Module 15: Reaching Definition

Lecture 29: Reaching Definition

The Lecture Contains:

Previous Next

Module 15: Reaching Definition

Lecture 29: Reaching Definition

Reaching Definition

- Definition: Assignment of a value to a variable
- A definition d of a variable reaches a point p if there is a path from d to p and value of a variable is same as assigned at d
- Use iterative forward bit vector problem
- A bit vector of a 8 bit can be used to represent all the definition
- Initial condition
  in(entry) = ∅ = 00000000
  in(i) = ∅ = 00000000 for all nodes i
- If an instruction re-defines a variable then it is killed

Bit position	Definition	Basic block
1	m in node 1	B1
2	f0 in node 2	
3	f1 in node 3	
4	i in node 5	B3
5	f2 in node 8	B6
6	f0 in node 9	
7	f1 in node 10	
8	i in node 11	

a definition not killed is preserved

prsv(B1) = 4, 5, 8 = 00011001
prsv(B3) = 1, 2, 3, 5, 6, 7 = 11101110
prsv(B6) = 1 = 10000000
prsv(i) = 1, 2, 3, 4, 5, 6, 7, 8 = 11111111 for $i \neq$ B1, B3, B6

Gen: Definitions generated in a basic block and not subsequently killed in it
gen(B1) = 1, 2, 3 = 11100000
gen(B3) = 4 = 00010000
gen(B6) = 5, 6, 7, 8 = 00001111
gen(i) = ∅ = 00000000 for i 6= B1, B3, B6

Out: Definitions which reach at the end of a basic block
out(i) = ∅ = 00000000 for all i
A definition reaches end of a basic block iff either

- It is generated in the basic block OR
- It reaches in and is preserved

Previous Next

Module 15: Reaching Definition

Lecture 29: Reaching Definition

Therefore,

out = gen U (in ∩ prsv) for all i

out = gen ∧ (in V prsv) for all i

in(i) = S out(j) for all predecessors of i

in(i) = W out(j) for all predecessors of i

Taxonomy of Dataflow Problems

Categorized along several dimensions

- The information they are designed to provide
- The direction of flow
- Tonfluence operator

Techniques

- Iterative dataflow analysis
- Control tree based method using intervals
- Control tree based method using structural analysis

Four Kinds of Dataflow Problems

Four varieties are distinguished by two orthogonal characteristic

- The operator used for confluence or divergence
- Data flows backward or forward

	∪	∩
Forward	reaching def	available expression
Backward	live variable	busy epxression

Transfer Function

- In forward dataflow out(B) is a transfer function of in(B)
- In backward dataflow in(B) is a transfer function of out(B)

out = (in - kill) [gen

A Lattice L Consists of

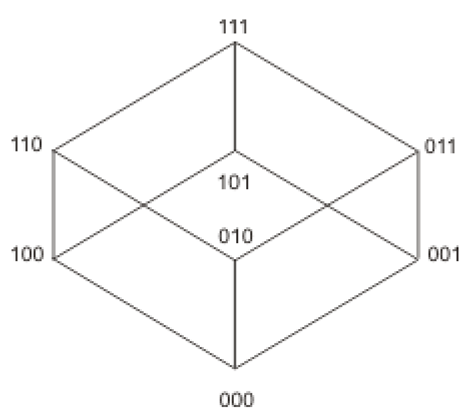
- A set of values
- 'Meet' operator $\sqcap$
- 'Join' operator $\sqcup$

Properties of Lattices

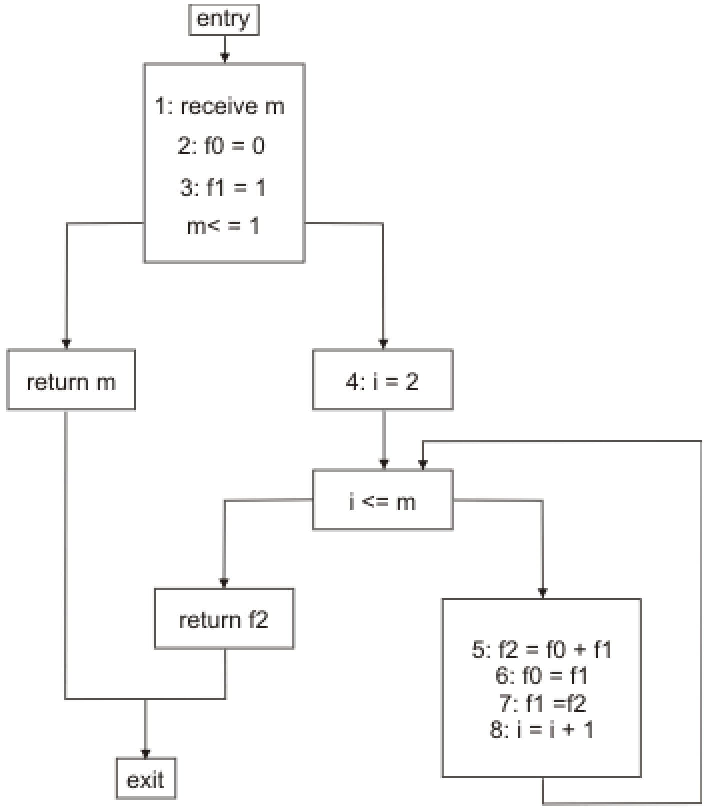
- $\forall x,y \in L \; \exists \; w,z \in L$ such that $x \sqcap y = w$ and $x \sqcup y = z$ (closure)
- $\forall x,y \in L$
  $x \sqcap y = y \sqcap x$ and $x \sqcup y = y \sqcup x$ (Commutative)
- $\forall x,y,z \in L$
  $(x \sqcap y) \sqcap z = x \sqcap (y \sqcap z)$ and
  $(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$ (associate)
- $\forall x,y,z \in L$
  $(x \sqcap y) \sqcup z = (x \sqcap z) \sqcap (y \sqcup z)$ and
  $(x \sqcup y) \sqcap z = (x \sqcap z) \sqcup (y \sqcap z)$ (distributive)
- It has two unique element top $\top$ and bottom $\perp$
  Such that $\forall x \in L$
  $x \sqcap \perp = \perp$ and $x \sqcup \top = \top$

Forward Reaching Definition Analysis

- Elements are bit vectors
- Meet is bitwise and
- Join is bitwise or
- Bottom is 0^n and top is 1^n
- BV^n denotes lattice of n bits for example BV^3 is



Example



- A function f mapping a lattice to itself
  f:L → L is monotonic if for all x, y
  $\sqsubseteq \mathcal{Y} \Rightarrow f(x) \sqsubseteq f(\mathcal{Y})$

  for example f : BV ^{3} → BV ^{3} defined as
  f : (x_1, x_2, x_3) → (x_1, 1, x_3) is monotonic
- Height: it is the length of the longest ascending chain such that there exists
  x_1, x_2, . . . , x_n such that
  $\perp = x_1 \sqsubseteq x_2 \sqsubseteq \cdots \sqsubseteq x_n = \top$
- Monotonicity and finite height ensure that the data-flow algorithms implementing function f terminate
- flow function maps lattice to lattice.
  Flow function for B_1 is given as
  $F_{B1}\left(< x_1, x_2, \ldots, x_8 >\right) =< 111 x_4\, x_5\, 00 x_8 >$
- Let $F_B()$ be the flow function representing flow through block B and F_p represent the composition of the flow functions encountered in following path p then $F_p = F_{Bn}\, o \ldots o\, F_{B1}$

Previous Next

Flow Functions For The Flow–graph in The Example

Fentry $= id$
$F_{B1}(< x_1, x_2, \ldots, x_8 >) =< 111 x_4\, x_5\, 00 x_8 >$
$F_{B2} = id$
$F_{B3}(< x_1, x_2, \ldots, x_8 >) =< x_1\, x_2\, x_3\, 1\, x_5\, x_6\, x_7 0 >$
$F_{B4} = id$
$F_{B5} = id$
$F_{B6}(< x_1, x_2, \ldots, x_8 >) =< x_1\, 000111 >$

Iterative (forward) Data-flow Analysis

Compute in (B) and out (B) $\in$ L for each B $\in$ flow graph

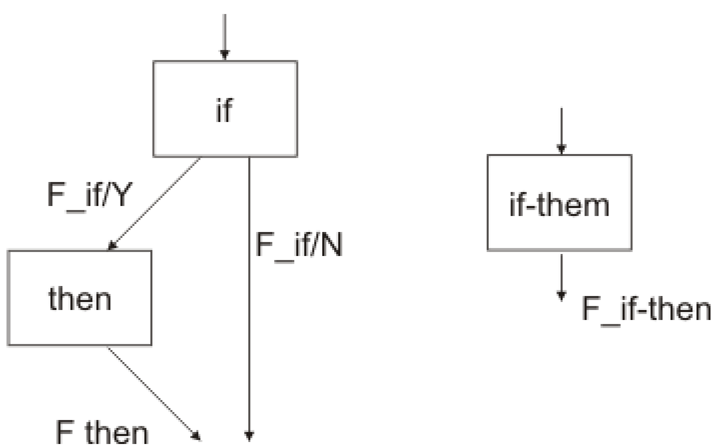
$$in\,(B) = \begin{cases} init & for\ B = entry \\ \underset{P \in pred(B)}{\sqcap} out\,(P) & otherwise \end{cases}$$

Out (B) = F_B (in(B))

Control Tree Based Data-Flow Analysis

- These methods are known as elimination methods
- Applies to both intervals and structures
- Significantly harder to implement - requiring node splitting, iteration, solving of data flow problems over improper regions
- Can be easily adapted to incremental updating of data flow information
- It makes two passes over the control tree
    - First pass (bottom up) constructs flow functions
    - Second pass (top down) constructs and evaluates data-flow equations for each region

If-then Construct



Previous Next

First Pass

$F_{if} - then\left(F_{if/Y}\right) \sqcap F_{if/N}$

If we do not distinguish between $F_{if/Y}$ and $F_{if/N}$ then

$F_{if-then=}(F\ then\ \ o\ \ F_{if}) \sqcap F_{if} = (F\ then \sqcap id) o F_{if}$

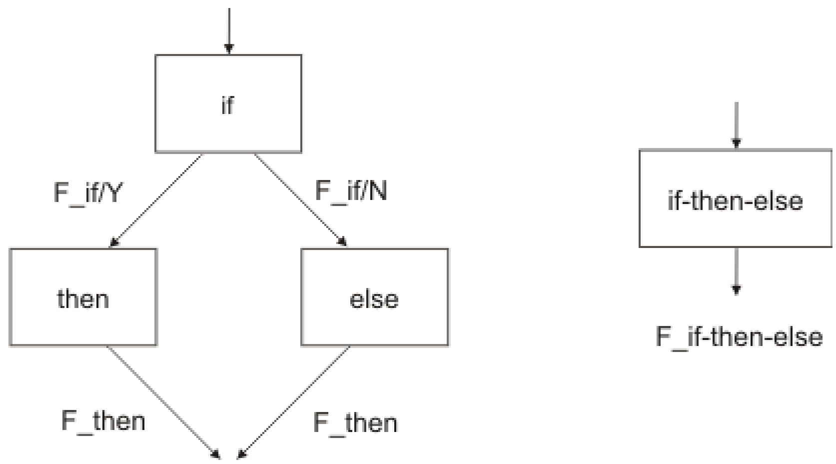
Second Pass

in(if) = in(if-then)
in(then) = $F_{if/Y}$ (in(if))

If we do not distinguish between the true and the false parts

in(if) = in(if-then)
in(then) = F_{if} (in(if))

If-then-else Construct



First Pass

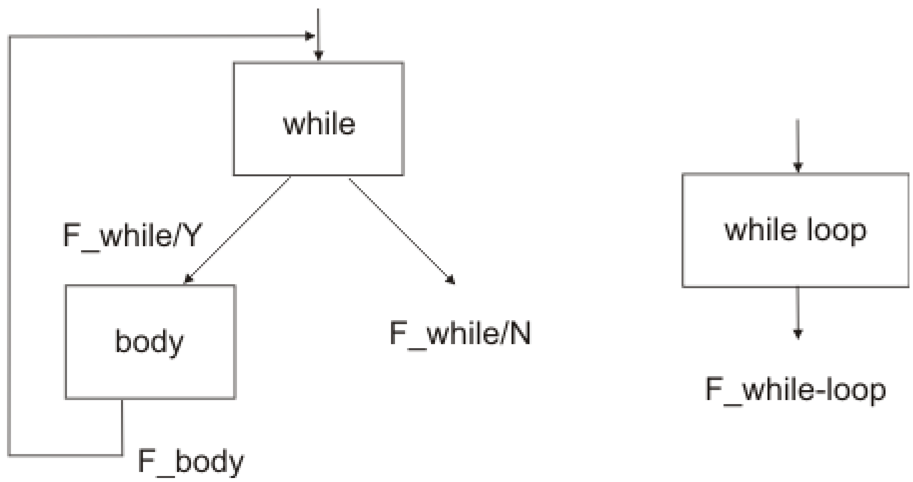
F_{if} -then –else = (F then $oF_{if/Y}$) $\sqcap$ (F else o $F_{if/N}$)

Second Pass

in(if) = in(if-then-else)
in(then) = $F_{if/Y}$ (in(if))
in(else) = $F_{if/N}$ (in(if))

Previous Next

While Loop



First Pass

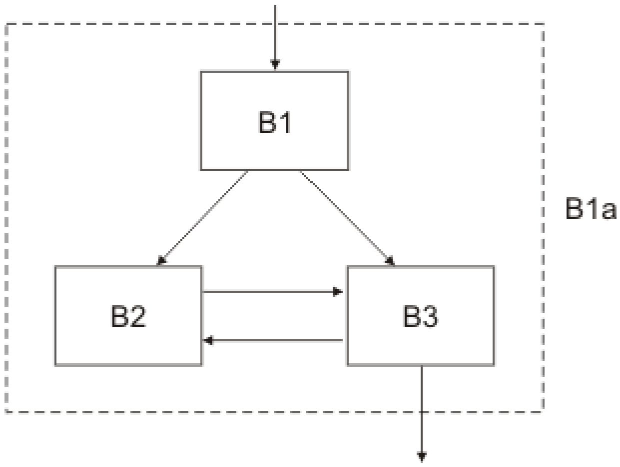
$F_{loop} = \left(F_{body} \circ F_{while/Y}\right)^*$
$F_{while-loop} = F_{while}/N \circ F_{loop}$

Second Pass

in(while) = F_{loop} (in(while-loop))
in(body) = $F_{while/Y}$ (in(while))

If we do not distinguish between $\boldsymbol{F_{while/Y}}$ and $F_{while/N}$ then
$F_{loop} = \left(F_{body} \circ F_{while}\right)^*$
$F_{while-loop} = F_{while} \circ F_{loop}$
$\quad = F_{while} \circ \left(F_{body} \circ F_{while}\right)^*$
In(while) = F_{loop} (in while- loop))
In(body) = F_{while} (in (while))

Improper Region



Bottom up equation:

$((F_{B1a} = F_{B3} \circ F_{B2}) + \circ F_{B1})$

$\sqcap ((F_{B3} \circ F_{B2}) * \circ F_{B3} \circ F_{B1}$

Top down equations:

$\text{in}(B1) = \text{in}(B1a)$

$\text{in}(B2) = F_{B1}(\text{in}(B1)) \sqcap F_{B3}(\text{in}(B3))$

$\text{in}(B3) = F_{B1}(\text{in}(B1)) \sqcap F_{B2}(\text{in}(B2))$

solve for in(B2) and in(B3) in function lattice

$\text{in}(B2) = (((F_{B3} \circ F_{B2}) * \circ F_{B1})$

$\sqcap \left((F_{B3} \circ F_{B2}) * \circ F_{B3} \circ F_{B1})\right)(\text{in}(B1))$

$= \left((F_{B3} \circ F_{B2}) * \circ (\text{id} \sqcap F_{B3}) \circ F_{B1}\right)(\text{in}(B1))$

$\text{in}(B3) = (((F_{B2} \circ F_{B3}) * \circ F_{B1})$

$\sqcap ((F_{B2} \circ F_{B3}) * \circ F_{B2} \circ F_{B1}))(in(B1))$

$\left((F_{B3} \circ F_{B2}) * (id \sqcap F_{B2}) \circ F_{B2}\right) \circ F_{B1})(in(B1))$

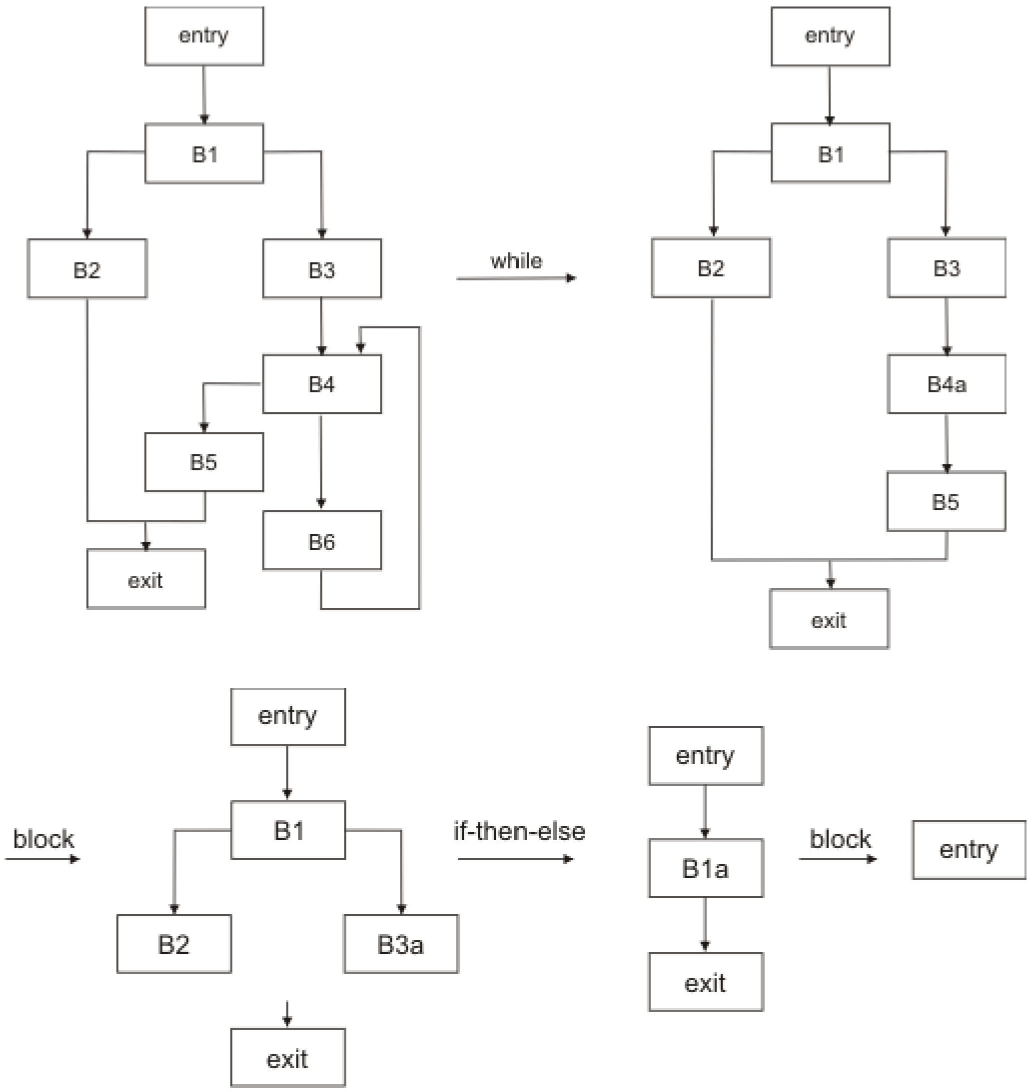
Previous Next

Module 15: Reaching Definition

Lecture 29: Reaching Definition

Example

Consider the flow graph in an earlier slide and its structural flow analysis



First Pass

$$F_{B_{4a}} = F_{B_4} \circ (F_{B_6} \circ F_{B_4})*$$

$$F_{B_{3a}} = F_{B_5} \circ F_{B_{4a}} \circ F_{B_3}$$

$$F_{B_{1a}} = (F_{B_2} \circ F_{B_1}) \sqcap (F_{B_{3a}} \circ F_{B_1})$$

$$F_{entrya} = F_{exit} \circ F_{B_{1a}} \circ F_{entry}$$

Second Pass

For entrya block

$$in\ (entry) = init$$

$in(B_{1a}) = F_{entry}(in(entry))$

$in(exit) = F_{B_{1a}}(in(B_{1a}))$

For if-then else block

$in(B_1) = in(B_{1a})$

$in(B_2) = F_{B_1}(in(B_1))$

$in(B_3) = F_{B_1}(in(B_1))$