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

Interprocedurally Valid Paths

- G^* ignores the special nature of call and return edges
- Not all paths in G^* are feasible

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

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

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)

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
- $\mathrm{IVP}(r_1, n)$: set of all interprocedurally valid paths from r_1 to n
- Path $q \in \mathrm{path}_{G^*}(r_1, n)$ is in $\mathrm{IVP}(r_1, n)$
  - iff sequence of all E^1 edges in q (denoted q_1)is *proper*

Proper sequence

- q_1 without any return edge is proper

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

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

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

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

- $\mathrm{IVP}_0(r_p, n)$ for procedure p and node $n \in N_p$

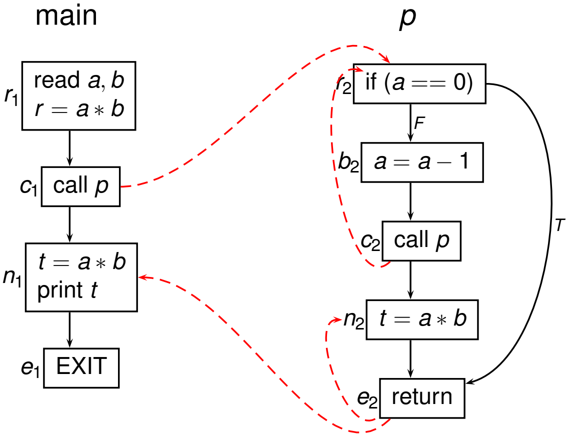
Interprocedurally Valid Complete Paths

- $\mathrm{IVP}_0(r_p, n)$ 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.

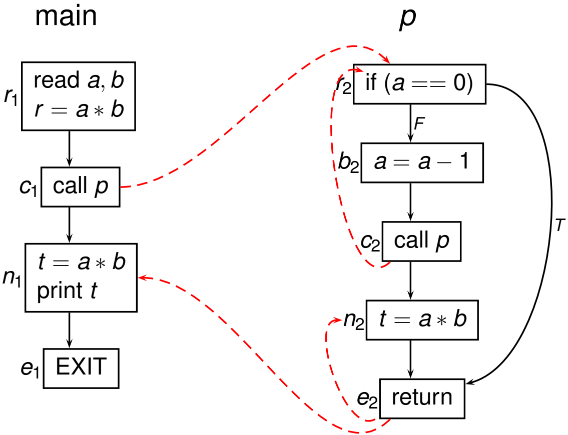
Interprocedurally Valid Complete Paths

- $\mathrm{IVP}_0(r_p, n)$ 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

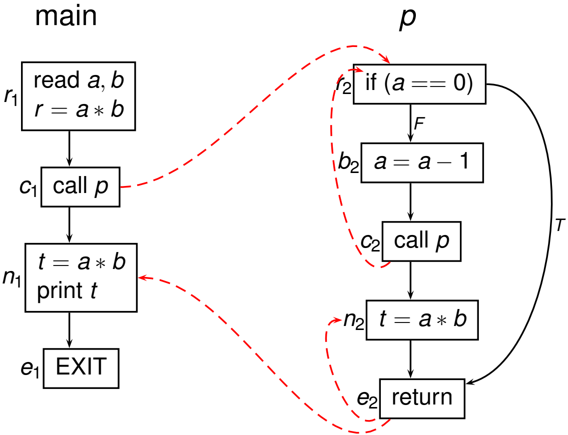

main
p
r_1
read a, b
$r = a * b$
c_1
call p
n_1
$t = a * b$
print t
e_1
EXIT
r_2
if $(a == 0)$
F
b_2
$a = a - 1$
c_2
call p
n_2
$t = a * b$
e_2
return
T

IVPs



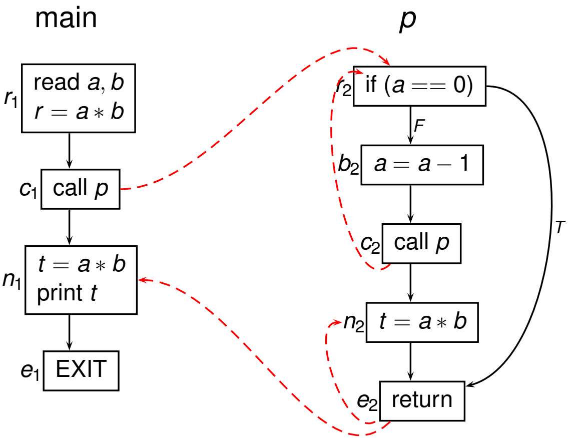
$$r_1 \to c_1 \to r_2 \to c_2 \to r_2 \to e_2 \to n_2 \to e_2 \to n_1 \to e_1$$

IVPs



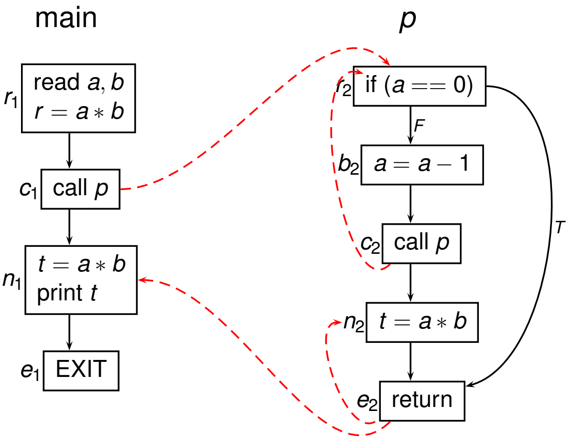
$$r_1 \rightarrow c_1 \rightarrow r_2 \rightarrow c_2 \rightarrow r_2 \rightarrow e_2 \rightarrow n_2 \rightarrow e_2 \rightarrow n_1 \rightarrow e_1$$
$$\in \mathrm{IVP}(r_1, e_1)$$

IVPs



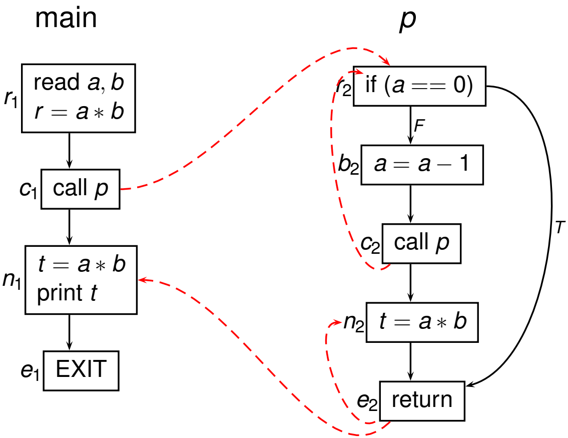
$$r_1 \rightarrow c_1 \rightarrow r_2 \rightarrow c_2 \rightarrow r_2 \rightarrow e_2 \rightarrow n_1 \rightarrow e_1$$

IVPs



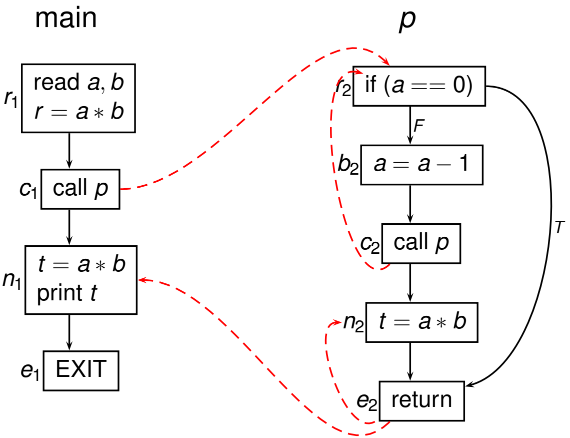
$$r_1 \rightarrow c_1 \rightarrow r_2 \rightarrow c_2 \rightarrow r_2 \rightarrow e_2 \rightarrow n_1 \rightarrow e_1 \notin \mathrm{IVP}(r_1, e_1)$$

IVPs



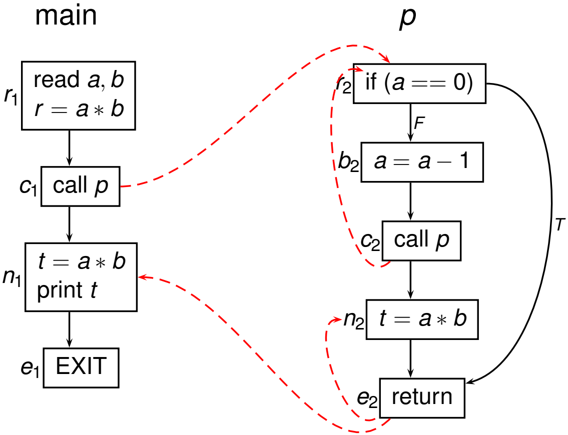
$$r_2 \rightarrow c_2 \rightarrow r_2 \rightarrow e_2 \rightarrow n_2$$

IVPs



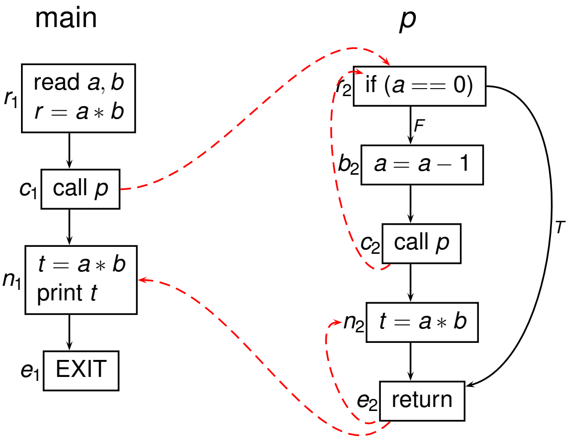
$$r_2 \to c_2 \to r_2 \to e_2 \to n_2 \in \text{IVP}_0(r_2, n_2)$$

IVPs



$$r_2 \to c_2 \to r_2 \to c_2 \to e_2 \to n_2$$

IVPs



$$r_2 \to c_2 \to r_2 \to c_2 \to e_2 \to n_2 \notin \mathrm{IVP}_0(r_2, n_2)$$

Path Decomposition

$$
\begin{aligned}
q &\in \mathrm{IVP}(r_{\mathrm{main}}, n) \\
&\Leftrightarrow \\
q &= q_1 \parallel (c_1, r_{p_2}) \parallel q_2 \parallel \cdots \parallel (c_{j-1}, r_{p_j}) \parallel q_j \\
& \text{where for each } i < j, q_i \in \mathrm{IVP}_0(r_{p_i}, c_i) and q_j \in \mathrm{IVP}_0(r_{p_j}, n)
\end{aligned}
$$

Functional Approach

- (L, F): a *distributive* data flow framework

Functional Approach

- (L, F): a *distributive* data flow framework
- Procedure p, node $n \in N_p$

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

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 $\mathrm{IVP}_0(r_p, n)$

Functional Approach Constraints

$$\begin{array}{rcll} \phi_{(r_p,r_p)} & \leq & id_L & \\ \phi_{(r_p,n)} & = & \bigwedge\limits_{(m,n)\in E_p} (h_{(m,n)} \circ \phi_{(r_p,m)}) & \text{for } n \in N_p \end{array}$$

where

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

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

$$\begin{aligned}
\phi^0_{(r_p,r_p)} &\leq id_L \\
\phi^0_{(r_p,n)} &\leq f_\Omega \qquad n \in N_p - \{r_p\}
\end{aligned}$$

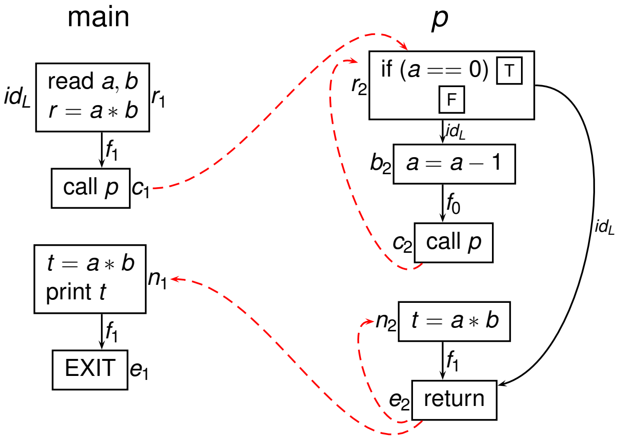
Solving ϕ Constraints

- Round-robin iterative approximations to initial values

$$
\begin{aligned}
\phi^0_{(r_p,r_p)} &\leq id_L \\
\phi^0_{(r_p,n)} &\leq f_\Omega \qquad n \in N_p - \{r_p\}
\end{aligned}
$$

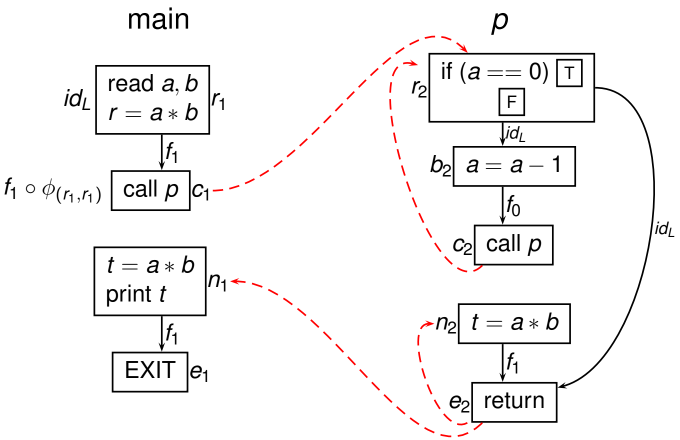
- Reach maximal fixed point

Example



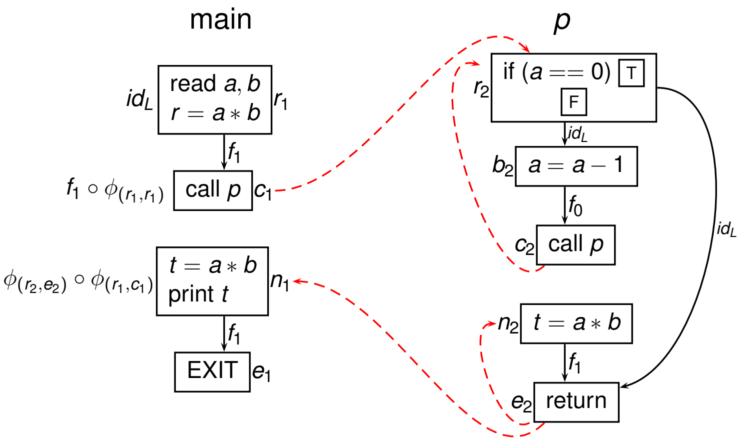
main
p
read a, b
r = a ∗ b
idL
r1
f1
call p
c1
t = a ∗ b
print t
n1
f1
EXIT
e1
if (a == 0)
T
F
r2
idL
b2
a = a − 1
f0
c2
call p
idL
n2
t = a ∗ b
f1
e2
return

Example

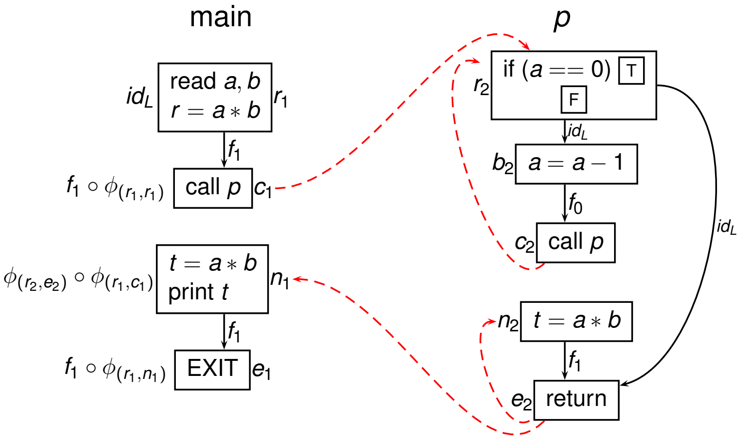


main
p
read a, b
$r = a * b$
id_L
r_1
f_1
$f_1 \circ \phi_{(r_1,r_1)}$
call p
c_1
$t = a * b$
print t
n_1
f_1
EXIT
e_1
if $(a == 0)$
T
F
r_2
id_L
b_2
$a = a - 1$
f_0
c_2
call p
id_L
n_2
$t = a * b$
f_1
e_2
return

Example

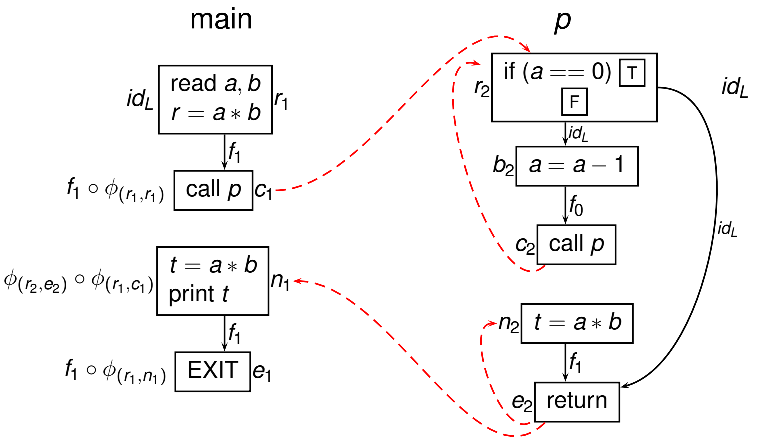

main
p
read a, b
$r = a * b$
id_L
r_1
f_1
call p
$f_1 \circ \phi_{(r_1,r_1)}$
c_1
$t = a * b$
print t
$\phi_{(r_2,e_2)} \circ \phi_{(r_1,c_1)}$
n_1
f_1
EXIT
e_1
if $(a == 0)$
T
F
r_2
id_L
b_2
$a = a - 1$
f_0
c_2
call p
id_L
n_2
$t = a * b$
f_1
e_2
return

Example

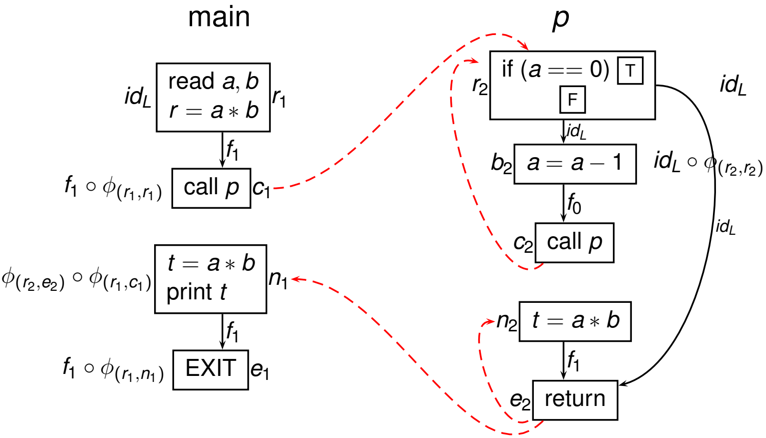


main
p
id_L
read a, b
$r = a * b$
r_1
f_1
$f_1 \circ \phi_{(r_1,r_1)}$
call p
c_1
$\phi_{(r_2,e_2)} \circ \phi_{(r_1,c_1)}$
$t = a * b$
print t
n_1
f_1
$f_1 \circ \phi_{(r_1,n_1)}$
EXIT
e_1
r_2
if ($a == 0$)
T
F
id_L
b_2
$a = a - 1$
f_0
c_2
call p
id_L
n_2
$t = a * b$
f_1
e_2
return

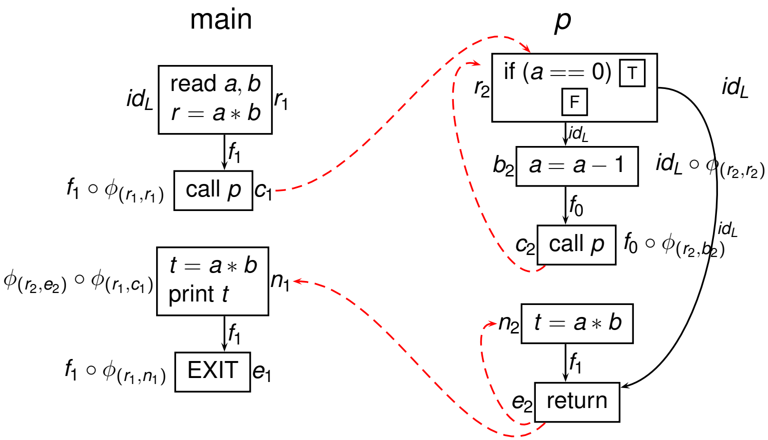
Example


main
p
read a, b
$r = a * b$
id_L
r_1
f_1
$f_1 \circ \phi_{(r_1,r_1)}$
call p
c_1
$t = a * b$
print t
$\phi_{(r_2,e_2)} \circ \phi_{(r_1,c_1)}$
n_1
f_1
$f_1 \circ \phi_{(r_1,n_1)}$
EXIT
e_1
if $(a == 0)$
T
F
r_2
id_L
id_L
b_2
$a = a - 1$
f_0
c_2
call p
id_L
n_2
$t = a * b$
f_1
e_2
return

Example

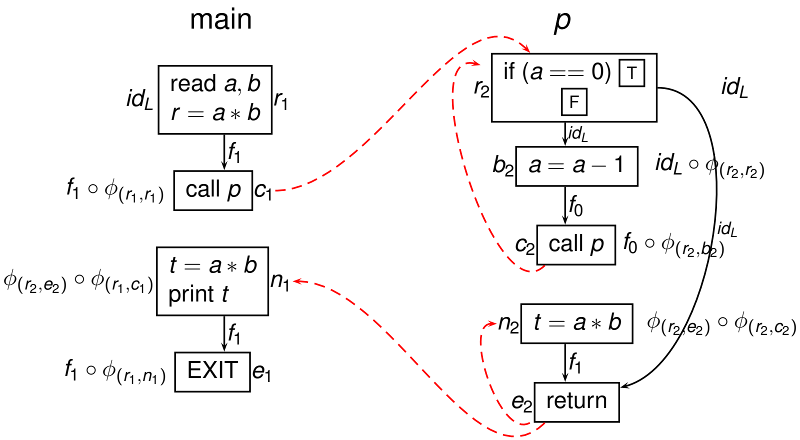

main
p
read a, b
r = a * b
call p
t = a * b
print t
EXIT
if (a == 0)
T
F
a = a − 1
call p
t = a * b
return

Example

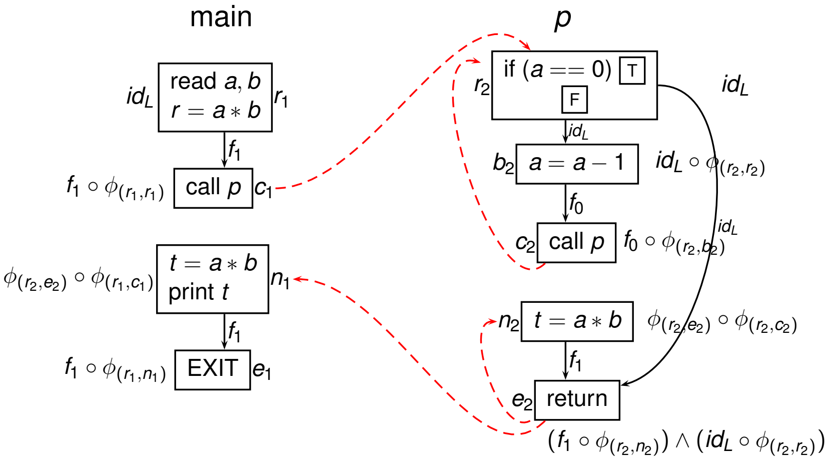


main
p
read a, b
r = a ∗ b
call p
t = a ∗ b
print t
EXIT
if (a == 0)
T
F
a = a − 1
call p
t = a ∗ b
return
id_L
r_1
f_1
$f_1 \circ \phi_{(r_1,r_1)}$
c_1
$\phi_{(r_2,e_2)} \circ \phi_{(r_1,c_1)}$
n_1
f_1
$f_1 \circ \phi_{(r_1,n_1)}$
e_1
r_2
id_L
id_L
b_2
$id_L \circ \phi_{(r_2,r_2)}$
f_0
c_2
$f_0 \circ \phi_{(r_2,b_2)}$
id_L
n_2
f_1
e_2

Example


main
p
read a, b
r = a * b
call p
t = a * b
print t
EXIT
if (a == 0)
T
F
a = a − 1
call p
t = a * b
return

Example



main
p
read a, b
r = a * b
idL
r1
f1
call p
c1
f1 ∘ φ(r1,r1)
t = a * b
print t
n1
φ(r2,e2) ∘ φ(r1,c1)
f1
EXIT
e1
f1 ∘ φ(r1,n1)
if (a == 0)
T
F
r2
idL
idL
b2
a = a − 1
idL ∘ φ(r2,r2)
f0
c2
call p
f0 ∘ φ(r2,b2)
idL
n2
t = a * b
φ(r2,e2) ∘ φ(r2,c2)
f1
e2
return
(f1 ∘ φ(r2,n2)) ∧ (idL ∘ φ(r2,r2))

Iterative Solution

Function	Constraint	Iteration #			
		Init	1^{st}	2^{nd}	3^{rd}

Iterative Solution

		Iteration #			
Function	**Constraint**	**Init**	1^{st}	2^{nd}	3^{rd}
$\phi_{(r_1,r_1)}$	id_L	id	id	id	id

Iterative Solution

		Iteration #			
Function	**Constraint**	**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

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

Iterative Solution

		Iteration #			
Function	**Constraint**	**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

Iterative Solution

		Iteration #			
Function	**Constraint**	**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

Iterative Solution

		Iteration #			
Function	**Constraint**	**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

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

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

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

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

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

for each procedure p

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

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

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

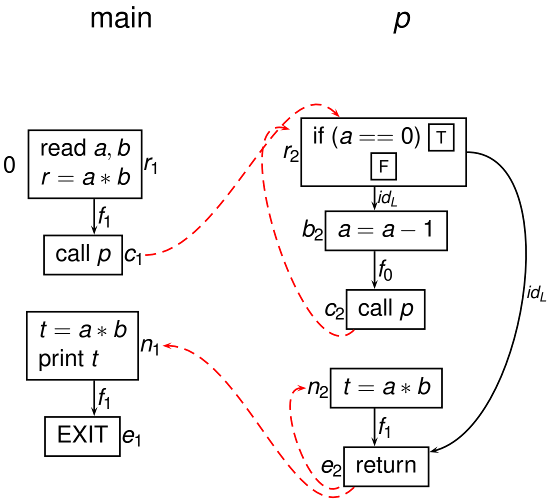
$$x_{r_{\text{main}}} = \mathit{BoundaryInfo}$$

for each procedure p

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

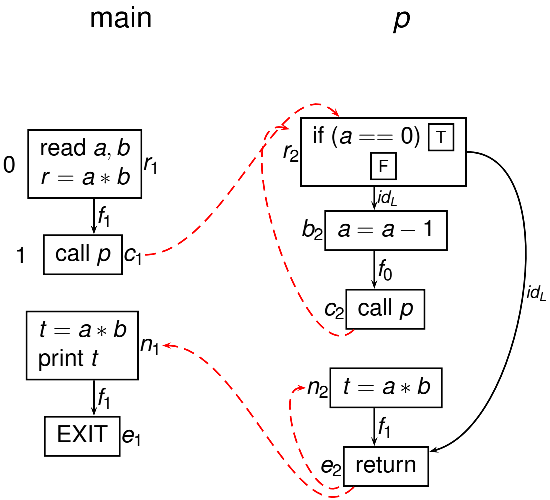
- Iterative algorithm for solution, maximal fixed point solution

Example



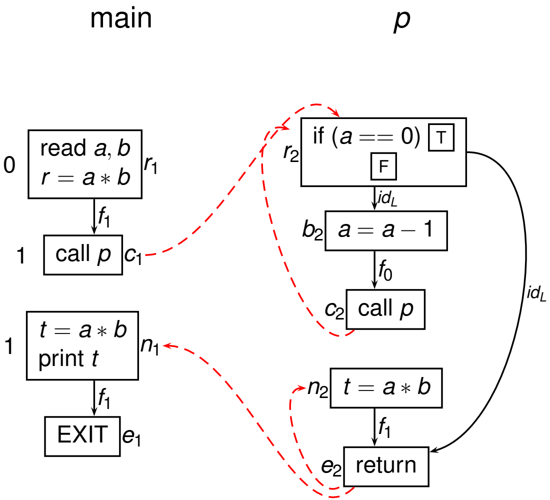
Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Example



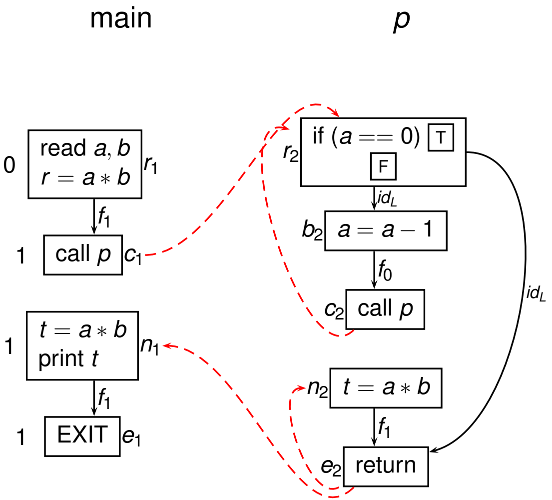
Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Example



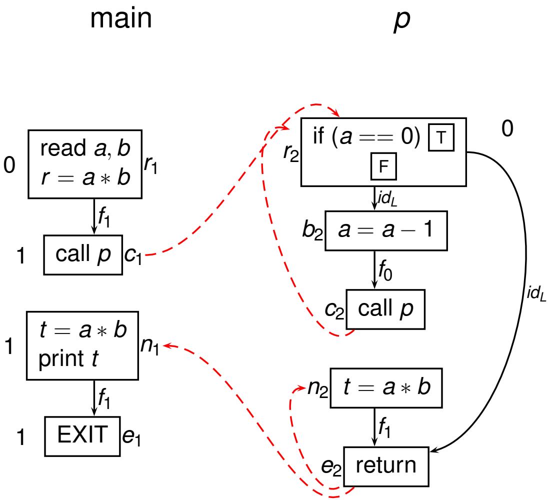
Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Example



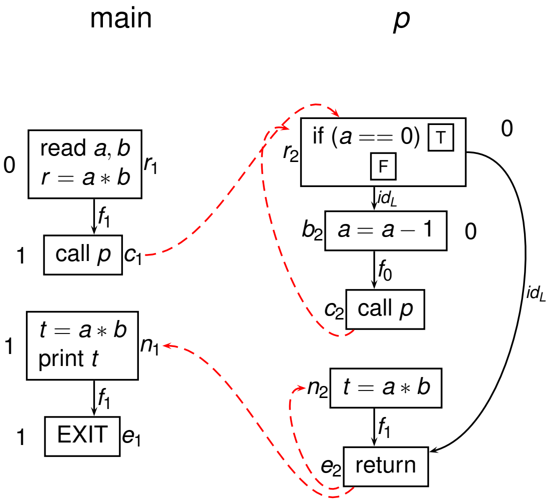
Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Example



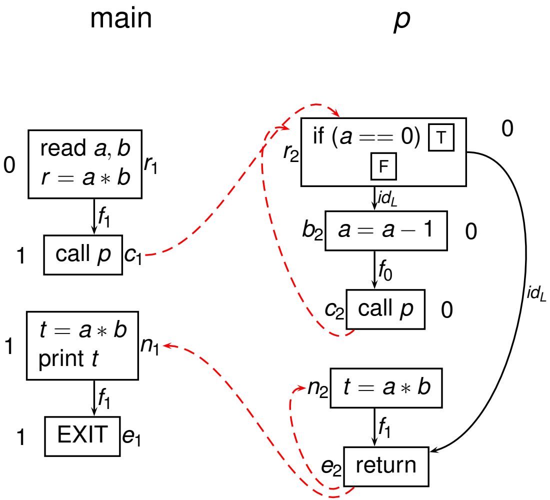
Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Example



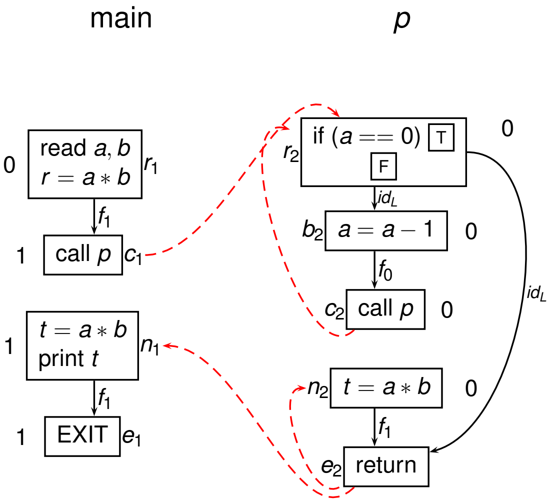
Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Example



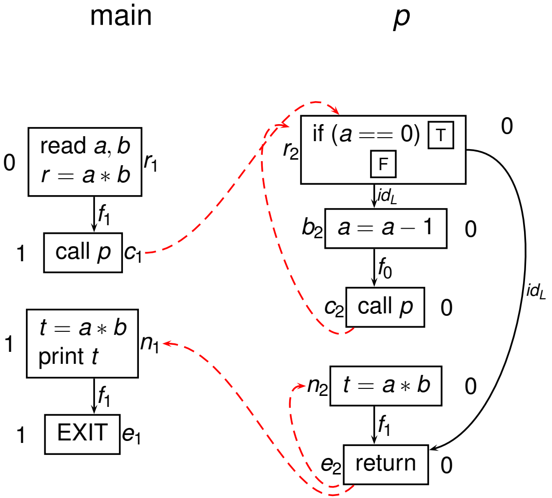
Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Example



Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Example



Functions
$\phi_{(r_1,r_1)} = id_L$
$\phi_{(r_1,c_1)} = f_1$
$\phi_{(r_1,n_1)} = f_1$
$\phi_{(r_1,e_1)} = f_1$
$\phi_{(r_2,r_2)} = id_L$
$\phi_{(r_2,b_2)} = id_L$
$\phi_{(r_2,c_2)} = f_0$
$\phi_{(r_2,n_2)} = f_0$
$\phi_{(r_2,e_2)} = id_L$

Interprocedural MOP

$$\Psi_n = \bigwedge \{ f_q : q \in \mathrm{IVP}(r_{\mathrm{main}}, n) \} \in F \qquad \forall n \in N^*$$
$$y_n = \Psi_n(BoundaryInfo) \qquad \forall n \in N^*$$

Interprocedural MOP

$$\begin{aligned}\Psi_n &= \bigwedge\{f_q : q \in \mathrm{IVP}(r_{\mathrm{main}}, n)\} \in F \qquad \forall n \in N^* \\ y_n &= \Psi_n(\mathit{BoundaryInfo}) \qquad \forall n \in N^*\end{aligned}$$

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

IVP_0 Lemma

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

IVP_0 Lemma

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

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

MOP

$$
\begin{aligned}
\Psi_n &= \bigwedge \{ f_q : q \in \mathrm{IVP}(r_{\mathrm{main}}, n) \} \in F \qquad \forall n \in N^* \\
\mathcal{X}_n &= \wedge \{ \phi_{(r_{p_j}, n)} \circ \phi_{(r_{p_{j-1}}, c_{j-1})} \circ \ldots \circ \phi_{(r_{p_1}, c_1)} \mid c_i \text{ calls } p_{i+1} \}
\end{aligned}
$$

MOP

$$\Psi_n = \bigwedge\{f_q : q \in \text{IVP}(r_{\text{main}}, n)\} \in F \qquad \forall n \in N^*$$
$$\mathcal{X}_n = \wedge\{\phi_{(r_{p_j},n)} \circ \phi_{(r_{p_{j-1}},c_{j-1})} \circ \ldots \circ \phi_{(r_{p_1},c_1)} \mid c_i \text{ calls } p_{i+1}\}$$

Then

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

Proof: IVP_0 Lemma and Path decomposition

MOP

$$
\begin{aligned}
\Psi_n &= \bigwedge \{ f_q : q \in \mathrm{IVP}(r_{\mathrm{main}}, n) \} \in F \qquad \forall n \in N^* \\
\mathcal{X}_n &= \wedge \{ \phi_{(r_{p_j}, n)} \circ \phi_{(r_{p_{j-1}}, c_{j-1})} \circ \ldots \circ \phi_{(r_{p_1}, c_1)} \mid c_i \text{ calls } p_{i+1} \}
\end{aligned}
$$

Then

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

Proof: IVP_0 Lemma and Path decomposition

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

MOP vs MFP

- F is distributive $\Rightarrow MFP = MOP$

MOP vs MFP

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

Practical Issues

- How to compute ϕs effectively?

Practical Issues

- How to compute ϕs effectively?
  - For general frameworks

Practical Issues

- How to compute ϕs effectively?
  - For general frameworks
    - L not finite

Practical Issues

- How to compute ϕs effectively?
  - For general frameworks
    - L not finite
    - F not bounded

Practical Issues

- How to compute ϕs effectively?
  - For general frameworks
    - L not finite
    - F not bounded
  - Does the solution process converge?

Practical Issues

- How to compute ϕs effectively?
  - For general frameworks
    - L not finite
    - F not bounded
  - Does the solution process converge?
- How much space is required to represent ϕ functions?

Practical Issues

- How to compute ϕs 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?