GDFA PROJ – 2

CS 738

Sandeep Dasgupta
Y9111031

Goal 1

- **Extend GDFA Architecture to Non-separable Framwork.**
  - To extend the bitvector framework where the dataflow information can be represented using bit vector but the frameworks are non-separable.
  - Implementing of ***faint variable analysis***, ***possibly uninitialised variable analysis*** using the extended framework.
  - The existing bit vector analysis should work fine with the extended framework.

Goal 2

- ❑ Output should be in
  - ▪ Textual format – print GIMPLE code lines showing impact chains.
  - ▪ Graphical format – Highlight the impact chains.

Some Concepts

Bit vector framework

Independence of data flow information of different entities.

$$f_n(X) = (X - Kill_n) \cup Gen_n$$

Non Separable Framewok

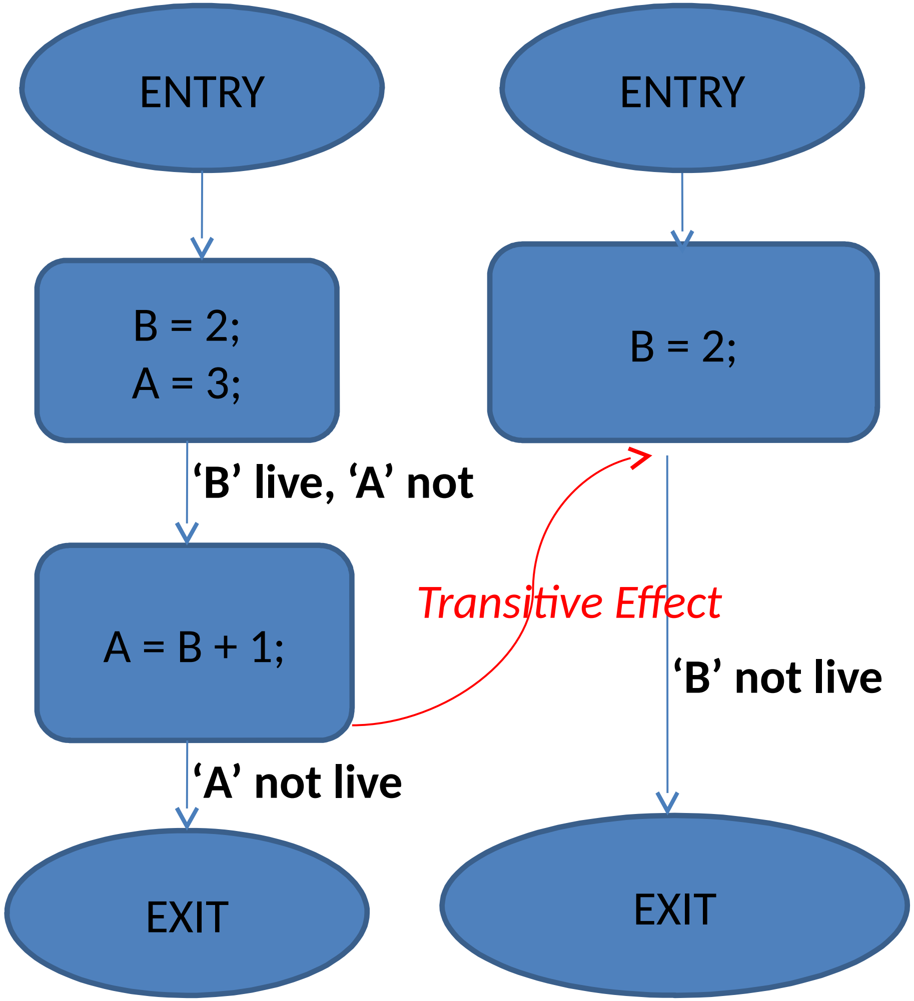
Data flow value of a given entity may depend on the data flow value of same entity or data flow value of some other entity.

$$Gen_n(x) = ConstGen_n \text{ U } DepGen_n\ (x)$$

$$Kill_n(x) = ConstKill_n \text{ U } DepKill_n\ (x)$$

$$f_n(X) = (X - Kill_n(x)) \cup Gen_n(x)$$

Transitive Effects : Impact Chains



ENTRY

B = 2;
A = 3;

'B' live, 'A' not

A = B + 1;

'A' not live

EXIT

ENTRY

B = 2;

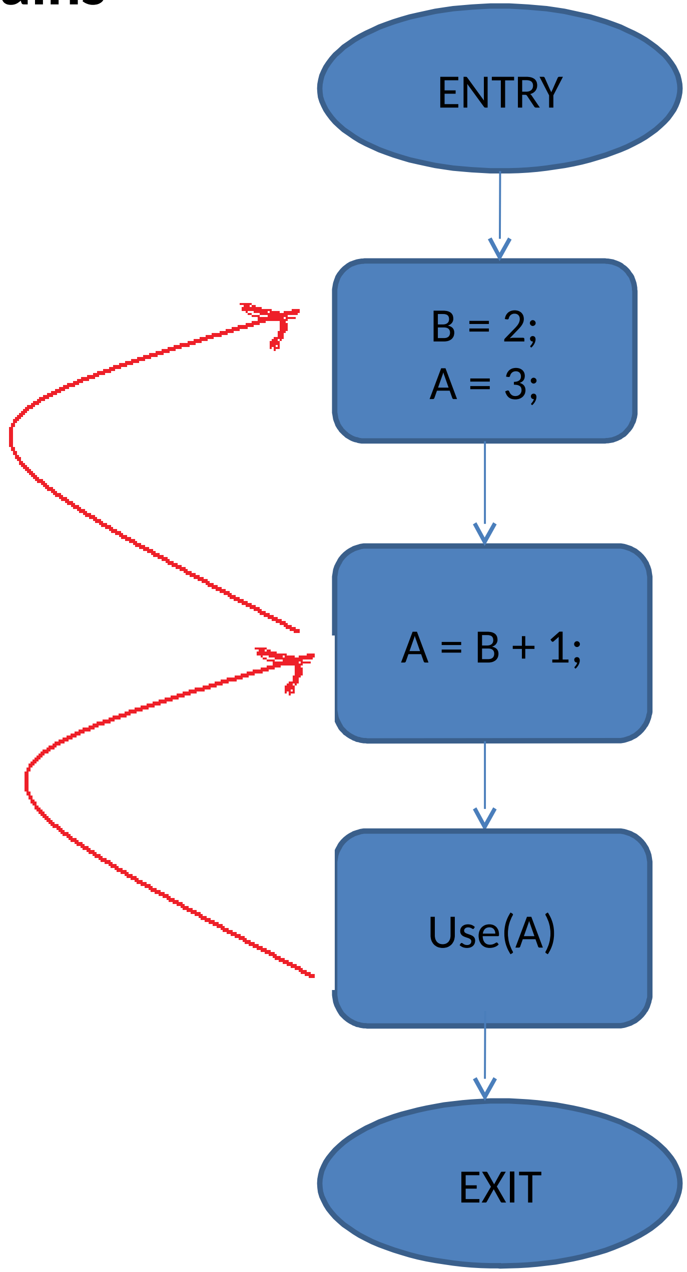
Transitive Effect

'B' not live

EXIT

Live Variable Analysis : Before Dead Code

Live Variable Analysis : After Dead Code Elimination



ENTRY

B = 2;
A = 3;

A = B + 1;

Use(A)

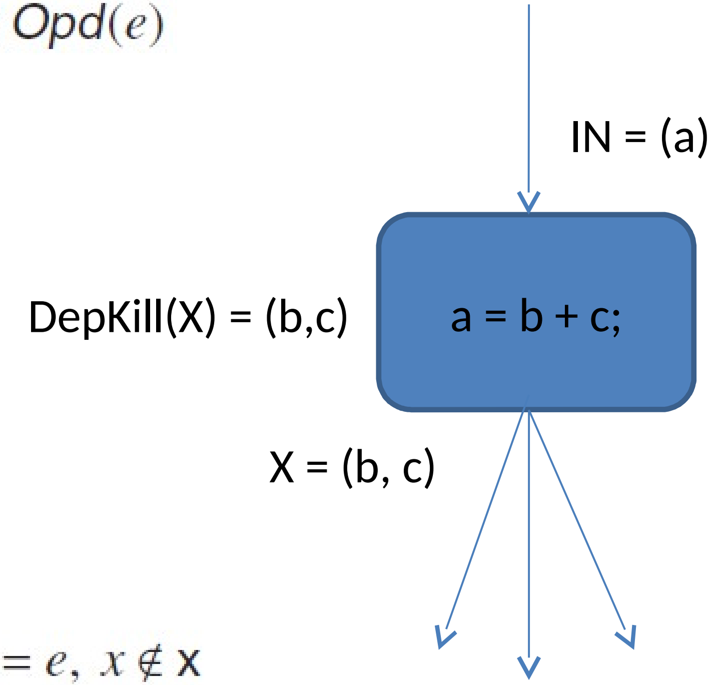
EXIT

Faint Variable Analysis

A variable x ∈ Var is faint at a program point u if along every path from u to End, it is either not used before being defined or is used to define a faint variable.

$$
\mathit{ConstGen}_n = \begin{cases} \{x\} & n \text{ is assignment } x = e,\ x \notin \mathit{Opd}(e) \\ \{x\} & n \text{ is } \mathit{read}(x) \\ \emptyset & \text{otherwise} \end{cases}
$$

$$
\mathit{DepGen}_n(\mathrm{x}) = \emptyset
$$



IN = (a)

DepKill(X) = (b,c)

a = b + c;

X = (b, c)

$$
\mathit{ConstKill}_n = \begin{cases} \{x\} & n \text{ is } \mathit{use}(x) \\ \emptyset & \text{otherwise} \end{cases}
$$

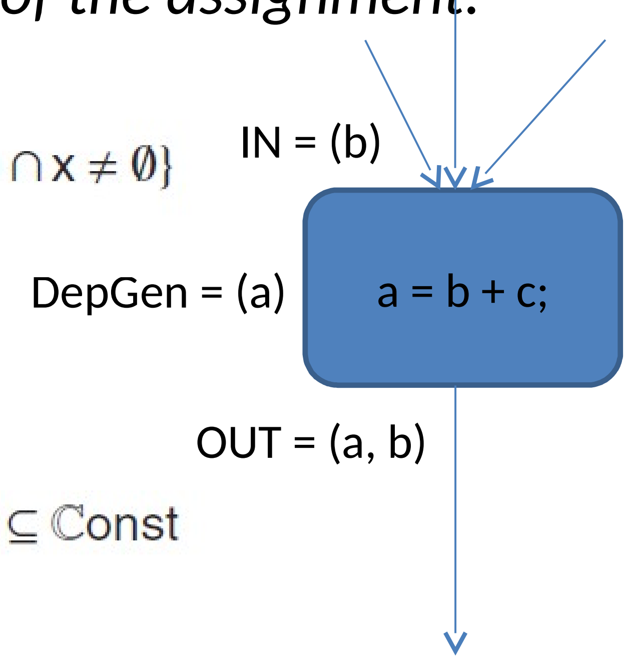
$$
\mathit{DepKill}_n(\mathrm{x}) = \begin{cases} \mathit{Opd}(e) \cap \mathbb{V}\mathrm{ar} & n \text{ is assignment } x = e,\ x \notin \mathrm{x} \\ \emptyset & \text{otherwise} \end{cases}
$$

Possibly Uninitialized Variable Analysis

A variable x ∈ Var is possibly uninitialized at a program point u if there exists a path from Start to u along which either no definition of the variable has been encountered or the definition uses a possibly uninitializedvariable on the right hand side of the assignment.

$$ConstGen_n = \emptyset$$

$$DepGen_n(\mathsf{x}) = \begin{cases} \{x\} & n \text{ is assignment } x = e,\ Opd(e) \cap \mathsf{x} \neq \emptyset\} \\ \emptyset & \text{otherwise} \end{cases}$$

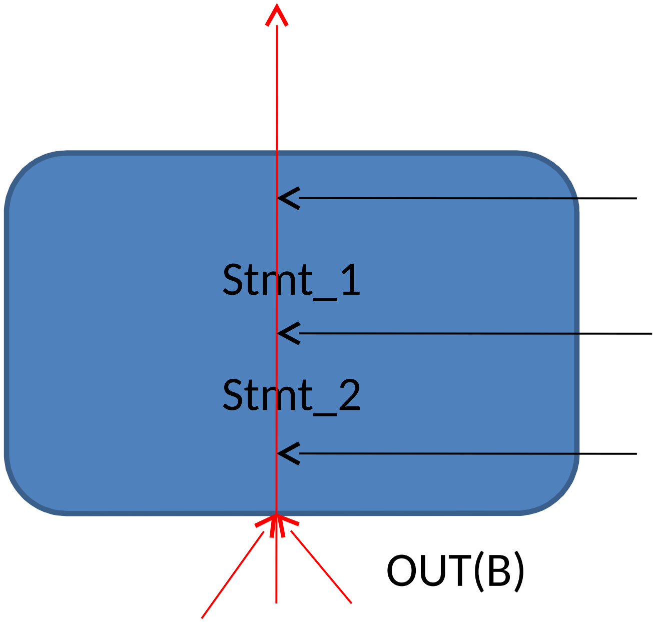


$$ConstKill_n = \begin{cases} \{x\} & n \text{ is assignment } x = e,\ Opd(e) \subseteq \mathbb{C}onst \\ \{x\} & n \text{ is } read(x) \\ \emptyset & \text{otherwise} \end{cases}$$

$$DepKill_n(\mathsf{x}) = \begin{cases} \{x\} & n \text{ is assignment } x = e,\ Opd(e) \cap \mathsf{x} = \emptyset\} \\ \emptyset & \text{otherwise} \end{cases}$$

Challenges

- **Basic blocks for non-separable analyses consist of single statements**
  - Local dfa : Calculate Const Gen and Const Kill of each statement.
  - Global dfa :



IN(Stmt_1) = (IN(Stmt_2) - KILL(Stmt_1)) U GEN(Stmt_1)

IN(Stmt_2) = (OUT(B) - KILL(Stmt_2)) U GEN(Stmt_2)

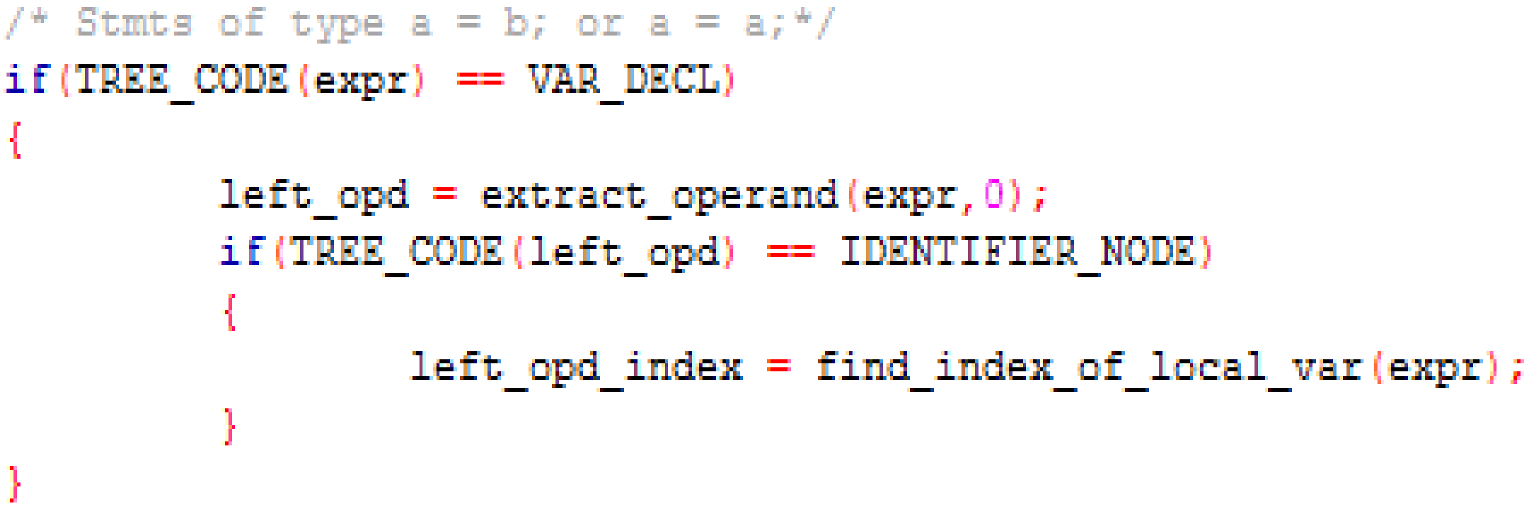
OUT(Stmt_2) = OUT(B)

GEN(Stmt_1) = ConstGen(Stmt_1) U DepGen $_{\text{Stmt\_1}}$(IN(Stmt_2))
KILL(Stmt_1) = ConstKill(Stmt_1) U DepKill $_{\text{Stmt\_1}}$(IN(Stmt_2))

GEN(Stmt_2) = ConstGen(Stmt_2) U DepGen $_{\text{Stmt\_2}}$((OUT(B))
KILL(Stmt_2) = ConstKill(Stmt_2) U DepKill $_{\text{Stmt\_2}}$((OUT(B))

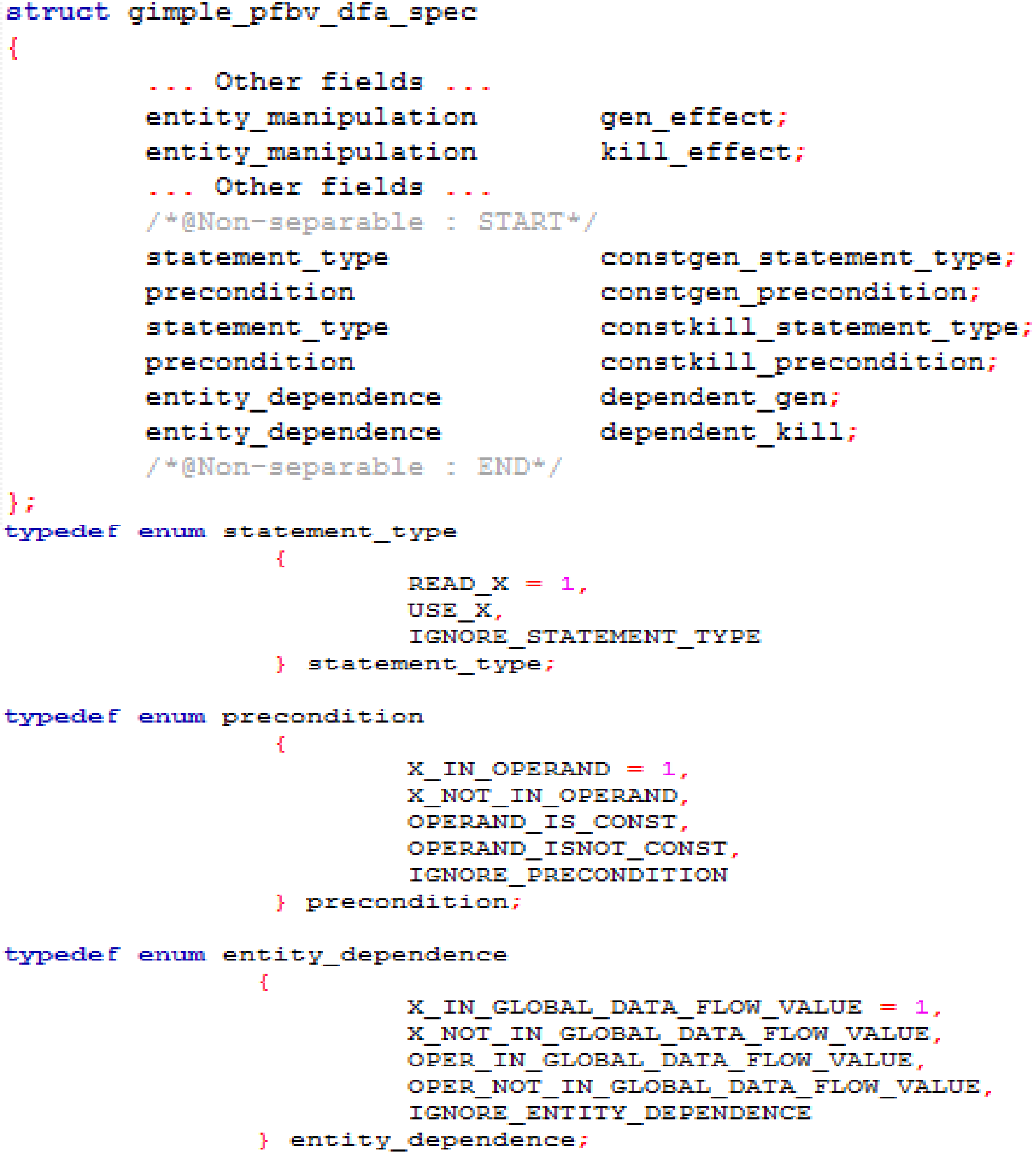
Challenges

- **To include some other statement types in the analysis domain.**
  - ➢ Copy statements :



```
/* Stmts of type a = b; or a = a;*/
if(TREE_CODE(expr) == VAR_DECL)
{
         left_opd = extract_operand(expr,0);
         if(TREE_CODE(left_opd) == IDENTIFIER_NODE)
         {
                 left_opd_index = find_index_of_local_var(expr);
         }
}
```

  - ➢ Indexing the statements keeping intact the indexing for reaching definition analysis.
  - ➢ Operand extraction from Scanf & Printf not yet done.



```
struct gimple_pfbv_dfa_spec
{
        ... Other fields ...
        entity_manipulation       gen_effect;
        entity_manipulation       kill_effect;
        ... Other fields ...
        /*@Non-separable : START*/
        statement_type            constgen_statement_type;
        precondition              constgen_precondition;
        statement_type            constkill_statement_type;
        precondition              constkill_precondition;
        entity_dependence         dependent_gen;
        entity_dependence         dependent_kill;
        /*@Non-separable : END*/
};
typedef enum statement_type
                {
                        READ_X = 1,
                        USE_X,
                        IGNORE_STATEMENT_TYPE
                } statement_type;

typedef enum precondition
                {
                        X_IN_OPERAND = 1,
                        X_NOT_IN_OPERAND,
                        OPERAND_IS_CONST,
                        OPERAND_ISNOT_CONST,
                        IGNORE_PRECONDITION
                } precondition;

typedef enum entity_dependence
               {
                        X_IN_GLOBAL_DATA_FLOW_VALUE = 1,
                        X_NOT_IN_GLOBAL_DATA_FLOW_VALUE,
                        OPER_IN_GLOBAL_DATA_FLOW_VALUE,
                        OPER_NOT_IN_GLOBAL_DATA_FLOW_VALUE,
                        IGNORE_ENTITY_DEPENDENCE
               } entity_dependence;
```

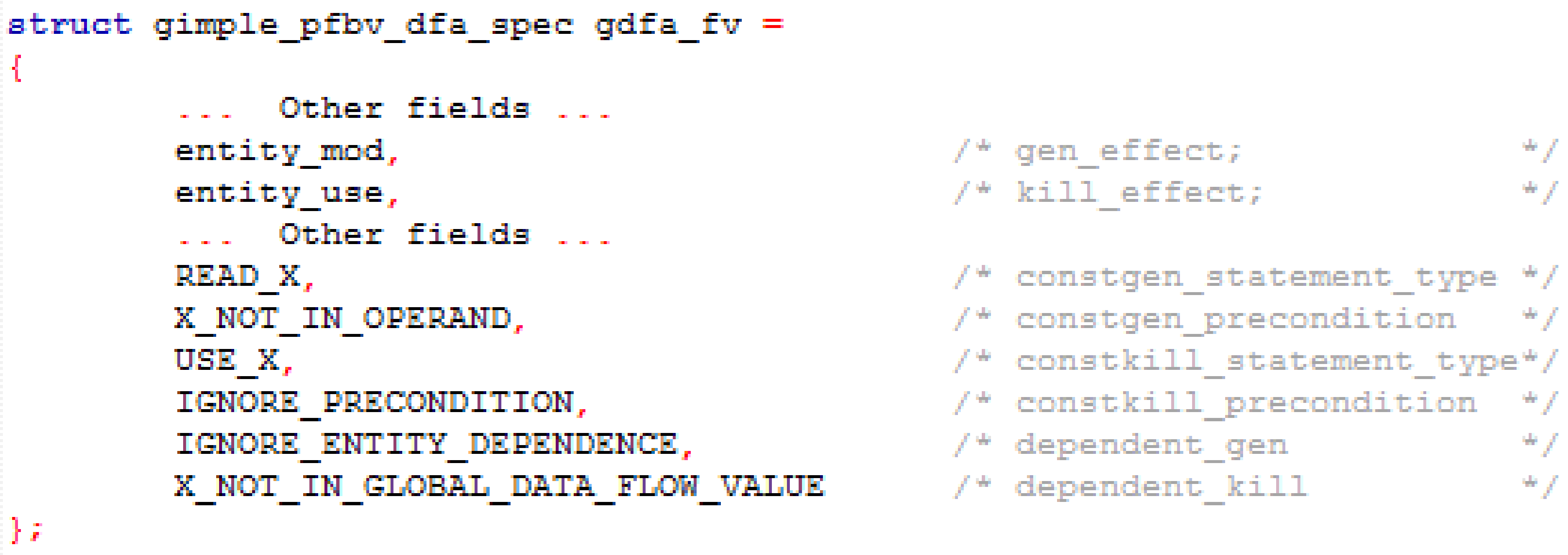
Specification Structure For Faint Variable Analysis

$$\mathit{ConstGen}_n = \begin{cases} \{x\} & n \text{ is assignment } x = e,\ x \notin \mathit{Opd}(e) \\ \{x\} & n \text{ is } \mathit{read}(x) \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathit{DepGen}_n(\mathsf{x}) = \emptyset$$

$$\mathit{ConstKill}_n = \begin{cases} \{x\} & n \text{ is } \mathit{use}(x) \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathit{DepKill}_n(\mathsf{x}) = \begin{cases} \mathit{Opd}(e) \cap \mathbb{V}\mathsf{ar} & n \text{ is assignment } x = e,\ x \notin \mathsf{x} \\ \emptyset & \text{otherwise} \end{cases}$$



```
struct gimple_pfbv_dfa_spec gdfa_fv =
{
        ...   Other fields ...
        entity_mod,                                 /* gen_effect;               */
        entity_use,                                 /* kill_effect;              */
        ...   Other fields ...
        READ_X,                                     /* constgen_statement_type */
        X_NOT_IN_OPERAND,                           /* constgen_precondition   */
        USE_X,                                      /* constkill_statement_type*/
        IGNORE_PRECONDITION,                        /* constkill_precondition  */
        IGNORE_ENTITY_DEPENDENCE,                   /* dependent_gen           */
        X_NOT_IN_GLOBAL_DATA_FLOW_VALUE             /* dependent_kill          */
};
```

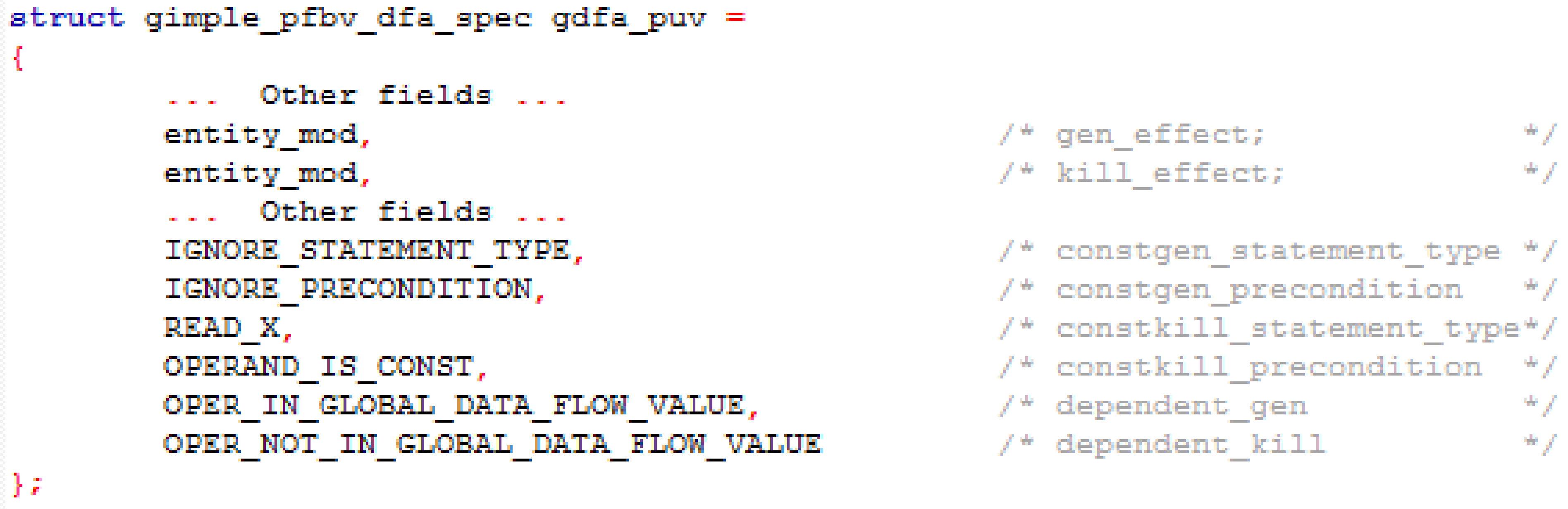
Specification Structure For UnInitialized Variable Analysis

$$ConstGen_n = \emptyset$$

$$DepGen_n(\mathsf{x}) = \begin{cases} \{x\} & n \text{ is assignment } x = e,\ Opd(e) \cap \mathsf{x} \neq \emptyset\} \\ \emptyset & \text{otherwise} \end{cases}$$

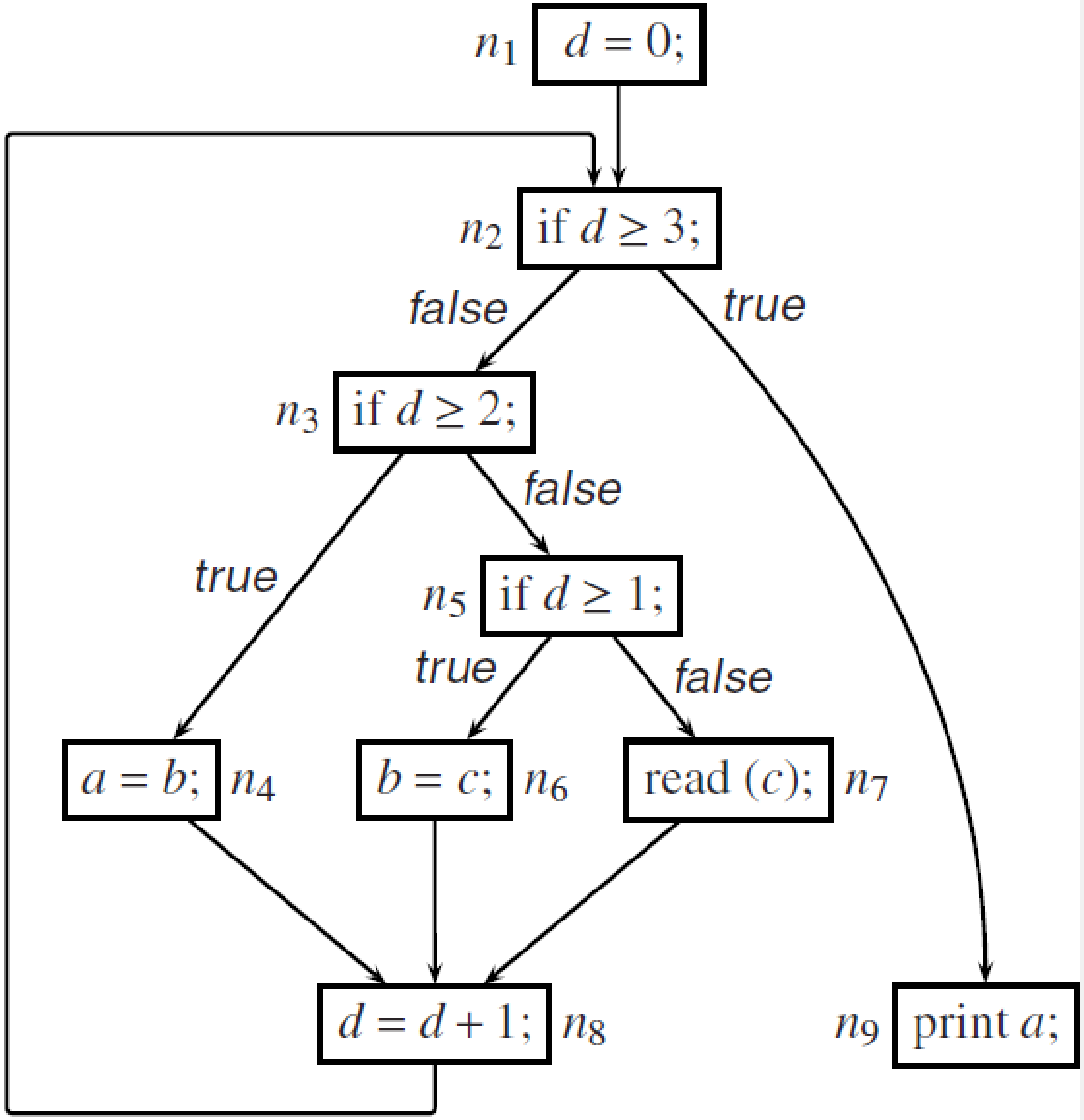
$$ConstKill_n = \begin{cases} \{x\} & n \text{ is assignment } x = e,\ Opd(e) \subseteq \mathbb{C}\mathrm{onst} \\ \{x\} & n \text{ is } \mathit{read}(x) \\ \emptyset & \text{otherwise} \end{cases}$$

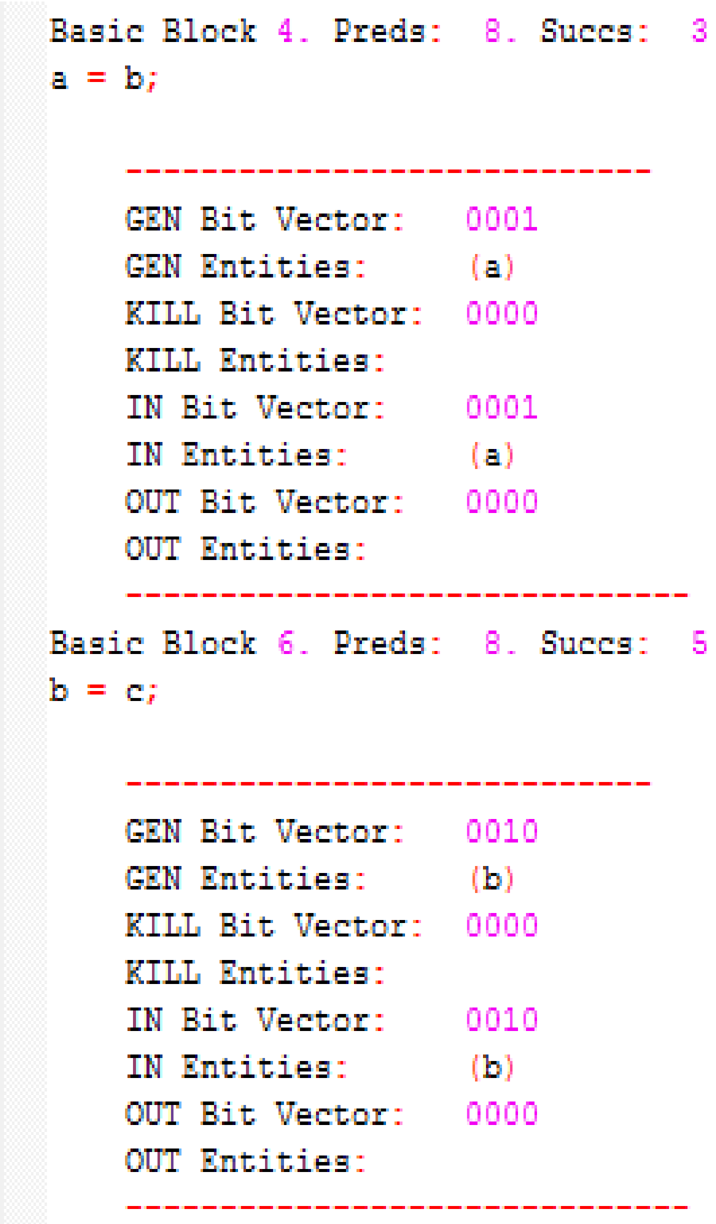
$$DepKill_n(\mathsf{x}) = \begin{cases} \{x\} & n \text{ is assignment } x = e,\ Opd(e) \cap \mathsf{x} = \emptyset\} \\ \emptyset & \text{otherwise} \end{cases}$$



```
struct gimple_pfbv_dfa_spec gdfa_puv =
{
        ...  Other fields ...
        entity_mod,                                         /* gen_effect;            */
        entity_mod,                                         /* kill_effect;           */
        ...  Other fields ...
        IGNORE_STATEMENT_TYPE,                              /* constgen_statement_type */
        IGNORE_PRECONDITION,                                /* constgen_precondition   */
        READ_X,                                             /* constkill_statement_type*/
        OPERAND_IS_CONST,                                   /* constkill_precondition  */
        OPER_IN_GLOBAL_DATA_FLOW_VALUE,                     /* dependent_gen           */
        OPER_NOT_IN_GLOBAL_DATA_FLOW_VALUE                  /* dependent_kill          */
};
```

Results 1 : Textual Format



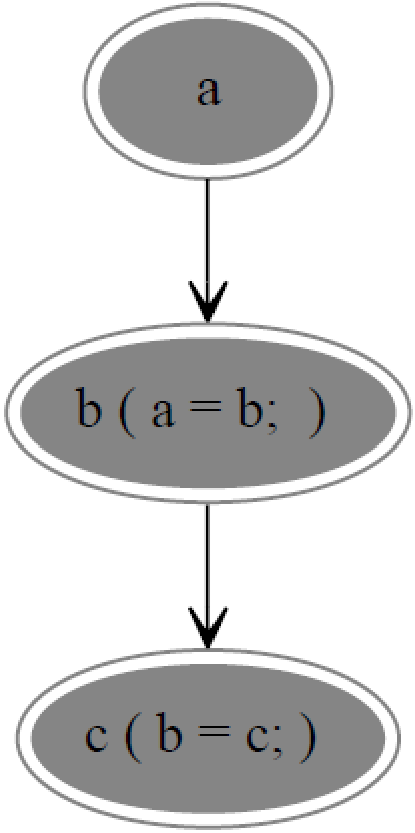


```
Basic Block 4. Preds:  8. Succs:  3
a = b;

    ----------------------------
    GEN Bit Vector:   0001
    GEN Entities:     (a)
    KILL Bit Vector:  0000
    KILL Entities:
    IN Bit Vector:    0001
    IN Entities:      (a)
    OUT Bit Vector:   0000
    OUT Entities:
    ------------------------------
Basic Block 6. Preds:  8. Succs:  5
b = c;

    ----------------------------
    GEN Bit Vector:   0010
    GEN Entities:     (b)
    KILL Bit Vector:  0000
    KILL Entities:
    IN Bit Vector:    0010
    IN Entities:      (b)
    OUT Bit Vector:   0000
    OUT Entities:
    ------------------------------
```

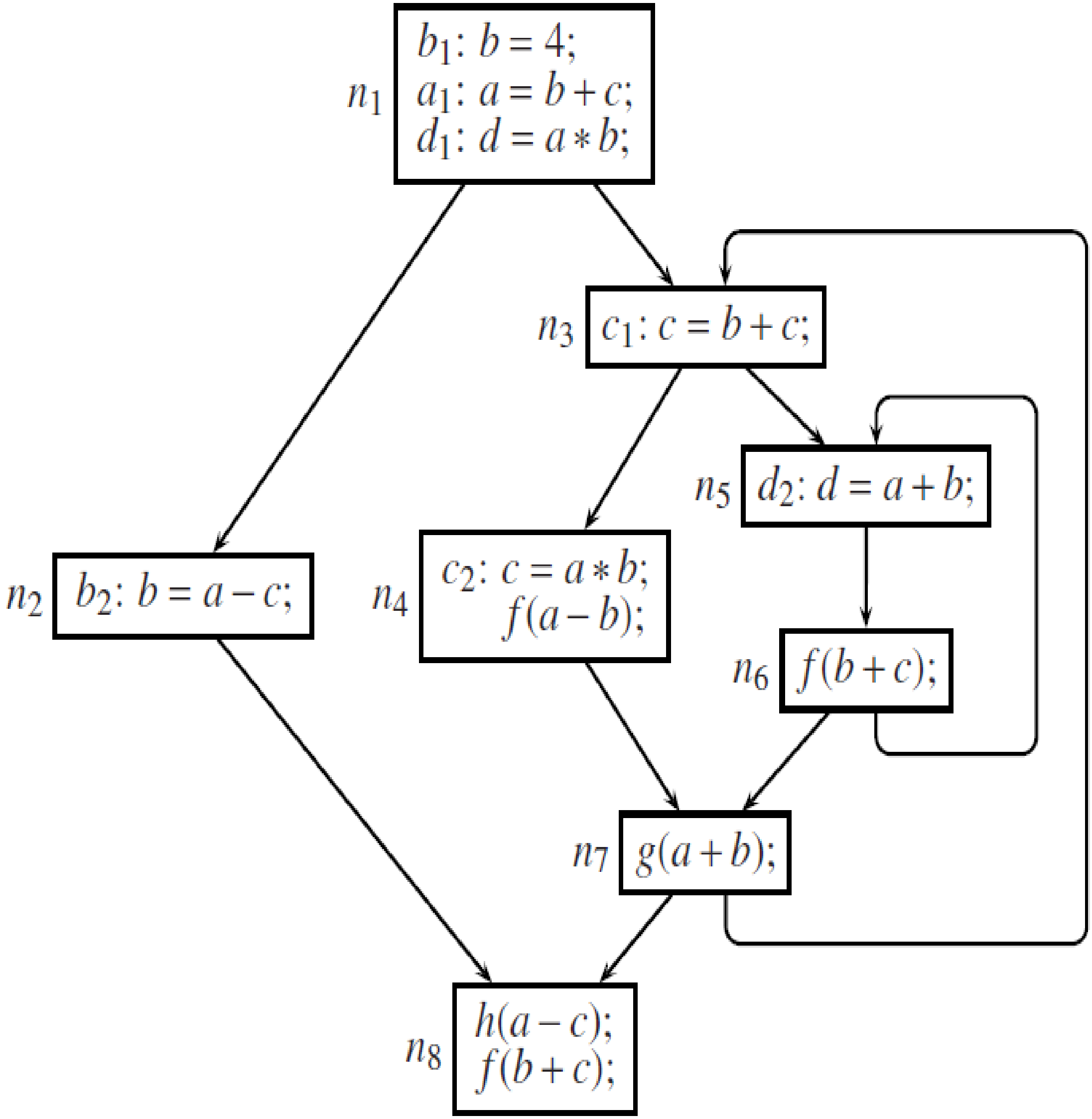
Results 1: Graphical Format

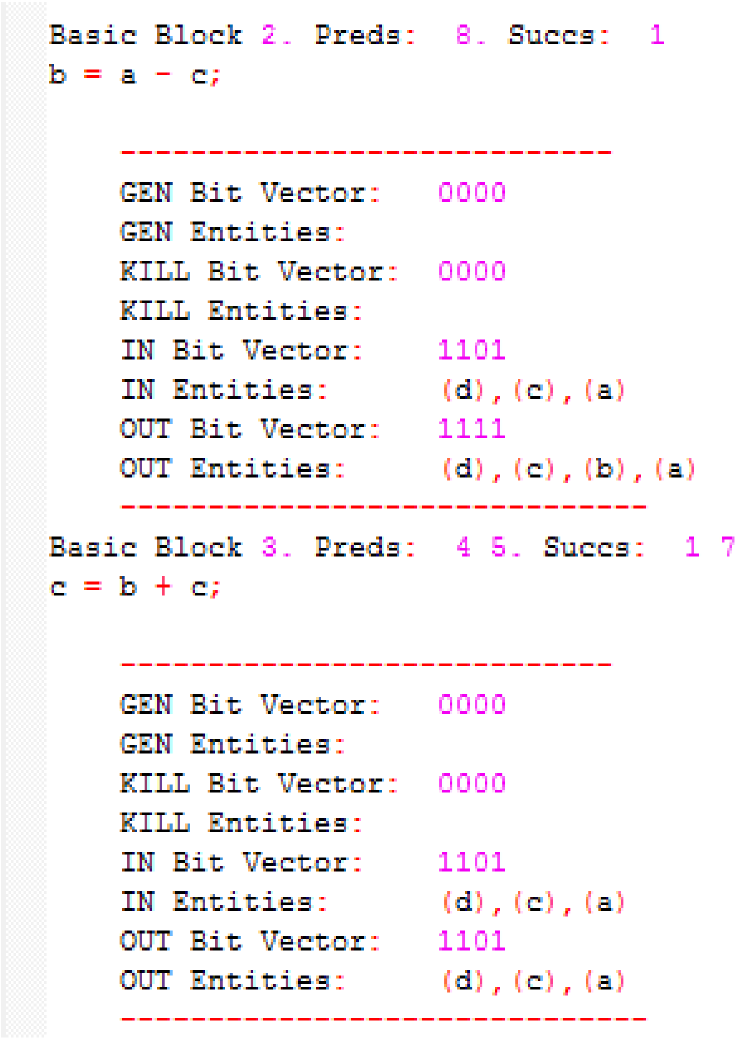


Kill Impact Chain For Faint Variable Analysis

Using Dot tool of ***GrapghViz Software***

Results 2 : Textual Format



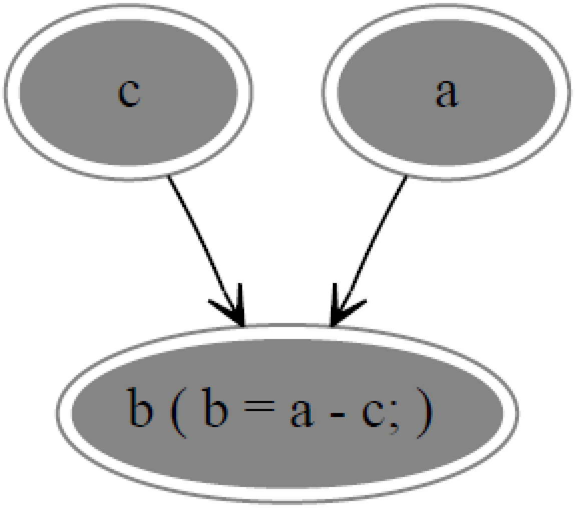


```
Basic Block 2. Preds:  8. Succs:  1
b = a - c;

    ----------------------------
    GEN Bit Vector:    0000
    GEN Entities:
    KILL Bit Vector:   0000
    KILL Entities:
    IN Bit Vector:     1101
    IN Entities:       (d),(c),(a)
    OUT Bit Vector:    1111
    OUT Entities:      (d),(c),(b),(a)
    ------------------------------
Basic Block 3. Preds:  4 5. Succs:  1 7
c = b + c;

    ----------------------------
    GEN Bit Vector:    0000
    GEN Entities:
    KILL Bit Vector:   0000
    KILL Entities:
    IN Bit Vector:     1101
    IN Entities:       (d),(c),(a)
    OUT Bit Vector:    1101
    OUT Entities:      (d),(c),(a)
    ------------------------------
```

Results 2: Graphical Format



Gen Impact Chain For UnInitialized Variable Analysis

Using Dot tool of ***GrapghViz Software***

THANK YOU