Interprocedural Data Flow Analysis

Is \( a \times b \) available at IN of \( n_1 \)?

Challenges

- Infeasible paths
- Recursion
- Function pointers and virtual functions
- Dynamic functions (functional programs)

Infeasible Paths

How to avoid data flowing along invalid paths?

\[ r_1 \rightarrow c_1 \rightarrow r_2 \rightarrow b_2 \rightarrow c_2 \rightarrow r_2 \rightarrow e_2 \rightarrow n_1 \]

\[ \text{main} \]

\[ \text{read } a, b \]

\[ r = a \times b \]

\[ \text{call } p \]

\[ t = a \times b \]

\[ \text{print } t \]

\[ e_1 \rightarrow \text{EXIT} \]

\[ \text{if } (a == 0) \]

\[ F \]

\[ a = a - 1 \]

\[ \text{call } p \]

\[ t = a \times b \]

\[ e_2 \rightarrow \text{return} \]

\[ T \]
Recursion

How to handle Infinite paths?
... → r₂ → c₂ → r₂ → c₂ → r₂ ...

main
r₁
read a, b
r = a ∗ b

c₁ call p

t = a ∗ b
print t

e₁ EXIT

r₂ if ( a == 0)
  b₂ a = a − 1
  c₂ call p

  n₁ t = a ∗ b
  n₁ return

Function Variables

- Target of a function can not be determined statically
- Function Pointers (including virtual functions)
  ```
  double (*fun)(double arg);
  ...
  if (cond)
      fun = sqrt;
  else
      fun = fabs;
  ...
  fun(x);
  ```
- Dynamically created functions (in functional languages)
- No static control flow graph!

Two Approaches

- Functional approach
  - procedures as structured blocks
  - input-output relation (functions) for each block
  - function used at call site to compute the effect of procedure on program state
- Call-strings approach
  - single flow graph for whole program
  - value of interest tagged with the history of unfinished procedure calls

Notations and Terminology

M. Sharir, and A. Pnueli. **Two Approaches to Inter-Procedural Data-Flow Analysis.**
In Jones and Muchnik, editors, Program Flow Analysis: Theory and Applications.
Prentice-Hall, 1981.
Control Flow Graph

Control Flow Graph for Procedure $p$

Assumptions

Data Flow Framework

Parameterless procedures, to ignore the problems of

- aliasing
- recursion stack for formal parameters

- No procedure variables (pointers, virtual functions etc.)

- $(L, F)$: data flow framework
- $L$: a meet-semilattice
  - Largest element $\Omega$
- $F$: space of propagation functions
  - Closed under composition and meet
  - Contains $id_L(x) = x$ and $f_\Omega(x) = \Omega$
- $f_{(m,n)} \in F$ represents propagation function for edge $(m,n)$ of control flow graph $G = (N, E)$
  - Change of DF values from the start of $m$, through $m$, to the start of $n$
Data Flow Equations

\[ x_r = \text{BoundaryInfo} \]
\[ x_n = \bigwedge_{(m,n) \in E} f_{(m,n)}(x_m) \quad n \in N - r \]

- MFP solution, approximation of MOP

\[ y_n = \bigwedge \{ f_p(\text{BoundaryInfo}) : p \in \text{path}_G(r, n) \} \quad n \in N \]

Functional Approach to Interprocedural Analysis

Functional Approach

- Procedures treated as structures of blocks
- Computes relationship between DF value at entry node and related data at any internal node of procedure
- At call site, DF value propagated directly using the computed relation

Interprocedural Flow Graph

First Representation:

\[ G = \bigcup \{ G_p : p \text{ is a procedure in program} \} \]
\[ G_p = (N_p, E_p, r_p) \]
\[ N_p = \text{set of all basic block of } p \]
\[ r_p = \text{root block of } p \]
\[ E_p = \text{set of edges of } p \]
\[ E_p = E_p^0 \cup E_p^1 \]
\[ (m, n) \in E_p^0 \iff \text{direct control transfer from } m \text{ to } n \]
\[ (m, n) \in E_p^1 \iff m \text{ is a call block, and } n \text{ immediately follows } m \]
Interprocedural Flow Graph: 1st Representation

\[
\begin{align*}
\text{main} & \quad r_1 \\
\quad E_{\text{main}}^0 & \quad \text{read } a, b \\
\quad c_1 & \quad \text{call } p \\
\quad E_{\text{main}}^1 & \quad t = a \ast b \\
\quad e_1 & \quad \text{print } t \\
\quad E_{\text{main}}^1 & \quad \text{EXIT}
\end{align*}
\]

\[
\begin{align*}
p & \quad r_2 \\
\quad E_p^0 & \quad \text{if } (a == 0) \\
\quad b_2 & \quad a = a - 1 \\
\quad E_p^1 & \quad c_2 \quad \text{call } p \\
\quad n_2 & \quad t = a \ast b \\
\quad E_p^1 & \quad \text{return}
\end{align*}
\]

Interprocedural Flow Graph: 2nd Representation

- Call edge \((m, n)\):
  - \(m\) is a call block, say calling \(p\)
  - \(n\) is root block of \(p\)
- Return edge \((m, n)\):
  - \(m\) is an exit block of \(p\)
  - \(n\) is a block immediately following a call to \(p\)
- Call edge \((m, r_p)\) corresponds to return edge \((e_q, n)\)
  - if \(p = q\) and
  - \((m, n) \in E_s^1\) for some procedure \(s\)

\[
\begin{align*}
\text{main} & \quad r_1 \\
\quad E_0^0 & \quad \text{read } a, b \\
\quad E_0^1 & \quad r = a \ast b \\
\quad c_1 & \quad \text{call } p \\
\quad E_0^1 & \quad t = a \ast b \\
\quad e_1 & \quad \text{EXIT}
\end{align*}
\]

\[
\begin{align*}
p & \quad r_2 \\
\quad E_0^1 & \quad \text{if } (a == 0) \\
\quad b_2 & \quad a = a - 1 \\
\quad E_0^1 & \quad c_2 \quad \text{call } p \\
\quad n_2 & \quad t = a \ast b \\
\quad E_0^1 & \quad \text{return}
\end{align*}
\]

\[
G^* = (N^*, E^*, r_1)
\]

- \(r_1\) = root block of main
- \(N^* = \bigcup_p N_p\)
- \(E^* = E_0^0 \cup E_1^1\)
- \(E_0^0 = \bigcup_p E_0^0_p\)

\[(m, n) \in E_1^1 \iff (m, n) \text{ is either a call edge or a return edge}\]
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

- IVP₀$(r_p, n)$ 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
  - $r_1 \xrightarrow{\text{read a, b}} r = a \ast b$
  - $c_1 \xrightarrow{\text{call p}}$
    - $t = a \ast b$
    - $n_1 \xrightarrow{\text{print t}}$
    - $e_1 \xrightarrow{\text{EXIT}}$
- $r_1 \to c_1 \to r_2 \to c_2 \to r_2 \to e_2 \to n_2 \to e_2 \to n_1 \to e_1 \in \text{IVP}(r_1, e_1)$
Path Decomposition

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