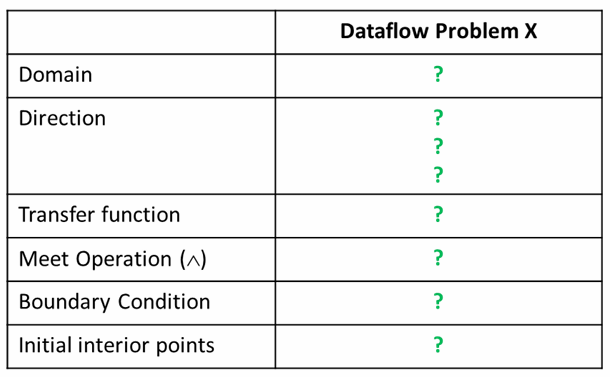
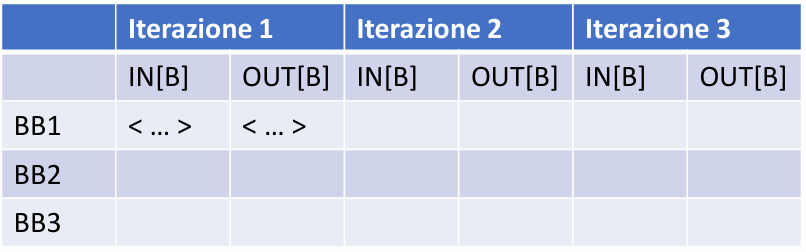
**Compilatori - Assignment 2**

Per ogni problema:

1. Derivare una formalizzazione per il framework di Dataflow Analysis, riempiendo lo specchietto coi parametri adeguati



1. Per il CFG di esempio fornito popolare una tabella con le iterazioni dell’algoritmo iterativo di soluzione del problema



**1 - Very Busy Expressions**

Un’espressione è very busy in un punto se, indipendentemente dal percorso preso da , l’espressione viene usata prima che uno dei suoi operandi venga definito.

* Un’espressione è very busy in un punto se è valutata in tutti i percorsi da a e non c’è una definizione di o 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

**1.A**

Ricordiamo che:

|  |  |
| --- | --- |
|  | **Very Busy Expression** |
| **Domain** | Sets of Expressions |
| **Direction** | Backward:  = |
| **Transfer Function** |  |
| **Meet Operation (**⋀**)** |  |
| **Boundary Condition** |  |
| **Initial Interior points** |  |

***Immagine che contiene testo, diagramma, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.*1.B**

**Risposta**: {b-a}, poiché è l’unica espressione utilizzata da BB2 a EXIT prima che uno dei suoi operandi venga definito

|  |  |  |
| --- | --- | --- |
|  | **Iterazione 1** | |
|  |  |  |
| BB1 = ENTRY | - | {b-a} |
| BB2 | {b-a} | {b-a} |
| BB3 | {b-a, a-b} | {a-b} |
| BB4 | {a-b} | Ø |
| BB5 | {b-a} | Ø |
| BB6 | Ø | {a-b} |
| BB7 | {a-b} | Ø |
| BB8 = EXIT | Ø | - |

**2 - 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 con un insieme

* se e solo se domina

Per definizione un nodo domina sé stesso

**2.A**

Ricordiamo che:

|  |  |
| --- | --- |
|  | **Dominator Analysis** |
| **Domain** | Sets of Basic Block |
| **Direction** | Forward: |
| **Transfer Function** |  |
| **Meet Operation (**⋀**)** |  |
| **Boundary Condition** |  |
| **Initial Interior points** |  |

Immagine che contiene diagramma, linea, Disegno tecnico

Il contenuto generato dall'IA potrebbe non essere corretto.**2.B**

|  |  |  |
| --- | --- | --- |
|  | **Iterazione 1** | |
|  |  |  |
| A = ENTRY | - | {A} |
| B | {A} | {A, B} |
| C | {A} | {A, C} |
| D | {A, C} | {A, C, D} |
| E | {A, C} | {A, C, E} |
| F | {A, C} | {A, C, F} |
| G = EXIT | {A} | - |
| A = ENTRY | - | {A} |

L’insieme dei dominatori di ogni nodo è l’insieme calcolato dal nostro algoritmo.

Non abbiamo usato le funzioni e perché ci basta considerare soltanto i Basic Blocks.

**3 - 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 del CFG è un insieme di coppie del tipo <variabile, valore costante>.

Se abbiamo la coppia <, > al nodo , significa che è garantito avere il valore ogni volta che viene raggiunto durante l’esecuzione del programma.

|  |
| --- |
| NOTA: L’analisi di CP riesce a determinare il valore costante di espressioni binarie in cui uno o entrambi gli operandi siano delle variabili il cui valore costante sia noto:   * w = 5 * x = 12 * y = x – 2 → y = 10 * z = w + x → z = 17   Tenere conto di questo aspetto nel determinare le equazioni. |

**3.A**

Ricordiamo che:

|  |  |
| --- | --- |
|  | **Constant Propagation** |
| **Domain** | Sets of pairs <variable, constant> |
| **Direction** | Forward: |
| **Transfer Function** |  |
| **Meet Operation (**⋀**)** |  |
| **Boundary Condition** |  |
| **Initial Interior points** |  |

**Immagine che contiene diagramma, Piano, Disegno tecnico, linea

Il contenuto generato dall'IA potrebbe non essere corretto.3.B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Iterazione 1** | | **Iterazione 2** | |
|  |  |  |  |  |
| ENTRY | - | Ø | - | Ø |
| k=2 | Ø | {(k,2)} | Ø | {(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)} | {(k,4), (a,4)} | {(k,2), (a,4)} | {(k,4), (a,4)} |
| while | {(k,4), (a,4)} | {(k,4), (a,4)} | {(a,4)} | {(a,4)} |
| b=2 | {(k,4), (a,4)} | {(k,4), (a,4), (b,2)} | {(a,4)} | {(a,4), (b,2)} |
| x=a+k | {(k,4), (a,4), (b,2)} | {(k,4), (a,4), (b,2), (x,8)} | {(a,4), (b,2)} | {(a,4), (b,2)} |
| y=a\*b | {(k,4), (a,4), (b,2), (x,8)} | {(k,4), (a,4), (b,2), (x,8), (y,8)} | {(a,4), (b,2)} | {(a,4), (b,2), (y,8)} |
| k++ | {(k,4), (a,4), (b,2), (x,8), (y,8)} | {(k,5), (a,4), (b,2), (x,8), (y,8)} | {(a,4), (b,2), (y,8)} | {(a,4), (b,2), (y,8)} |
| print(a+x) | {(k,4), (a,4)} | {(k,4), (a,4)} | {(a,4)} | {(a,4)} |
| EXIT | {(k,4), (a,4)} | - | {(a,4)} | - |

Nelle caselle evidenziate di giallo si può notare che non produciamo coppie poiché non è stata propagata in avanti, e perciò non è determinabile il valore di e di .