GDFA PROJ - 2

Sandeep Dasgupta
Y9111031

Some Concepts

Bit vector framework

Independence of data flow information of different entities.

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

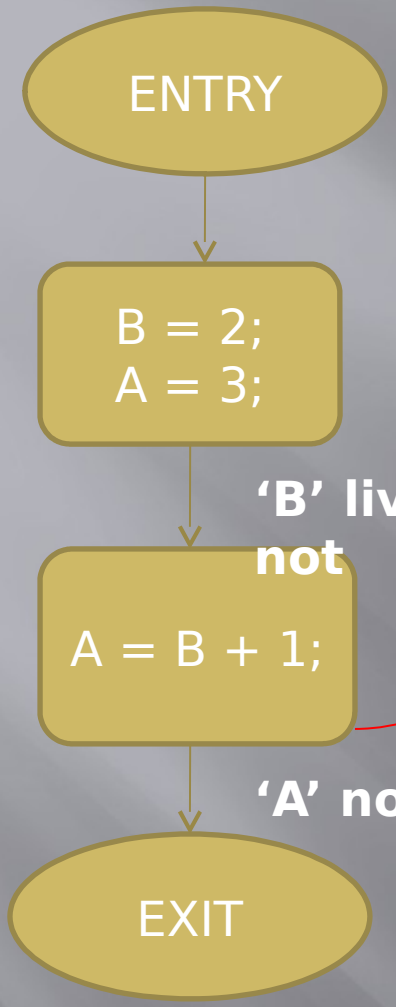
Non Separable Framewok

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.

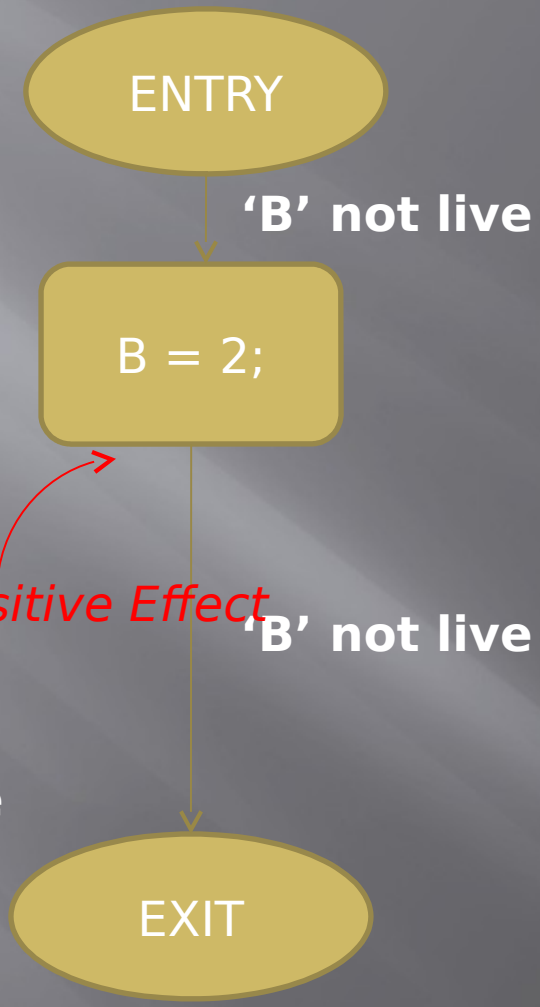
$$Gen_n(x) = ConstGen_n \cup DepGen_n(x)$$

$$Kill_n(x) = ConstKill_n \cup DepKill_n(x)$$

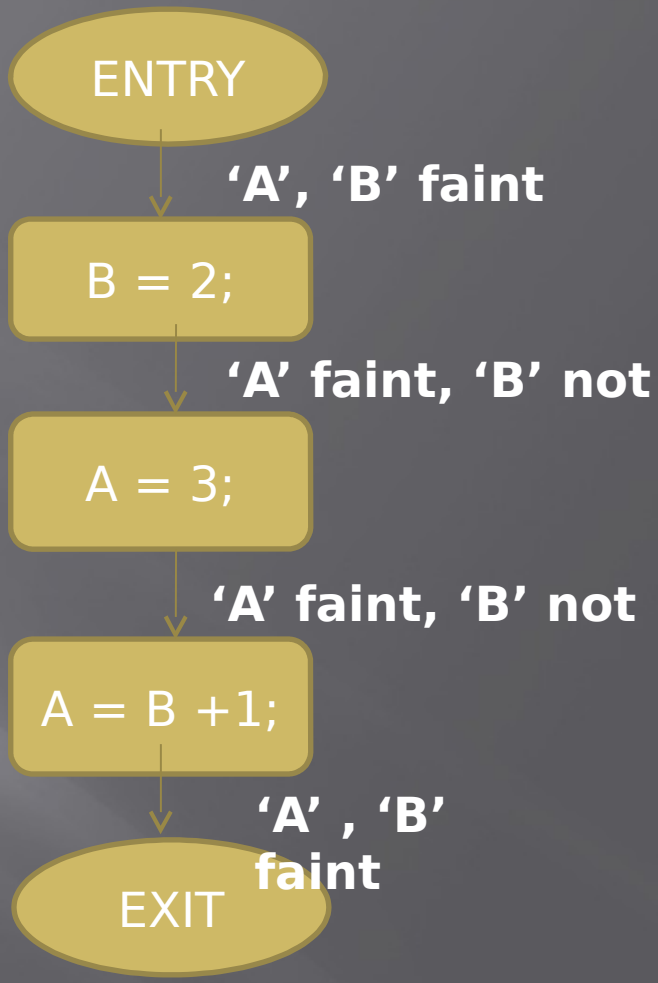
$$f_n(X) = (X - Kill_n(x)) \cup Gen_n(x)$$



Live Variable Analysis : Before Dead Code Elimination



Live Variable Analysis : After Dead Code Elimination



Faint Variable Analysis -- ***Non-Separable framework***

Status of gdfa in GCC

- **Intraprocedural bit vector data flow analysis.**
- Gdfa uses generic data flow equations which makes it possible to execute a wide variety of specifications without having to know the name med.

```
 0 struct gimple_pfbv_dfa_spec gdfa_ave =
 1 {
 2       entity_expr,                  /* entity              */
 3       ONES,                         /* top_value           */
 4       ZEROS,                        /* entry_info          */
 5       ONES,                         /* exit_info           */
 6       FORWARD,                      /* traversal_order     */
 7       INTERSECTION,                 /* confluence          */
 8       entity_use,                   /* gen_effect          */
 9       down_exp,                     /* gen_exposition      */
10       entity_mod,                   /* kill_effect         */
11       any_where,                    /* kill_exposition     */
12       global_only,                  /* preserved_dfi       */
13       identity_forward_edge_flow,   /* forward_edge_flow   */
14       stop_flow_along_edge,         /* backward_edge_flow  */
15       forward_gen_kill_node_flow,   /* forward_node_flow   */
16       stop_flow_along_node          /* backward_node_flow  */
17 };
```

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

Goal

- Extend gdfa to the dataflow frameworks where flow information can be represented using bit vector but the frameworks are non-separable.
- Output should be in
  - Textual format – print GIMPLE code lines showing impact chains.
  - Graphical format – show the entire control flow graph with basic block/gimple statements in the impact chain highlighted.

An Approach

- $Gen_n(x) = ConstGen_n \cup DepGen_n(x)$

  $Kill_n(x) = ConstKill_n \cup DepKill_n(x)$

  $f_n(X) = (X - Kill_n(x)) \cup Gen_n(x)$
- To treat a statement as an independent basic block.
- To debug one particular analysis(live variable analysis) to know the nitty gritty of the code.
- The dependent part of Gen Kill need to be implemented in gdfa.
- The specification structure need to be updated. We propose introduction of four more fields in the existing structure.

Expectations

- The existing bit vector analysis should work fine with the new framework (passing void in the DepGen, DepKill form).
- Implementation of ***faint variable analysis***, ***possibly uninitialised variable analysis*** successfully.