Workshop on Essential Abstractions in GCC

GDFA: Generic Data Flow Analyser for GCC

GCC Resource Center

(www.cse.iitb.ac.in/grc)

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

July 2009

Outline

- Motivation
- Introduction to data flow analysis
  - Live variables analysis
  - Available expressions analysis
- Common abstractions in data flow analysis
- Implementing data flow analysis using *gdfa*
- Design and Implementation of *gdfa*

Part 1

Introduction to Data Flow Analysis

Motivation behind gdfa

- Specification Vs. implementation
- Orthogonality of specification of data flow analysis and the process of performing data flow analysis
- Practical significance of generalizations
- Ease of extending data flow analysers

July 09 GDFA: Introduction to Data Flow Analysis 2/37

Motivation behind gdfa

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

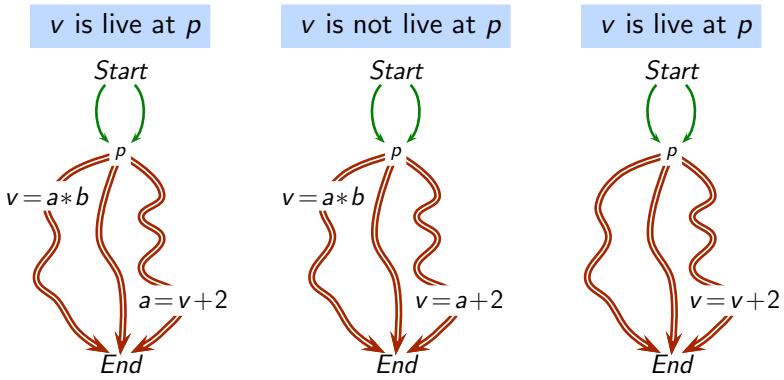
Part 2

Introduction to Data Flow Analysis

Defining Live Variables Analysis

A variable v is live at a program point p, if some path from p to program exit contains an r-value occurrence of v which is not preceded by an l-value occurrence of v.

Path based specification



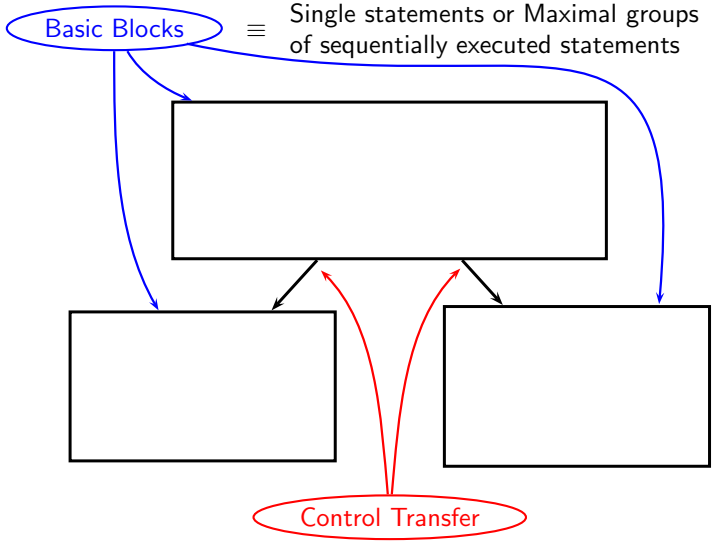
Notes

July 09 GDFA: Introduction to Data Flow Analysis 3/37

Defining Live Variables Analysis

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Defining Data Flow Analysis for Live Variables Analysis



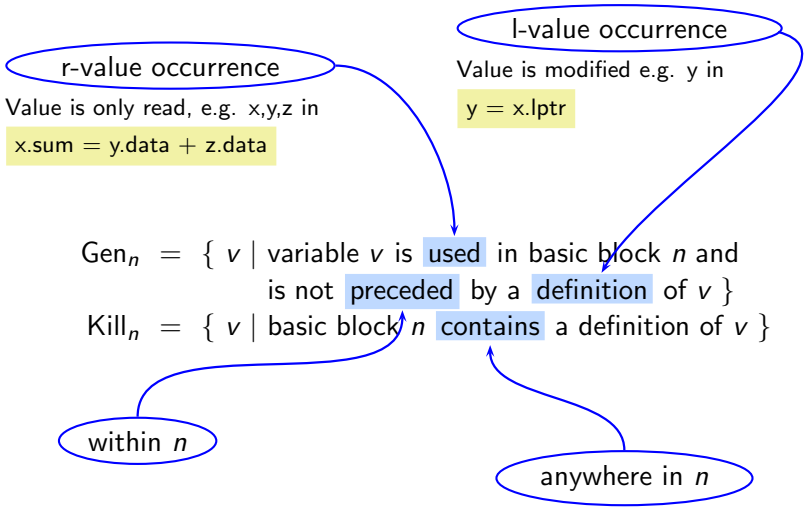
Notes

July 09 GDFA: Introduction to Data Flow Analysis 4/37

Defining Data Flow Analysis for Live Variables Analysis

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Local Data Flow Properties for Live Variables Analysis



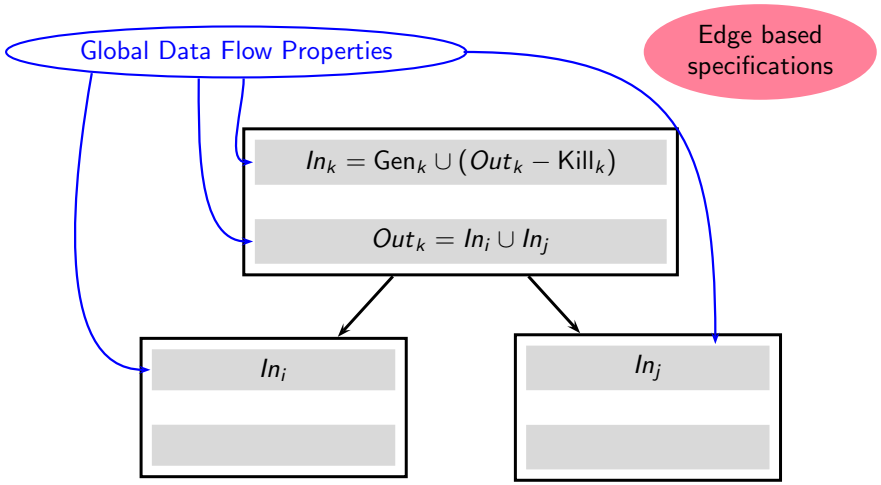
July 09 GDFA: Introduction to Data Flow Analysis 5/37

Local Data Flow Properties for Live Variables Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Defining Data Flow Analysis for Live Variables Analysis



July 09 GDFA: Introduction to Data Flow Analysis 6/37

Defining Data Flow Analysis for Live Variables Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Data Flow Equations For Live Variables Analysis

$$In_n = (Out_n - \text{Kill}_n) \cup \text{Gen}_n$$

$$Out_n = \begin{cases} BI\ n \text{ is } End \text{ block} \\ \bigcup_{s \in succ(n)} In_s & \text{otherwise} \end{cases}$$

In_n and Out_n are sets of variables.

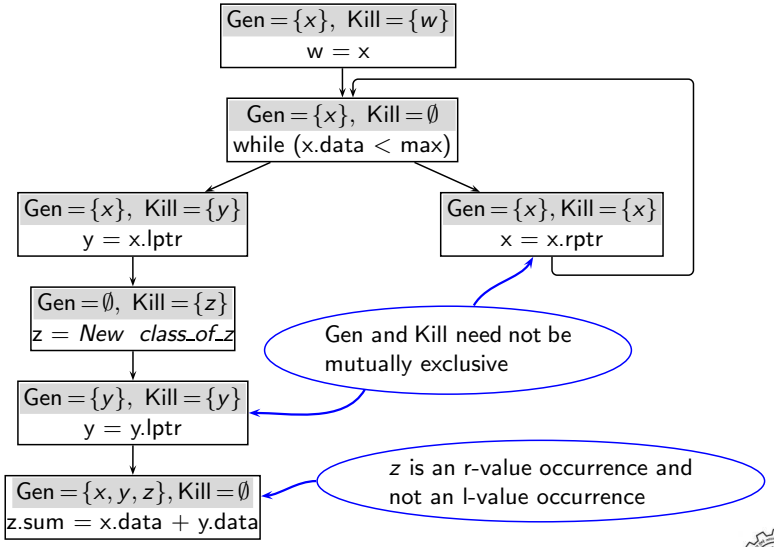
July 09 GDFA: Introduction to Data Flow Analysis 7/37

Data Flow Equations For Live Variables Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Performing Live Variables Analysis



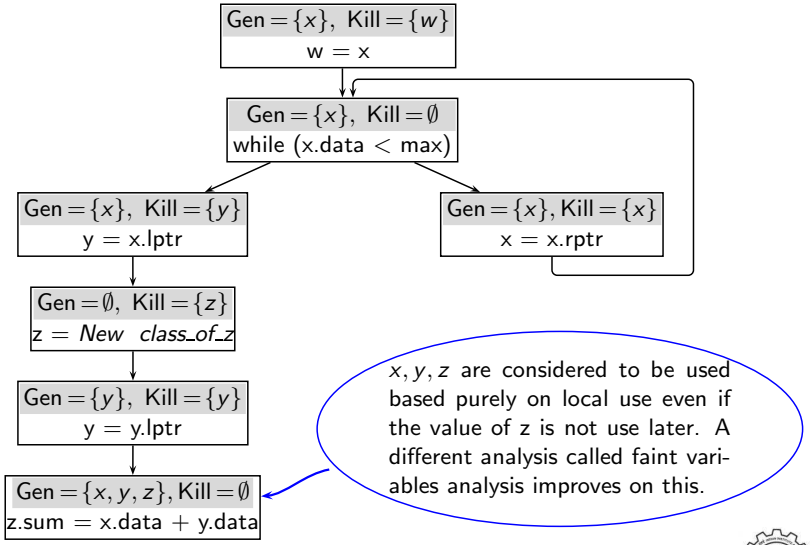
Essential Abstrations in GCC GCC Resource Center, IIT Bombay

July 09 GDFA: Introduction to Data Flow Analysis 8/37

Performing Live Variables Analysis

Notes

Performing Live Variables Analysis



Notes

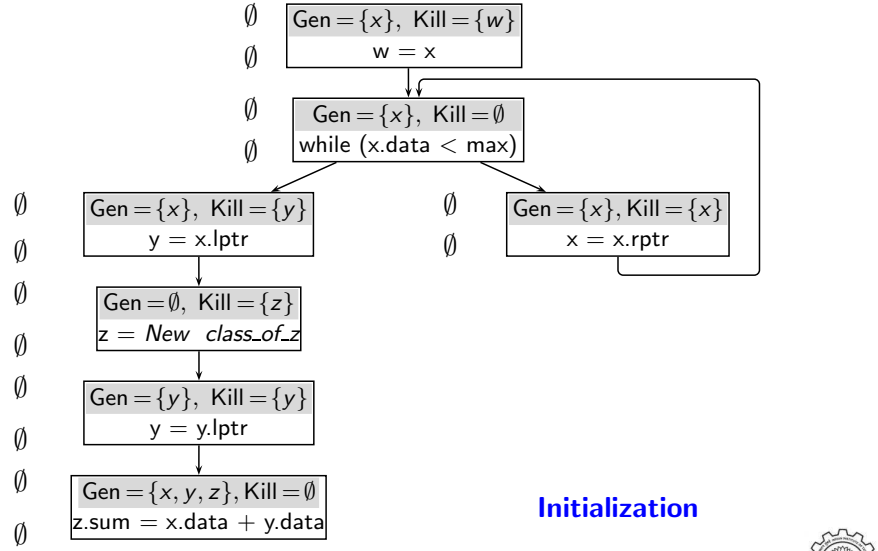
July 09 GDFA: Introduction to Data Flow Analysis 8/37

Performing Live Variables Analysis

Notes

July 09 GDFA: Introduction to Data Flow Analysis 8/37

Performing Live Variables Analysis



Essential Abstrations in GCC GCC Resource Center, IIT Bombay

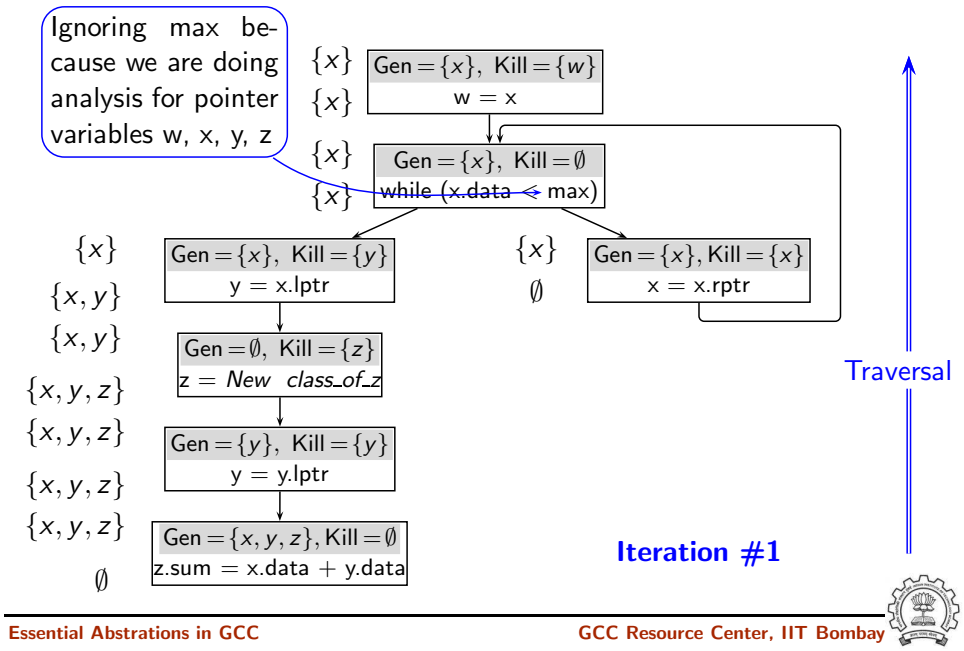
Notes

July 09 GDFA: Introduction to Data Flow Analysis 8/37

Performing Live Variables Analysis

Notes

Performing Live Variables Analysis



Notes

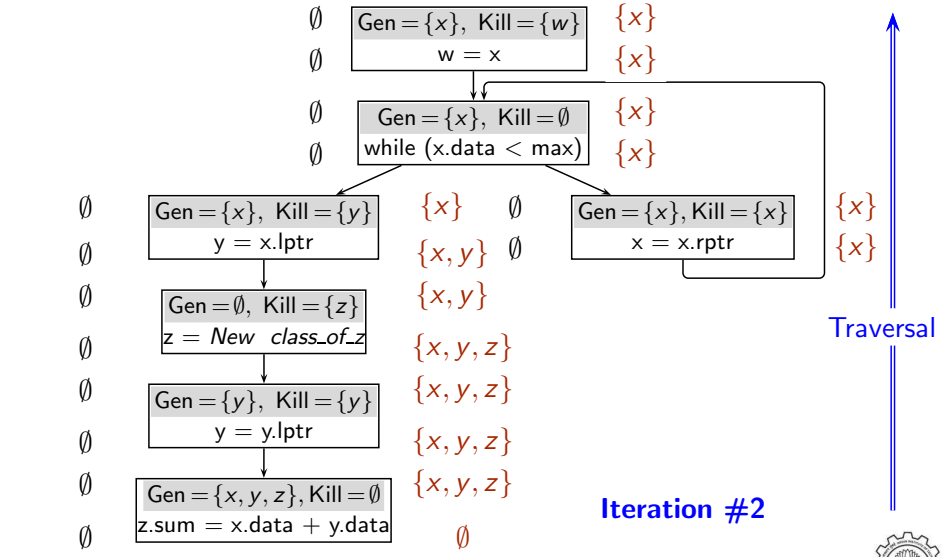
July 09 GDFA: Introduction to Data Flow Analysis 8/37

Performing Live Variables Analysis

Notes

July 09 GDFA: Introduction to Data Flow Analysis 8/37

Performing Live Variables Analysis



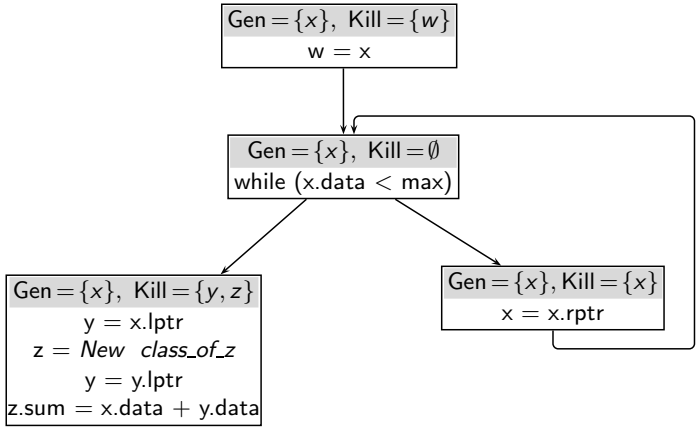
Notes

July 09 GDFA: Introduction to Data Flow Analysis 8/37

Performing Live Variables Analysis

Notes

Performing Live Variables Analysis



Performing Live Variables Analysis

Notes

Using Data Flow Information of Live Variables Analysis

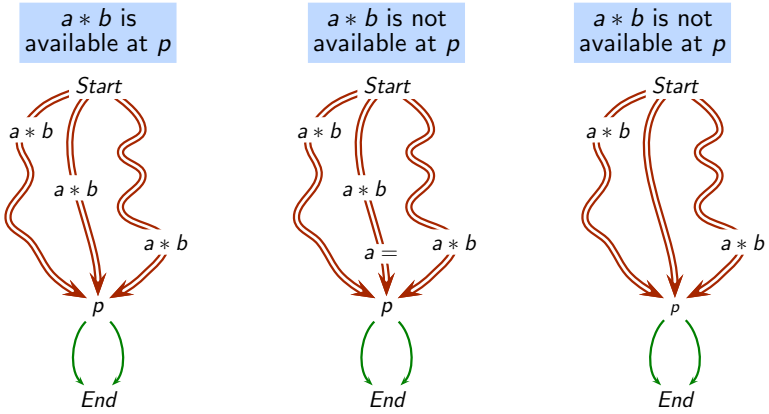
- Used for register allocation.
  If variable x is live in a basic block b, it is a potential candidate for register allocation.
- Used for dead code elimination.
  If variable x is not live after an assignment $x = \ldots$, then the assginment is redundant and can be deleted as dead code.

Using Data Flow Information of Live Variables Analysis

Notes

Defining Available Expressions Analysis

An expression e is available at a program point p, if
every path from program entry to p contains an evaluation of e
which is not followed by a definition of any operand of e.



July 09 GDFA: Introduction to Data Flow Analysis 11/37

Defining Available Expressions Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Local Data Flow Properties for Available Expressions Analysis

Gen_n = { e | expression e is evaluated in basic block n and this evaluation is not followed by a definition of any operand of e}

Kill_n = { e | basic block n contains a definition of an operand of e}

July 09 GDFA: Introduction to Data Flow Analysis 12/37

Local Data Flow Properties for Available Expressions Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Data Flow Equations For Available Expressions Analysis

$$In_n = \begin{cases} BIn \text{ is } Start \text{ block} \\ \bigcap_{p \in pred(n)} Out_p \quad \text{otherwise} \end{cases}$$

$$Out_n = \text{Gen}_n \cup (In_n - \text{Kill}_n)$$

Alternatively,

$$Out_n = f_n(In_n), \quad \text{where}$$

$$f_n(X) = \text{Gen}_n \cup (X - \text{Kill}_n)$$

In_n and Out_n are sets of expressions.

July 09 GDFA: Introduction to Data Flow Analysis 13/37

Data Flow Equations For Available Expressions Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Using Data Flow Information of Available Expressions Analysis

- Used for common subexpression elimination.
  - If an expression is available at the entry of a block b **and**
  - a computation of the expression exists in b **such that**
  - it is not preceded by a definition of any of its operands

  Then the expression is redundant.
- Expression must be upwards exposed or locally anticipable.
- Expressions in Gen_n are downwards exposed.

July 09 GDFA: Introduction to Data Flow Analysis 14/37

Using Data Flow Information of Available Expressions Analysis

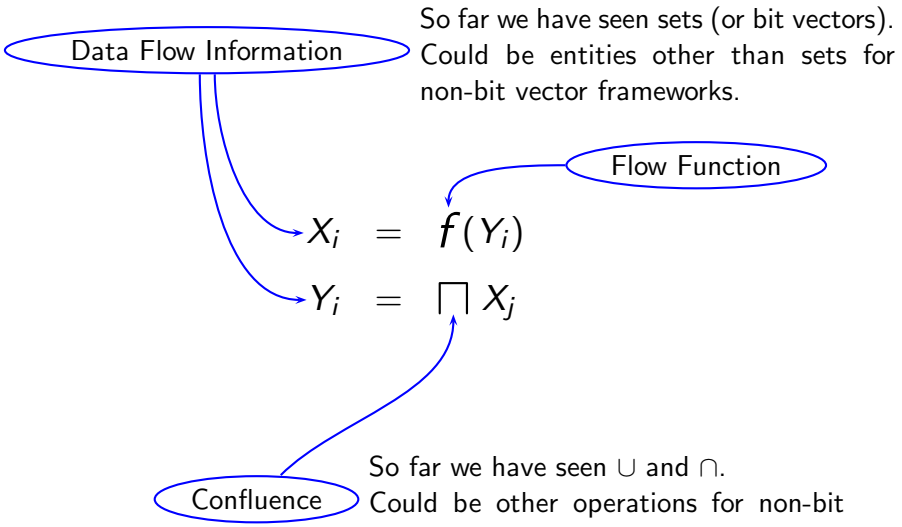
Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Part 3

Common Abstractions in Data Flow Analysis

Common Form of Data Flow Equations



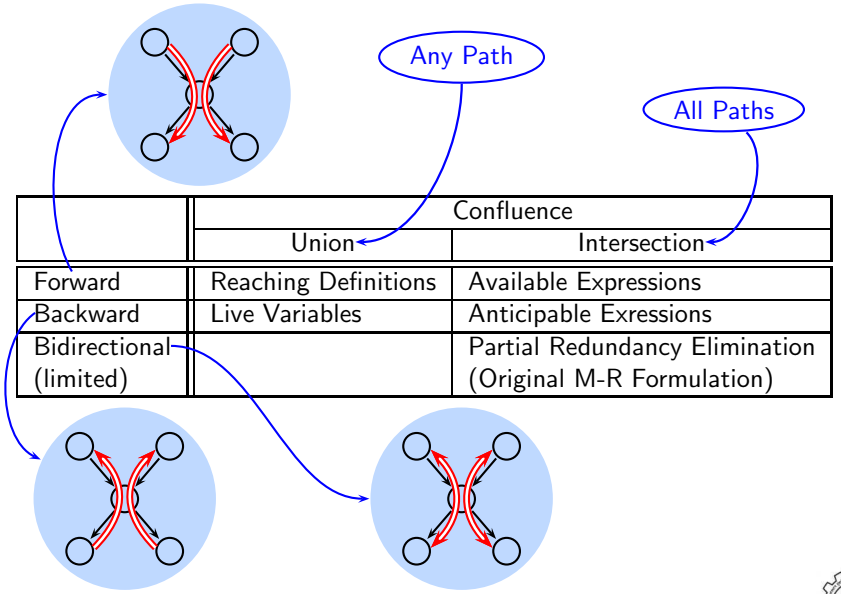
July 09 GDFA: Common Abstractions in Data Flow Analysis 15/37

Common Form of Data Flow Equations

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

A Taxonomy of Bit Vector Data Flow Frameworks

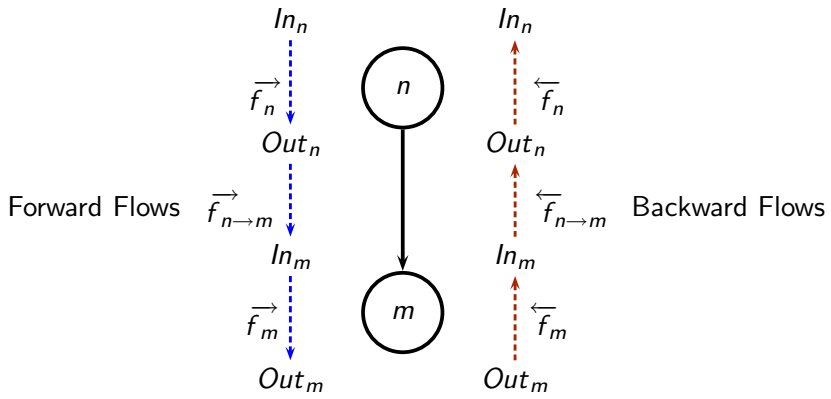


	Confluence	
	Union	Intersection
Forward	Reaching Definitions	Available Expressions
Backward	Live Variables	Anticipable Exressions
Bidirectional (limited)		Partial Redundancy Elimination (Original M-R Formulation)

A Taxonomy of Bit Vector Data Flow Frameworks

Notes

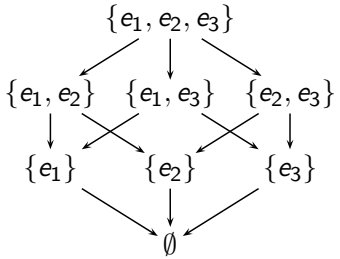
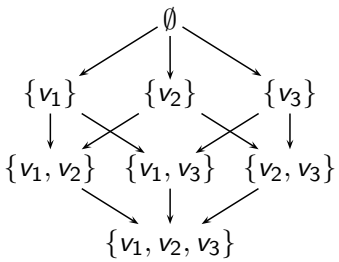
The Abstraction of Flow Functions



The Abstraction of Flow Functions

Notes

The Abstraction of Data Flow Values

Available Expressions Analysis	Live Variables Analysis
	
$\sqsubseteq$ is $\subseteq$	$\sqsubseteq$ is $\supseteq$
$\sqcap$ is $\cap$	$\sqcap$ is $\cup$

July 09 GDFA: Common Abstractions in Data Flow Analysis 18/37

The Abstraction of Data Flow Values

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

July 09 GDFA: Common Abstractions in Data Flow Analysis 19/37

The Abstraction of Data Flow Equations

$$In_n = \begin{cases} BI_{Start} \sqcap \overleftarrow{f}_n(Out_n) & n = Start \\ \left(\prod_{m \in pred(n)} \overrightarrow{f}_{m \to n}(Out_m) \right) \sqcap \overleftarrow{f}_n(Out_n) & \text{otherwise} \end{cases}$$

$$Out_n = \begin{cases} BI_{End} \sqcap \overrightarrow{f}_n(In_n) & n = End \\ \left(\prod_{m \in succ(n)} \overleftarrow{f}_{m \to n}(In_m) \right) \sqcap \overrightarrow{f}_n(In_n) & \text{otherwise} \end{cases}$$

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

July 09 GDFA: Common Abstractions in Data Flow Analysis 19/37

The Abstraction of Data Flow Equations

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Iterative Methods of Performing Data Flow Analysis

Successive recomputation after conservative initialization ($\top$)

- *Round Robin*. Repeated traversals over nodes in a fixed order

  Termination : After values stabilise

  + Simplest to understand and implement
  – May perform unnecessary computations

  Our examples use this method.

- *Work List*. Dynamic list of nodes which need recomputation

  Termination : When the list becomes empty

  + Demand driven. Avoid unnecessary computations.
  – Overheads of maintaining work list.

July 09 GDFA: Common Abstractions in Data Flow Analysis 20/37

Iterative Methods of Performing Data Flow Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Common Form of Flow Functions

$$f_n(X) = (X - \text{Kill}_n(X)) \cup \text{Gen}_n(X)$$

- For General Data Flow Frameworks

$$\begin{aligned} \text{Gen}_n(X) &= \mathit{ConstGen}_n \cup \mathit{DepGen}_n(X) \\ \text{Kill}_n(X) &= \mathit{ConstKill}_n \cup \mathit{DepKill}_n(X) \end{aligned}$$

- For bit vector frameworks

$$\begin{aligned} \text{Gen}_n(X) &= \mathit{ConstGen}_n \\ \text{Kill}_n(X) &= \mathit{ConstKill}_n \end{aligned}$$

July 09 GDFA: Common Abstractions in Data Flow Analysis 21/37

Common Form of Flow Functions

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Defining Flow Functions for Bit Vector Frameworks

- Live variables analysis

	Entity	Manipulation	Exposition
$ConstGen_n$	Variable	Use	Upwards
$ConstKill_n$	Variable	Modification	Anywhere

- Available expressions analysis

	Entity	Manipulation	Exposition
Gen_n	Expression	Use	Downwards
$Kill_n$	Expression	Modification	Anywhere

July 09 GDFA: Common Abstractions in Data Flow Analysis 22/37

Defining Flow Functions for Bit Vector Frameworks

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Part 5

Implementing Data Flow Analysis using gdfa

Implementing Available Expressions Analysis

1. Specifying available expressions analysis
2. Implementing the entry function of available expressions analysis pass
3. Registering the available expressions analysis pass
   - 3.1 Declaring the pass
   - 3.2 Registering the pass
   - 3.3 Positioning the pass

July 09 GDFA: Implementing Data Flow Analysis using gdfa 23/37

Implementing Available Expressions Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Step 1: Specifying Available Expressions Analysis

```
struct gimple_pfbv_dfa_spec gdfa_ave =
{
     entity_expr,                /* entity             */
     ONES,                       /* top_value          */
     ZEROS,                      /* entry_info         */
     ONES,                       /* exit_info          */
     FORWARD,                    /* traversal_order    */
     INTERSECTION,               /* confluence         */
     entity_use,                 /* gen_effect         */
     down_exp,                   /* gen_exposition     */
     entity_mod,                 /* kill_effect        */
     any_where,                  /* kill_exposition    */
     global_only,                /* preserved_dfi      */
     identity_forward_edge_flow, /* forward_edge_flow  */
     stop_flow_along_edge,       /* backward_edge_flow */
     forward_gen_kill_node_flow, /* forward_node_flow  */
     stop_flow_along_node        /* backward_node_flow */
};
```

July 09 GDFA: Implementing Data Flow Analysis using gdfa 24/37

Step 1: Specifying Available Expressions Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Step 2: Implementing Available Expressions Analysis Pass

```
pfbv_dfi ** AV_pfbv_dfi = NULL;

static unsigned int
gimple_pfbv_ave_dfa(void)
{

        AV_pfbv_dfi = gdfa_driver(gdfa_ave);

        return 0;
}
```

July 09 GDFA: Implementing Data Flow Analysis using gdfa 25/37

Step 2: Implementing Available Expressions Analysis Pass

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Step 3.1: Declaring the Available Expressions Analysis Pass

```
struct tree_opt_pass pass_gimple_pfbv_ave_dfa =
{
  "gdfa_ave",           /* name */
  NULL,                 /* gate */
  gimple_pfbv_ave_dfa,  /* execute */
  NULL,                 /* sub */
  NULL,                 /* next */
  0,                    /* static_pass_number */
  0,                    /* tv_id */
  0,                    /* properties_required */
  0,                    /* properties_provided */
  0,                    /* properties_destroyed */
  0,                    /* todo_flags_start */
  0,                    /* todo_flags_finish */
  0                     /* letter */
};
```

July 09 GDFA: Implementing Data Flow Analysis using gdfa 26/37

Step 3.1: Declaring the Available Expressions Analysis Pass

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Step 3.2: Registering the Available Expressions Analysis Pass

In file file `tree-pass.h`

```
extern struct tree_opt_pass pass_gimple_pfbv_ave_dfa;
```

July 09 GDFA: Implementing Data Flow Analysis using gdfa 27/37

Step 3.2: Registering the Available Expressions Analysis Pass

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Step 3.3: Positioning the Pass

In function `init_optimization_passes` in file `passes.c`.

```
  NEXT_PASS (pass_build_cfg);
/* Intraprocedural dfa passes begin */
  NEXT_PASS (pass_init_gimple_pfbvdfa);
  NEXT_PASS (pass_gimple_pfbv_ave_dfa);
```

July 09 GDFA: Implementing Data Flow Analysis using gdfa 28/37

Step 3.3: Positioning the Pass

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Specifying Live Variables Analysis

- Entity should be `entity_var`
- $\top$, BI_{start} and BI_{End} should be `ZEROS`
- Direction should be `BACKWARD`
- Confluence should be `UNION`
- Exposition should be `up_exp`
- Forward edge flow should be `stop_flow_along_edge`
- Forward node flow should be `stop_flow_along_node`
- Backward edge flow should be `identity_backward_edge_flow`
- Backward node flow should be `backward_gen_kill_node_flow`

Specifying Live Variables Analysis

Notes

Part 7

gdfa: Design and Implementation

Specification Data Structure

```
struct gimple_pfbv_dfa_spec
{
   entity_name               entity;
   initial_value             top_value_spec;
   initial_value             entry_info;
   initial_value             exit_info;
   traversal_direction       traversal_order;
   meet_operation            confluence;
   entity_manipulation       gen_effect;
   entity_occurrence         gen_exposition;
   entity_manipulation       kill_effect;
   entity_occurrence         kill_exposition;
   dfi_to_be_preserved       preserved_dfi;
   dfvalue (*forward_edge_flow)(basic_block src, basic_block dest);
   dfvalue (*backward_edge_flow)(basic_block src, basic_block dest);
   dfvalue (*forward_node_flow)(basic_block bb);
   dfvalue (*backward_node_flow)(basic_block bb);
};
```

July 09 GDFA: gdfa: Design and Implementation 30/37

Specification Data Structure

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Specification Primitives

Enumerated Type	Possible Values
entity_name	entity_expr, entity_var, entity_defn
initial_value	ONES, ZEROS
traversal_direction	FORWARD, BACKWARD, BIDIRECTIONAL
meet_operation	UNION, INTERSECTION
entity_manipulation	entity_use, entity_mod
entity_occurrence	up_exp, down_exp, any_where
dfi_to_be_preserved	all, global_only, no_value

July 09 GDFA: gdfa: Design and Implementation 31/37

Specification Primitives

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Pre-Defined Edge Flow Functions

- Edge Flow Functions

Edge Flow Function	Returned value
identity_forward_edge_flow(src, dest)	CURRENT_OUT(src)
identity_backward_edge_flow(src, dest)	CURRENT_IN(dest)
stop_flow_along_edge(src, dest)	top_value

- Node Flow Functions

Node Flow Function	Returned value
identity_forward_node_flow(bb)	CURRENT_IN(bb)
identity_backward_node_flow(bb)	CURRENT_OUT(bb)
stop_flow_along_node(bb)	top_value
forward_gen_kill_node_flow(bb)	CURRENT_GEN(bb) ∪ (CURRENT_IN(bb) - CURRENT_KILL(bb))
backward_gen_kill_node_flow(bb)	CURRENT_GEN(bb) ∪ (CURRENT_OUT(bb) - CURRENT_KILL(bb))

July 09 GDFA: gdfa: Design and Implementation 32/37

Pre-Defined Edge Flow Functions

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

The Generic Driver for Global Data Flow Analysis

```
pfbv_dfi ** gdfa_driver(struct gimple_pfbv_dfa_spec dfa_spec)
{   if (find_entity_size(dfa_spec) == 0) return NULL;
    initialize_special_values(dfa_spec);
    create_dfi_space();
    traversal_order = dfa_spec.traversal_order;
    confluence = dfa_spec.confluence;

    local_dfa(dfa_spec);

    forward_edge_flow = dfa_spec.forward_edge_flow;
    backward_edge_flow = dfa_spec.backward_edge_flow;
    forward_node_flow = dfa_spec.forward_node_flow;
    backward_node_flow = dfa_spec.backward_node_flow;
    perform_pfbvdfa();

    preserve_dfi(dfa_spec.preserved_dfi);
    return current_pfbv_dfi;
}
```

July 09 GDFA: gdfa: Design and Implementation 33/37

The Generic Driver for Global Data Flow Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

The Generic Driver for Local Data Flow Analysis

- The Main Difficulty: Interface with the intermediate representation details
- State of Art: The user is expected to supply the flow function implementation
- Our Key Ideas:
  - Local data flow analysis is a special case of global data flow analysis Other than the start and end blocks (≡ statements), every block has just one predecessor and one successor
  - $ConstGen_n$ and $ConstKill_n$ are just different names given to particular sets of entities accumulated by traversing these basic blocks

The Generic Driver for Local Data Flow Analysis

Notes

The Generic Driver for Local Data Flow Analysis

- Traverse statements in a basic block in appropriate order

Exposition	Direction
`up_exp`	backward
`down_exp`	forward
`any_where`	don't care

- Solve the recurrence

$$\texttt{accumulated\_entities} = (\texttt{accumulated\_entities} - \texttt{remove\_entities}) \cup \texttt{add\_entities}$$

The Generic Driver for Local Data Flow Analysis

Notes

Example for Available Expressions Analysis

Entity is `entity_expr`.

Let expr(x) denote the set of all expressions of x

Exposition	Manipulation	$a = b * c$		$b = b * c$	
		add	remove	add	remove
upwards	use	$b * c$	expr(a)	$b * c$	expr(b)
downwards	use	$b * c$	expr(a)	$\emptyset$	expr(b)
upwards	modification	expr(a)	$b * c$	expr(b) - $\{b * c\}$	$b * c$
downwards	modification	expr(a)	$b * c$	expr(b)	$\emptyset$

Note: In the case of modifications, if we first add then remove the entities modication, the set difference is not required

July 09 GDFA: gdfa: Design and Implementation 36/37

Example for Available Expressions Analysis

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay

Future Work

Main thrust

- Supporting general data flow frameworks
- Supporting interprocedural analysis

July 09 GDFA: Future Work 37/37

Future Work

Notes

Essential Abstrations in GCC GCC Resource Center, IIT Bombay