizione di “nodo dominante un altro nodo”:

Il nodo “d” domina il nodo “x” se e solo 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:

|  |  |
| --- | --- |
|  | DOMINATOR ANALYSIS |
| DOMAIN | Sets of nodes (BB) |
| DIRECTION | Forward  Out[b] = f(in[b])  in[BB] = Ù out(pred[B]) |
| TRANSFER FUNCTION | f(x) = gen[B] È (x - kill[B]) |
| MEET OPERAND (Ù) | Intersezione Ç |
| BOUNDARY CONDITION | Out[entry] = {entry} |
| INITIAL INTERIOR POINTS | Out[BB] = U (universal set) |

Specifichiamo la funzione di trasferimento nel seguente modo:

* Gen[B]: genera sé stesso, poiché per definizione il nodo si autodomina
* Kill[B]: sempre vuoto, in quanto non esiste modo di “killare” un dominatore

Pertanto, la funzione è semplificabile come:

* f(x) = gen[B] È x

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

|  |  |  |
| --- | --- | --- |
|  | **ITERAZIONE 1** | |
|  | **IN** | **OUT** |
| **BB1 (A)** | Æ | {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] Ç Out[E] = {A,C} | F U {A,C} = {A,C,F} |
| **BB7 (G)** | Out[B] Ç Out[F] = {A} | G U A = {G,A} |

**ESERCIZIO 3 – 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] = Ù out(pred[B]) |
| TRANSFER FUNCTION | f(x) = gen[B] È (x - kill[B]) |
| MEET OPERAND (Ù) | Intersezione Ç |
| BOUNDARY CONDITION | Out[entry] = Æ |
| INITIAL INTERIOR POINTS | Out[BB] = U (universal set) |

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

* Gen = es.: ‘x = y op. Z’, se gli operandi sono costanti e/o sono presenti nelle coppie in input, allora si genera <x, c>, dove c è il risultato dell’operazione.
* Kill = se viene generata la coppia <x,c>, si killa l’eventuale coppia in input <x, c2>
* Univ.Set = Dato un insieme generico {<x,c>, <y,c1>} Ç univ.Set, allora l’insieme risultante corrisponde all’insieme di partenza
* Intersezione = {<x,c1>, <y,c2>} Ç {<x, c1,>, <y, c3>} = {<x,c1>}. L’insieme risultante e’ l’insieme di coppie che hanno stessa costante e stessa variabile.

Il grafico delle iterazioni per questo esercizio è il seguente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ITERAZIONE 1 | | ITERAZIONE 2 | |
|  | IN | OUT | IN | OUT |
| Entry | Æ | Æ | Æ | Æ |
| K=2 | Æ | {k,2} | invariato | invariato |
| If | {k,2} | {k,2} | invariato | invariato |
| a=k+2 | {k,2} | **({a,4}**, {k,2}) | invariato | invariato |
| B5 x=5 | ({a,4}, {k,2}) | ({a,4}, {k,2}, **{x,5}**) | invariato | invariato |
| a=k\*2 | {k,2} | (**{a,4},** {k,2}) | invariato | invariato |
| B7 x=8 | ({a,4}, {k,2}) | ({a,4}, {k,2}, **{x,8}**) | invariato | invariato |
| B8 k=a | Out[B5] Ç Out[B7] = ({k,2}, {a,4}) | **({k,4}**, {a,4}) | invariato | invariato |
| While | Out[B8] Ç Out[B13] = ({k,4}, {a,4}) | ({k,4}, {a,4}) | Out[B8] Ç Out[B13] = **{a,4}** | {a,4} |
| B=2 | ({k,4}, {a,4}) | ({k,4}, {a,4**},{b,2}**) | {a,4} | **{{b,2},** {a,4}} |
| X=a+k | ({k,4}, {a,4},{b,2}) | ({k,4}, {a,4},{b,2},{**x,8}**) | {{b,2}, {a,4}} | {{b,2}, {a,4}} |
| Y=a\*b | ({k,4}, {a,4},{b,2},{x,8}) | ({k,4},{a,4},{b,2},{x,8**},{y,8})** | {{b,2}, {a,4}} | {{b,2}, {a,4},**{y,8}}** |
| B13 K++ | ({k,4},{a,4},{b,2},{x,8},{y,8}) | **({k,5}**,{a,4},{b,2},{x,8},{y,8}) | {{b,2}, {a,4},**{y,8}}** | {{b,2}, {a,4},**{y,8}}** |
| Print(a+x) | ({k,4}, {a,4}) | ({k,4}, {a,4}) | **{a,4}** | **{a,4}** |
| exit | ({k,4}, {a,4}) | ({k,4}, {a,4}) | **{a,4}** | **{a,4}** |

Alla prossima iterazione:

In[“while”] = Out[B8] Ç Out[B13] = {a,4} per cui l’algoritmo è convergente.