Intraprocedural Analysis

- Individual procedures are analyzed in isolation;
- worst-case assumption are made while processing call sites.

Example:

\[
\begin{align*}
X &= 3; \\
P(); \\
\ldots &= X
\end{align*}
\]

Does definition \(X=3\) reach this point? Yes
Is \(X\) a constant? No
We assume that \(P()\) does not kill any definition for reaching definitions analysis and we assume \(P()\) kills all constants that reach the call \(P()\).
Interprocedural Analysis

By considering interactions between calling and called procedures, analyze the whole program.

Challenges

• Scoping Rules
• Aliasing due to reference parameters

Procedure P (f1,f2)

\[ f1 = \ldots \]
\[ f2 = \ldots \]
\[ \ldots = f1 \] where if f1 defined \( f1 = \ldots \)

what about call P(X,X)?

Interprocedural Analysis

Challenges contd...

• Calling Context - cannot represent call-return using simple control flow edges.

Invalid paths are created by simple control flow edges causing definition of X at statement 1 to reach use of X at statement 4.
Handling Calling Context

Various Approaches:

1. Procedure Inlining - replace the call by procedure body.
   Drawbacks
   - Code size increases
   - Recursion causes problems

2. Call String Approach - add calling context information to data flow values by tagging them with call-return history.

3. Functional Approach
   Two phases
   - Analyze effect of procedure invocation independent of calling context → find a summary transfer function for the entire procedure
   - Analyze each procedure using summary functions for all called procedures
Functional Approach

(CL,F) distributed data flow framework

L - bounded semi-lattice
F - distributed function space

for each edge (m,n) where n is not a call site

f(m,n): L -> L E F flow function describing the effects of executing code in node m
(i.e., defined locally by inspecting the code)
Functional Approach Contd..

Goal: Determine the flow function to describe the effects of call sites: 

\[ \Phi(p, e_p) \in \text{Ef}: L \rightarrow L \]

describes the manner in which attributes are propagated from start of \( p \) to exit of \( p \).

Once summary functions are known, each procedure can be analyzed without descending into called procedures. 

→ calling context problem is handled approximately.

Functional Approach Contd..

Phase 1

compute function \( \Phi(p, e_p) \) for each procedure \( p \).

\( \Phi(p, n) \) maps data flow facts from entry of \( p \) to entry of node \( n \).

\[ \Phi(p, p) = \text{identity} \]

\[ \Phi(p, n) = \bigwedge_{(m, n) \in E} h(m, n) \circ \Phi(p, m) \]

where \( h(m, n) = \begin{cases} f(m, n) & \text{if } m \text{ is not a call site} \\ \Phi(p, e_p) & \text{if } m \text{ is a call site for procedure } p \end{cases} \]
Iteratively solve the equation system:

Initialization: \( \varphi^{0}(r_p, r_p) = \text{identity} \)

\[ \varphi^{n}(r_p, n) = f_{a} \rightarrow \text{constant function} \]

\[ \forall x \in L \quad f_{a}(x) = \omega \]

\( \omega = \text{undefined value} \)

Fixed point exists only if \( F \) is bounded

If \( L \) is finite then \( F \) is finite (i.e., bounded)

---

Phase 2:

Incorporate summary function at call sites to determine data flow solution.

\[ x^{[r_{\text{min}}]} : \perp \]

\[ x^{[r_{p}]} : \bigwedge \left\{ g_{q}^{(r_{k})} (x^{[r_{q}]}) : q \text{ is a procedure that}\right\} \]

\[ \text{removes a call site } c \text{ calling } p \]

\[ x^{[r_{p}]} : \varphi^{n}(r_{p}, n) (x^{[r_{p}]} ) \]

\[ \varphi^{n}(r_{q}, c) (x^{[r_{q}]} ) \hspace{1cm} \left\{ \begin{array}{l}
\text{if } r \quad \rightarrow r_{p} \text{ then } \varphi^{n}(r_{q}, c) (x^{[r_{q}]} ) \text{ of } c \text{ calling } p \\
\end{array} \right. \]
Example: Available Expressions

Available expressions: data flow analysis:

\[ L = \{ \alpha, \tau, \bot \} \]

Lattice:

- \( \alpha \) undefined
- \( \tau \) available
- \( \bot \) unavailable

Summary functions: analysis:

\[ F = \{ f_\alpha, f_\tau, f_\bot, \text{identity} \} \]

\[ f_\alpha \rightarrow \forall x \in L, f_\alpha(x) = \alpha \]

\[ f_\tau \rightarrow \forall x \in L, f_\tau(x) = \tau \] \quad \text{constant functions}

\[ f_\bot \rightarrow \forall x \in L, f_\bot(x) = \bot \]

identity \( \rightarrow \forall x \in L, \text{identity}(x) = x \)

Example: Available Expressions

Two operators need to be defined:

a) meet operator

b) composition operator

c) Meet Operator:
**Example: Available Expressions**

### Phase 2

#### Linhashahn

| $\gamma (r_{\text{min}}, r_{\text{min}})$ | 0d | 0d | 0d | local functions
| $\gamma (r_{\text{min}}, c,)$ | f_L | f_T | f_T | $f (r_{\text{min}}, c,): f_T$
| $\gamma (r_{\text{min}}, n_{\text{min}})$ | f_L | f_T | f_T | $f (n_{\text{min}}, r_{\text{min}}): f_T$
| $\gamma (r_{\text{min}}, n_{\text{min}})$ | f_L | f_T | f_T | $f (n_{\text{min}}, r_{\text{min}}): f_T$
| $\gamma (r_{T}, r_{T})$ | 0d | 0d | 0d | $f (r_{T}, r_{T}): 0d$
| $\gamma (r_{T}, n_{T})$ | f_L | f_T | f_T | $f (n_{T}, r_{T}): f_T$
| $\gamma (r_{T}, c_{T})$ | f_L | f_T | f_T | $f (c_{T}, r_{T}): f_T$
| $\gamma (r_{T}, n_{T})$ | f_L | f_T | f_T | $f (n_{T}, r_{T}): f_T$
| $\gamma (r_{T}, r_{T})$ | f_L | f_T | f_T | $f (r_{T}, r_{T}): f_T$

#### Summary functions

| $\gamma (r_{\text{max}}, n_{\text{min}})$ | $f (r_{\text{max}}, n_{\text{min}}): f_T$
| $\gamma (c, r_{T})$ | $f (c, r_{T}): f_T$
Example: Available Expressions

\[ x \in X_{\text{main}} \] 
\[ x[c_1] = \not\in (X_{\text{main}}, c_1) \times (X_{\text{main}}) \] 
\[ f_T (x[c_1]) = T \] 
\[ x[n_1] = \not\in (X_{\text{main}}, n_1) \times (X_{\text{main}}) \] 
\[ f_T (n_1) = T \] 
\[ x[e_{\text{main}}] = \not\in (X_{\text{main}}, e_{\text{main}}) \times (X_{\text{main}}) \] 
\[ f_T (e_{\text{main}}) = T \] 

\[ x[r_1] = \not\in (X_{\text{main}}, c_1) \times (X_{\text{main}}) \land \not\in (X_{\text{main}}, c_1) \times (X_{\text{main}}) \] 
\[ f_T (r_1) \land f_T (e_{\text{main}}) = T \] 
\[ x[n_2] = \not\in (X_{\text{main}}, n_2) \times (X_{\text{main}}) \] 
\[ x[n_2] = \not\in (X_{\text{main}}, n_2) \times (X_{\text{main}}) \] 
\[ x[e_2] = \not\in (X_{\text{main}}, e_2) \times (X_{\text{main}}) \] 
\[ x[e_2] = \not\in (X_{\text{main}}, e_2) \times (X_{\text{main}}) \] 

Sample Problems

Interprocedural Data Flow Analysis