Interprocedural Data Flow Analysis

Aditya V. Nori
MSR India

Intraprocedural Analysis

procedureless program $\rightsquigarrow$ flow graph

Assumption: All paths in the graph represent actual executions of the program.

Not true!

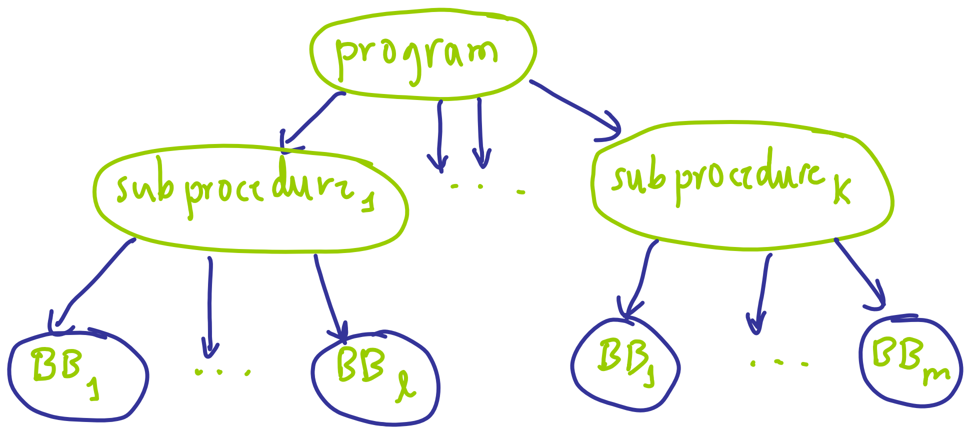
But this model is popular:

a) Relatively simple structure enables the development of simple algorithms for program analysis.

b) Isolation of feasible paths from infeasible paths is an undecidable problem.

Reference: Flow Analysis of Computer Programs – Hecht 1977.

Notation



A **basic block** is a maximal single entry multi exit sequence of code.

Each procedure p is assumed to have a single entry block r_p and a single exit block e_p.

The **flow graph** G_p of p is a rooted directed graph whose nodes are the basic blocks of p, with root r_p. There is an edge (m,n) in G_p if there is a direct transfer of control from basic block m to basic block n.

Notation

Let $G = (N, E, r)$ be a directed graph.

nodes edges root

A path p in G is a sequence of nodes in N $(n_1, n_2, \ldots, n_k)$ s.t. for each $1 \leq j < k$, $(n_j, n_{j+1}) \in E$.

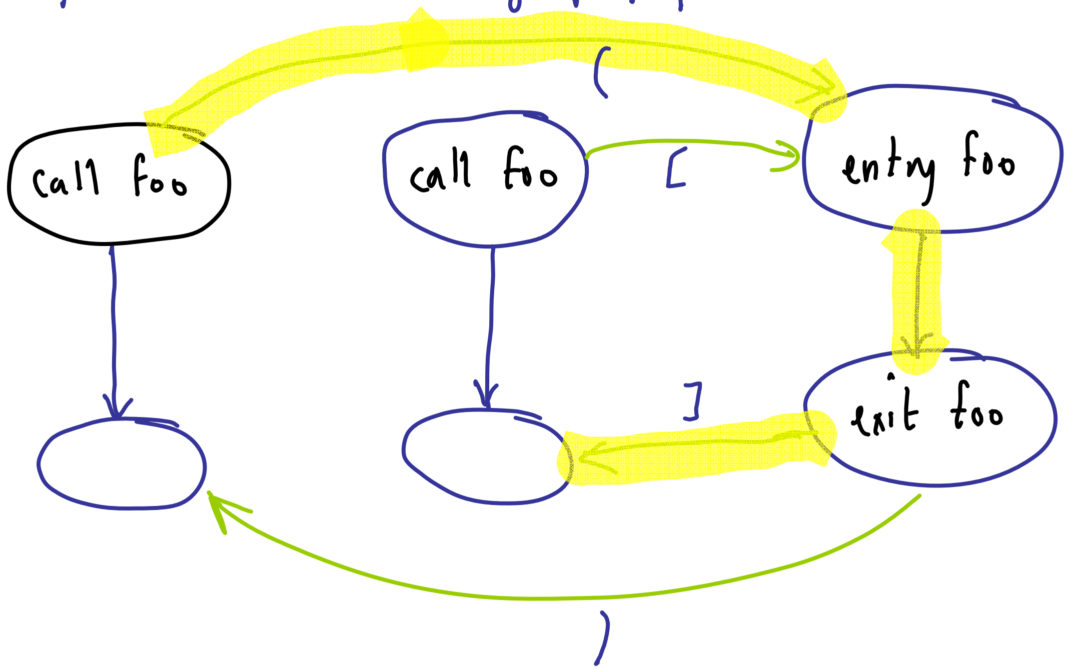
The length of p is the no. of edges along p, denoted $\mathrm{length}(p)$.

$\mathrm{path}_G(m, n)$ = set of all paths in G leading from m to n.

Interprocedural Analysis

(A) **Functional Approach:** Views procedures as collections of structured program blocks and aims to establish I/O relations for each block.

(B) **Call string Approach:** Information is tagged with an encoded history of procedure calls during propagation.



A global data flow frame is a pair (L, F), where L is a semilattice of data and $F : L \to L$ is a space of functions. F is closed under composition and meet and is monotone.

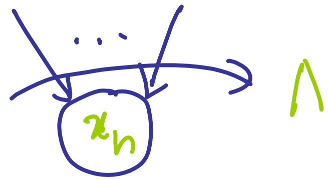
(L, F) is a distributive framework if:

$$\forall f \in F \text{ and } x, y \in L, \quad f(x \wedge y) = f(x) \wedge f(y)$$

<u>Data propagation equations</u> : (Given (L, F) and G)

$$x_r = 0$$

$$x_n = \bigwedge_{(m,n) \in E} f_{(m,n)}(x_m) \qquad n \in N - \{r\}$$



Note: every edge (m, n) of G is associated with a propagation fn $f_{(m,n)}$ which represents change of data as control transfers from m to n.

Available Expression Analysis

Aim: For each program point, determine which expressions have already been computed, and not later modified, on all paths to a program point.

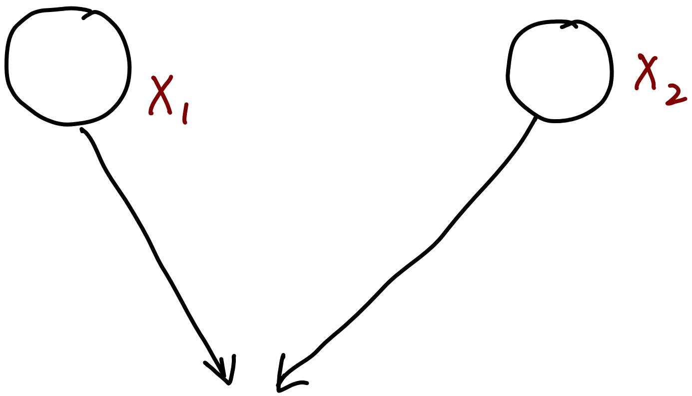
Example:

```
x := a + b ;
y := a * b ;
while y > a + b
do
    a := a + 1 ;
    x := a + b ;
od
```

$\longrightarrow$

```
x := a + b ;
y := a * b ;
while y > x
do
    a := a + 1 ;
    x := a + b ;
od
```

Basic Idea



$M = X_1 \cap X_2$

$\boxed{x := a}$

$$X = M \setminus \overbrace{\{\text{expressions with an } x\}}^{\text{kill}} \cup \overbrace{\{\text{subexpressions of } a \text{ with an } x\}}^{\text{gen}}$$

$L = (\mathcal{P}(Exps), \supseteq, \cap, Exps) \longrightarrow$ semilattice

The least upper bound operator is $\cap$.

The least element $\perp$ is $Exps$.

L satisfies the ascending chain condition.

F is the set of transfer functions (gen & kill).

<u>Exercise:</u> Show that this is a distributive dfa framework.

<u>Examples:</u> [ASU].

Meet Over all Paths (MOP):

$$y_n = \bigwedge \{ f_p(0) : \text{path}_G(r,n) \}$$

For each path $p = (n_1, n_2, \ldots, n_k)$, $f_p = f_{(n_{k-1}, n_k)} \circ f_{(n_{k-2}, n_{k-1})} \circ \ldots \circ f_{(n_1, n_2)}$

If p is empty, $f_p = id_L$.

Computing y_n is undecidable in general.

[Kildall]: If (L,F) is distributive, then $x_n = y_n$, $\forall n \in N$.

[Kam]: If (L,F) is monotone, then $x_n \leq y_n$, $\forall n \in N$.

Interprocedural Flow Graphs

$G = \bigcup \{ G_p : p \text{ is a procedure in the program} \}$, where

$G_p = (N_p, E_p, r_p)$ s.t.

- N_p : set of all basic blocks in p.
- $E_p = E_p^0 \cup E_p^1$: set of edges of G_p.
  - $(m,n) \in E_p^0 \rightarrow$ there can be a direct flow of control from m to n.
  - $(m,n) \in E_p^1 \rightarrow$ m is a call block and n follows that block.

$G^* = (N^*, E^*, r_1)$, where $N^* = \bigcup_p N_p$, and $E^* = E^0 \cup E^1$;

$E^0 = \bigcup_p E_p^0$ ← defined earlier.

$(m,n) \in E^1$ → m is a call block and n is the called proc. entry (call edge)

→ m is an exit block of some proc p and n follows the call to p (return edge).

The call edge (m, r_p) and a return edge (e_q, n) are said to correspond to each other if $p = q$ and $(m,n) \in E_s^1$ for some procedure s.

Note: All paths in G^* are not feasible!

$IVP(r_1, n)$: set of all interprocedurally valid paths in G^*, from r_1 to n.

$q \in path_{G^*}(r_1, n) \in IVP(r_1, n)$ if $q|_{E'}$ is proper.

(a) $q|_{E'}$ is proper if it has no return edges.

(b) If $q|_{E'}$ contains return edges $\rightarrow$ they are matched by corresponding calls.

$$\boxed{\begin{array}{l} \bigcup_{n} IVP(r_1, n) \text{ is a CFL.} \\ path_{G^*}(r_1, n) \text{ is regular} \end{array}}$$

A path $q \in IVP_0(r_p, n)$ iff $q_1 = q|_{E^1}$ is complete

- The null tuple is complete.
- q_1 is complete if
  (a) it is the concatenation of two complete tuples, or
  (b) it starts with a call edge and terminates with a corresponding return edge, and the rest of the tuple is complete.

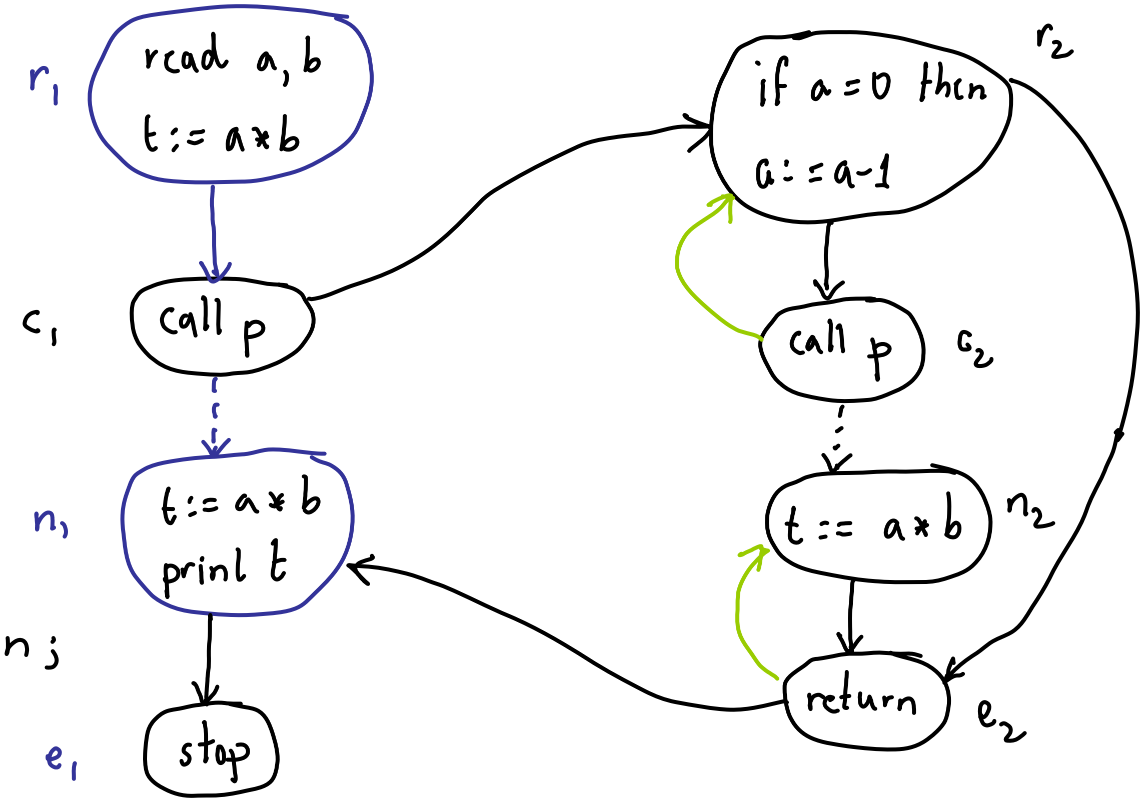
Example

main program

```
read a, b;
t := a*b;
call p;
t := a*b;
print t;
stop;
```

procedure p

```
if a=0 then return;
else
    a := a-1;
    call p;
    t := a*b;
endif;
return;
```



$$q_1 = (r_1, c_1, r_2, c_2, r_2, e_2, n_2, e_2, n_1, e_1) \in IVP(r_1, e_1)$$

$$q_2 = (r_1, c_1, r_2, c_2, r_2, e_2, n_1, e_1) \notin IVP(r_1, e_1)$$

$$q_3 = (r_2, c_2, r_2, e_2, n_2) \in IVP_0(r_2, n_2)$$

$$q_4 = (r_2, c_2, r_2, c_2, e_2, n_2) \notin IVP_0(r_2, n_2)$$

Lemma: Let $n \in N^*$ and $q \in IVP(r_1, n)$. Then there exist procedures $p_1, p_2, \cdots, p_j$, where p_1 is the main program and p_j the procedure containing n, and calls $c_1, \ldots, c_{j-1}$ such that for each $i<j$, c_i is in p_i and calls p_{i+1}, and q can be represented as

$$q = q_1 \,||\, (c_1, r_{p_2}) \,||\, q_2 \,||\, (c_2, r_{p_3}) \,||\, \ldots \,||\, (c_{j-1}, r_{p_j}) \,||\, q_j$$

where for each $i<j$, $q_i \in IVP_0(r_{p_i}, c_i)$ and $q_j \in IVP_0(r_{p_j}, n)$. Conversely, any path which admits such a decomposition is in $IVP(r_1, n)$ and this decomposition is unique.

Proof: Simple induction on path length.

(Reading exercise!)

A Functional Approach to Interprocedural Analysis

Let (L, F) be a distributive flow data flow framework for G.

Phase I :

$$\phi_{(r_p, r_p)} \leq id_L \quad \text{for each procedure } p.$$

$$\phi_{(r_p, n)} = \bigwedge_{(m,n) \in E_p} \left(h_{(m,n)} \circ \phi_{(r_p, m)} \right) \quad \forall n \in N_p .$$

where

$$h_{(m,n)} = \begin{cases} f_{(m,n)} & \text{if } (m,n) \in E_p^0 \\ \phi_{(r_q, e_q)} & \text{if } (m,n) \in E_p^1 \text{ and } m \text{ calls procedure } q. \end{cases}$$

Define an ordering over F as follows:

$$\forall g_1, g_2 \in F, \quad g_1 \geq g_2 \iff g_1(x) \geq g_2(x) \quad \forall x \in L.$$

$$\left.\begin{aligned} &\text{Let } \phi^0_{(r_p, r_p)} = Id_L \text{ for each procedure } p. \\ &\phi^0_{(r_p, n)} = f_\Omega \text{ for each } n \in N_p - \{r_p\}. \end{aligned}\right\} \text{initialization}$$

Compute the maximum fixed point for this set of equations.

<u>Phase II</u> For each basic block n, let $x_n \in L$ denote the info available at the start of n.

$$x_{r_{main}} = 0 \in L$$

$$x_{r_p} = \bigwedge \{ \phi_{(r_q, c)}(x_{r_q}) : q \text{ is a procedure and } c \text{ is a call to } p \text{ in } q \}$$

for each procedure p.

$$x_n = \phi_{(r_p, n)}(x_{r_p}) \text{ for each procedure } p, \text{ and } n \in N_p - \{r_p\}.$$

Example - Available expressions

$L = \{0, 1, \Omega\}$ $\qquad\qquad$ $F = \{\underline{0}, \underline{1}, id_L, f_\Omega\}$

a*b is available

Phase I:

$$\phi_{(r_1,r_1)} = \phi_{(r_2,r_2)} = id_L$$

$$\phi_{(r_1,c_1)} = \underline{1} \circ \phi_{(r_1,r_1)}$$

$$\phi_{(r_1,n_1)} = \phi_{(r_2,e_2)} \circ \phi_{(r_1,c_1)}$$

$$\phi_{(r_1,e_1)} = \underline{1} \circ \phi_{(r_1,n_1)}$$

$$\phi_{(r_2,c_2)} = \underline{0} \circ \phi_{(r_2,r_2)}$$

$$\phi_{(r_2,n_2)} = \phi_{(r_2,e_2)} \circ \phi_{(r_2,c_2)}$$

$$\phi_{(r_2,e_2)} = [id_L \circ \phi_{(r_2,r_2)}] \wedge [\underline{1} \circ \phi_{(r_2,n_2)}]$$

Function	Initial value	It 1	It 2	It 3
$\phi_{(r_1,r_1)}$	id_L	id_L	id	id
$\phi_{(r_1,c_1)}$	f_Ω	$\underline{1}$	$\underline{1}$	$\underline{1}$
$\phi_{(r_1,n_1)}$	f_Ω	f_Ω	$\underline{1}$	$\underline{1}$
$\phi_{(r_1,e_1)}$	f_Ω	f_Ω	$\underline{1}$	$\underline{1}$
$\phi_{(r_2,r_2)}$	id_L	id_L	id_L	id_L
$\phi_{(r_2,c_2)}$	f_Ω	$\underline{0}$	0	0
$\phi_{(r_2,n_2)}$	f_Ω	f_Ω	0	0
$\phi_{(r_2,e_2)}$	f_Ω	id_L	id_L	id_L

fixed point !

Phase II :

$$x_{r_1} = 0$$

$$x_{r_2} = \phi_{(r_1, e_1)}(x_{r_1}) \wedge \phi_{(r_2, c_2)}(x_{r_2})$$

$$= \underline{1}\,(x_{r_1}) \wedge \underline{0}\,(x_{r_2})$$

After two iterations: $x_{r_1} = x_{r_2} = 0$

Complete solution: $x_{r_1} = x_{r_2} = x_{c_2} = x_{n_2} = x_{e_2} = 0$

$$x_{c_1} = x_{n_1} = x_{e_1} = 1$$

$\Rightarrow$ a*b is available at the start of n_1.

Interprocedural MOP

$\psi_n = \bigwedge \{ f_q : q \in IVP(r_{main}, n) \} \in F$ for each $n \in N^*$.

$y_n = \psi_n(0)$ for each $n \in N^*$.

this is the MOP solution.

Lemma: Let $n \in N_p$ for some procedure p. Then

$$\phi_{(r_p, n)} = \bigwedge \{ f_q : q \in IVP_0(r_p, n) \}$$

Proof:

Step 1: Prove by induction on $i \rightarrow \phi^i_{(r_p,n)} \geq \bigwedge \{ f_q : q \in IVP_0(r_p, n) \}$.

Basis: $i = 0 \rightarrow$ (a) $n = r_p \rightarrow \phi^0_{(r_p, r_p)} = id_L \geq \bigwedge \{ f_q : q \in IVP_0(r_p, r_p) \} = id_L$

$\rightarrow$ (b) $n \neq r_p \rightarrow \phi^0_{(r_p, n)} = f_\Omega \geq f, \forall f \in F$

Inductive hypothesis: Assume $\phi^{i}_{(r_p,n)} \geq \bigwedge \{ f_q : q \in IVP_0(r_p,n) \}$ for some i.

$$\phi^{i+1}_{(r_p,n)} = \bigwedge_{(m,n)\in E_p} \left(h_{(m,n)} \circ \phi^{i}_{(r_p,m)} \right)$$

$$\geq \bigwedge_{(m,n)\in E_p} \left(h_{(m,n)} \circ \bigwedge \{ f_q : q \in IVP_0(r_p,m) \} \right)$$ for each procedure p and $n \in N - \{r_p\}$

$$= \bigwedge_{(m,n)\in E_p^0} \left(f_{(m,n)} \circ \bigwedge \{ f_q : q \in IVP_0(r_p,m) \right)$$

$$\bigwedge_{\substack{(m,n)\in E_p' \\ m \text{ calls } p'}} \left(\phi_{(r_{p'},e_{p'})} \circ \bigwedge \{ f_q : q \in IVP_0(r_p,m) \} \right.$$

$$\geq \bigwedge_{(m,n)\in E_p^0} \left(\bigwedge \{ f_{q\|(m,n)} : q \in IVP_0(r_p,m) \} \right) \wedge$$

$$\bigwedge_{\substack{(m,n)\in E_p' \\ m \text{ calls } p}} \left(\bigwedge \{ f_{q'} : q' \in IVP_0(r_{p'},e_{p'}) \} \circ \bigwedge \{ f_q : q \in IVP_0(r_p,m) \} \right)$$

$$= \bigwedge_{(m,n)\in E_p^0} \left(\bigwedge \{ f_{q \| (m,n)} : q \in IVP_0(r_p, m) \} \right) \wedge$$

$$\bigwedge_{\substack{(m,n)\in E_p' \\ m \text{ calls } p'}} \left(\bigwedge \{ f_{q \| (m, r_{p'}) \| q' \| (e_{p'}, n)} : q \in IVP_0(r_p, m),\ q' \in IVP_0(r_{p'}, e_{p'}) \} \right)$$

$$\geq \{ f_q : q \in IVP_0(r_p, n) \}.$$

<u>Step 2:</u> For each $q \in IVP_0(r_p, n)$, show that $f_q \geq \phi_{(r_p, n)}$.

Proof by induction on the length of q.

<u>Basis:</u> If $\text{length}(q) = 0$, then $n = r_p \rightarrow f_q = \phi_{(r_p, r_p)} = id_L$.

<u>Inductive hypothesis:</u> Assume that $f_q \geq \phi_{(p,n)}$, $\forall p, n$ and $\forall q \in IVP_0(r_p, n)$ s.t. $\text{length}(q) \leq k$.

<u>Induction:</u> Let p, q, n be such that $\text{length}(q) = k+1$. Let (m,n) be the last edge in $q \rightarrow q = q_1 \| (m,n)$.

If $(m,n) \in E_p^0 \Rightarrow q_1 \in IVP_0(r_p, m)$ and $\text{length}(q_1) \leqslant k$.

$\therefore f_{q_1} \geqslant \phi_{(r_p,m)}$ and $f_q = f_{(m,n)} \circ f_{q_1} \geqslant h_{(m,n)} \circ \phi_{(r_p,m)} \geqslant \phi_{(r_p,n)}$

If $(m,n) \in E^1$, then $m = e_{p'}$ for some procedure p'.

$\therefore q = q_1 \,\|\, (m_1, r_{p'}) \,\|\, q_2 \,\|\, (e_{p'}, n)$

$\uparrow$ $IVP_0(r_p, m_1)$ $\quad\quad$ $\uparrow$ $IVP_0(r_{p'}, e_{p'})$ $\quad\quad$ & $(m_1, n) \in E_p^1$

Since $f_{(m_1, r_{p'})} = f_{(e_{p'}, n)} = id_L \Rightarrow f_q = f_{q_2} \circ f_{q_1}$

$$\Rightarrow f_q \geqslant \underbrace{\phi_{(r_{p'}, e_{p'})} \circ \phi_{(r_p, m_1)}}_{\text{by IH.}} = h_{(m_1,n)} \circ \phi_{(r_p, m_1)} \geqslant \phi_{(r_p, n)}$$

For each basic block n, define

$$\chi_n = \bigwedge \{ \phi_{(r_{p_j}, n)} \circ \phi_{(r_{p_{j-1}}, c_j)} \circ \ldots \circ \phi_{(r_{p_1}, c_1)} : p_1 = \text{main program},$$

p_j is the procedure containing n, and for each $i<j$, c_j is a call to p_{i+1} from p_i }

$$z_n = \chi_n(0)$$

<u>Thm:</u> $\varphi_n = \chi_n$ for each $n \in N^*$.

<u>Proof:</u> Let $q \in IVP(r_{main}, n)$. Then $q = q_1 \| (c_1, r_{p_2}) \| q_2 \| \ldots \| (c_{j-1}, r_{p_j}) \| q_j$ (path decomposition thm).

p_1 = main program

$p_2, \ldots, p_{j-1}$ = procedures

p_j = procedure containing n

$c_1, \ldots, c_{j-1}$ = calls s.t. $\forall i<j$, c_i is a call to p_{i+1} from p_i,

and $q_i \in IVP_0(r_{p_i}, c_i)$ and $q_j \in IVP_0(r_{p_j}, n)$

By the earlier lemma:

$$f_q = f_{q_j} \circ f_{q_{j-1}} \circ \dots \circ f_{q_1} \geq \phi_{(r_{p_j}, n)} \circ \phi_{(r_{p_{j-1}}, c_{j-1})} \circ \dots \circ \phi_{(r_{p_1}, c_1)} \geq \chi_n$$

Therefore $\psi_n \geq \chi_n$.

Conversely, let $p_1, \dots, p_j, c_1, \dots, c_{j-1}$ be defined as before.

$$\phi_{(r_{p_j}, n)} \circ \phi_{(r_{p_{j-1}}, c_{j-1})} \circ \dots \circ \phi_{(r_{p_1}, c_1)} = \bigwedge \{ f_{q_j} \circ f_{q_{j-1}} \circ \dots \circ f_{q_1} : q_i \in IVP_0(r_{p_i}, c_i) \text{ for each } i<j \text{ and } q_j \in IVP_0(r_{p_j}, n) \}$$

$$= \{ f_{q_1 \| (c_1, r_{p_1}) \| \dots \| (c_{j-1}, r_{p_j}) \| q_j} : \dots \} = \mathbb{P}$$

Every path in $\mathbb{P}$ belongs to $IVP(r_{main}, n)$

$$\Rightarrow \mathbb{P} \geq \bigwedge \{ f_q : q \in IVP(r_{main}, n) \}$$

Therefore $\chi_n \geq \psi_n \Rightarrow \chi_n = \psi_n \quad \forall n \in \mathbb{N}^*$.

Main Result

Thm: For each basic block $n \in N^*$, $x_n = y_n = z_n$.

Proof: $\underbrace{\because \psi_n = \chi_n \quad \forall n \in N^*}_{\text{previous thm}} \Rightarrow y_n = z_n$. (Note: $y_n = \psi_n(0)$ and $z_n = \chi_n(0)$)

Claim: $x_{r_p} = z_{r_p}$, $\forall p \in$ Procedures.

If this claim is true, then by

$$\underbrace{\begin{array}{ll}
\text{(a)} & x_n = \phi_{(r_p, n)}(x_{r_p}) \text{ for each procedure } p \text{ and } n \in N - \{r_p\}. \\
\text{(b)} & \chi_n = \wedge \left\{ \phi_{(r_{p_j}, n)} \circ \phi_{(r_{p_{j-1}}, c_{j-1})} \circ \ldots \circ \phi_{(r_{p_1}, c_1)} : \ldots \right\} \\
\text{(c)} & z_n = \chi_n(0)
\end{array}}$$

$$\Downarrow$$

$x_n = z_n$ for each $n \in N^*$

Proof of the claim

Define a new flow graph $G_c = (N_c, E_c, r_c)$, where N_c is the set of all call and entry blocks in the program.

$E_c = E_c^0 \cup E_c^1$.

$(m,n) \in E_c^0 \iff m$ is the entry node of some procedure p and n is a call within p.

$(m,n) \in E_c^1 \iff m$ is a call to some procedure p and n is the entry of p.

Define a DFP for G_c by associating $g_{(m,n)} \in F$, for each $(m,n) \in E_c$.

$$g_{(m,n)} = \begin{cases} \phi_{(m,n)} & \text{if } (m,n) \in E_c^0 \\ id_L & \text{if } (m,n) \in E_c^1 \end{cases}$$

Iterative eqns for this DFP:

$$x_{r_{main}} = 0 \in L$$

$$x_{r_p} = \bigwedge \{ \phi_{(r_q, c)} (x_{r_c}) : q \text{ is a procedure and } c \text{ is a call to } p \text{ in } q \}$$

for each procedure p.

The MOP solution for the new problem:

$$x_n = \bigwedge \{ \phi_{(r_{p_j}, n)} \circ \phi_{(r_{p_{j-1}}, c_{j-1})} \circ \dots \circ \phi_{(r_{p_1}, c_1)} :$$

$$p_1 = \text{main program},\ p_2, \dots, p_j \text{ are procedures} \},\ n \in \{\text{call}, \text{entry}\}$$

$$z_n = x_n(0).$$

Since F is distributive, from Kildall's thm $x_{r_p} = z_{r_p}$ for each procedure p.