

CS738: Advanced Compiler Optimizations

Interprocedural Data Flow Analysis Functional Approach

Amey Karkare

karkare@cse.iitk.ac.in

<http://www.cse.iitk.ac.in/~karkare/cs738>
Department of CSE, IIT Kanpur



Interprocedurally Valid Paths

- ▶ G^* ignores the special nature of call and return edges
- ▶ Not all paths in G^* are feasible
 - ▶ do not represent potentially valid execution paths
- ▶ $IVP(r_1, n)$: set of all interprocedurally valid paths from r_1 to n
- ▶ Path $q \in \text{path}_{G^*}(r_1, n)$ is in $IVP(r_1, n)$
 - ▶ iff sequence of all E^1 edges in q (denoted q_1) is *proper*

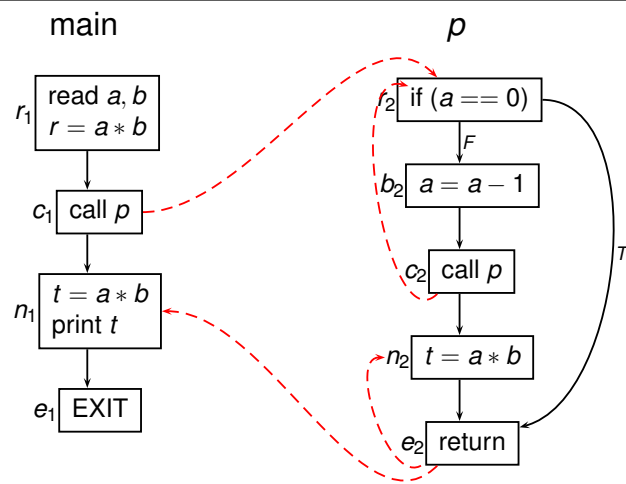
Proper sequence

- ▶ q_1 without any return edge is proper
- ▶ let $q_1[i]$ be the first return edge in q_1 . q_1 is proper if
 - ▶ $i > 1$; and
 - ▶ $q_1[i-1]$ is call edge corresponding to $q_1[i]$; and
 - ▶ q_1' obtained from deleting $q_1[i-1]$ and $q_1[i]$ from q_1 is proper

Interprocedurally Valid Complete Paths

- ▶ $IVP_0(r_p, n)$ for procedure p and node $n \in N_p$
- ▶ set of all interprocedurally valid paths q in G^* from r_p to n s.t.
 - ▶ Each call edge has corresponding return edge in q restricted to E^1

IVPs



$r_1 \rightarrow c_1 \rightarrow r_2 \rightarrow c_2 \rightarrow r_2 \rightarrow e_2 \rightarrow n_2 \rightarrow e_2 \rightarrow n_1 \rightarrow e_1$
 $\in \text{IVP}(r_1, e_1)$

$r_1 \rightarrow c_1 \rightarrow r_2 \rightarrow c_2 \rightarrow r_2 \rightarrow e_2 \rightarrow n_1 \rightarrow e_1 \notin \text{IVP}(r_1, e_1)$

$r_2 \rightarrow c_2 \rightarrow r_2 \rightarrow e_2 \rightarrow n_2 \in \text{IVP}_0(r_2, n_2)$

$r_2 \rightarrow c_2 \rightarrow r_2 \rightarrow c_2 \rightarrow e_2 \rightarrow n_2 \notin \text{IVP}_0(r_2, n_2)$

Path Decomposition

$q \in \text{IVP}(r_{\text{main}}, n)$

\Leftrightarrow

$q = q_1 \parallel (c_1, r_{p_2}) \parallel q_2 \parallel \dots \parallel (c_{j-1}, r_{p_j}) \parallel q_j$
 where for each $i < j$, $q_i \in \text{IVP}_0(r_{p_i}, c_i)$ and $q_j \in \text{IVP}_0(r_{p_j}, n)$

Functional Approach

- ▶ (L, F) : a *distributive* data flow framework
- ▶ Procedure p , node $n \in N_p$
- ▶ $\phi_{(r_p, n)} \in F$ describes flow of data flow information from start of r_p to start of n
 - ▶ along paths in $\text{IVP}_0(r_p, n)$

Functional Approach Constraints

$\phi_{(r_p, r_p)} \leq id_L$

$\phi_{(r_p, n)} = \bigwedge_{(m, n) \in E_p} (h_{(m, n)} \circ \phi_{(r_p, m)})$ for $n \in N_p$

where

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

Information x at r_p translated to information $\phi_{(r_p, n)}(x)$ at n

Solving ϕ Constraints

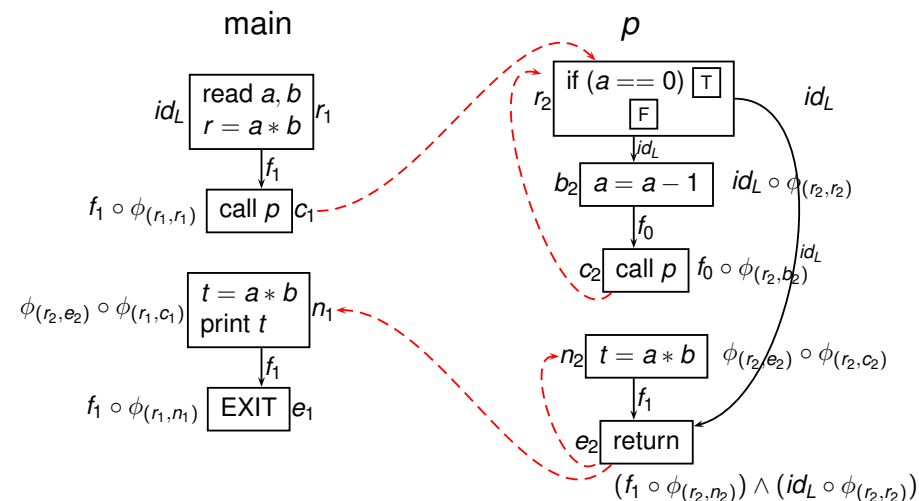
- ▶ Round-robin iterative approximations to initial values

$$\phi_{(r_p, r_p)}^0 \leq id_L$$

$$\phi_{(r_p, n)}^0 \leq f_\Omega \quad n \in N_p - \{r_p\}$$

- ▶ Reach maximal fixed point

Example



Iterative Solution

Function	Constraint	Iteration #			
		Init	1 st	2 nd	3 rd
$\phi_{(r_1, r_1)}$	id_L	id	id	id	id
$\phi_{(r_1, c_1)}$	$f_1 \circ \phi_{(r_1, r_1)}$	f_Ω	f_1	f_1	f_1
$\phi_{(r_1, n_1)}$	$\phi_{(r_2, e_2)} \circ \phi_{(r_1, c_1)}$	f_Ω	f_Ω	f_1	f_1
$\phi_{(r_1, e_1)}$	$f_1 \circ \phi_{(r_1, n_1)}$	f_Ω	f_1	f_1	f_1
$\phi_{(r_2, r_2)}$	id_L	id	id	id	id
$\phi_{(r_2, b_2)}$	$id_L \circ \phi_{(r_2, r_2)}$	f_Ω	id	id	id
$\phi_{(r_2, c_2)}$	$f_0 \circ \phi_{(r_2, b_2)}$	f_Ω	f_0	f_0	f_0
$\phi_{(r_2, n_2)}$	$\phi_{(r_2, e_2)} \circ \phi_{(r_2, c_2)}$	f_Ω	f_Ω	f_0	f_0
$\phi_{(r_2, e_2)}$	$(f_1 \circ \phi_{(r_2, n_2)}) \wedge (id_L \circ \phi_{(r_2, r_2)})$	f_Ω	id	id	id

Solving Data Flow Problem

- ▶ The above process gives solution to ϕ functions
- ▶ Use it to compute data flow information x_n associated with start of block n

$$x_{r_{\text{main}}} = \text{BoundaryInfo}$$

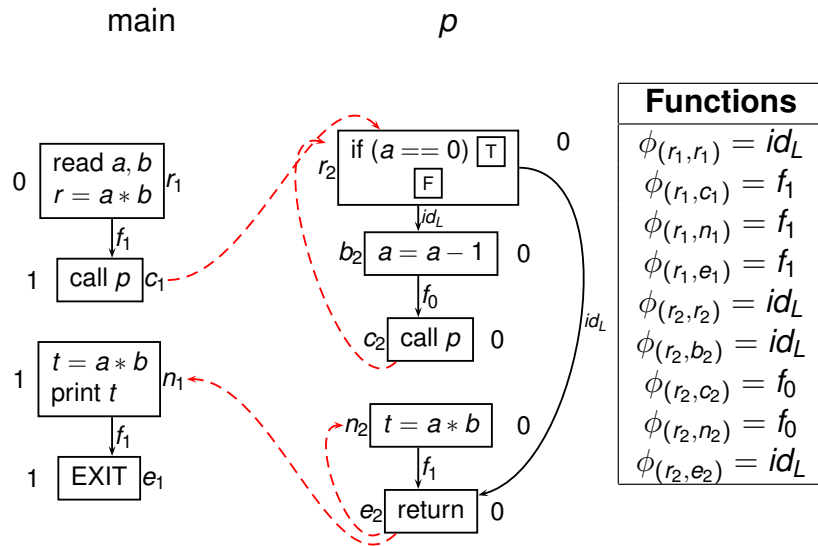
for each procedure p

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

$$x_n = \phi_{(r_p, n)} \quad n \in N_p - \{r_p\}$$

- ▶ Iterative algorithm for solution, maximal fixed point solution

Example



Interprocedural MOP

$$\Psi_n = \bigwedge \{f_q : q \in \text{IVP}(r_{\text{main}}, n)\} \in F \quad \forall n \in N^*$$

$$y_n = \Psi_n(\text{BoundaryInfo}) \quad \forall n \in N^*$$

y_n is the *meet-over-all-paths solution* (MOP).

IVP₀ Lemma

$$\phi(r_p, n) = \bigwedge \{f_q : q \in \text{IVP}_0(r_p, n)\} \quad \forall n \in N_p$$

Proof: By induction (**Exercise/Reading Assignment**)

MOP

$$\Psi_n = \bigwedge \{f_q : q \in \text{IVP}(r_{\text{main}}, n)\} \in F \quad \forall n \in N^*$$

$$\mathcal{X}_n = \bigwedge \{\phi(r_{p_i}, n) \circ \phi(r_{p_{i-1}}, c_{i-1}) \circ \dots \circ \phi(r_{p_1}, c_1) \mid c_i \text{ calls } p_{i+1}\}$$

Then

$$\Psi_n = \mathcal{X}_n$$

Proof: IVP₀ Lemma and Path decomposition

$$y_n = \Psi_n(\text{BoundaryInfo}) = \mathcal{X}_n(\text{BoundaryInfo})$$

MOP vs MFP

- ▶ F is distributive $\Rightarrow MFP = MOP$
- ▶ F is monotone $\Rightarrow MFP \leq MOP$

Practical Issues

- ▶ How to compute ϕ s effectively?
 - ▶ For general frameworks
 - ▶ L not finite
 - ▶ F not bounded
 - ▶ Does the solution process converge?
- ▶ How much space is required to represent ϕ functions?
- ▶ Is it possible to avoid explicit function compositions and meets?