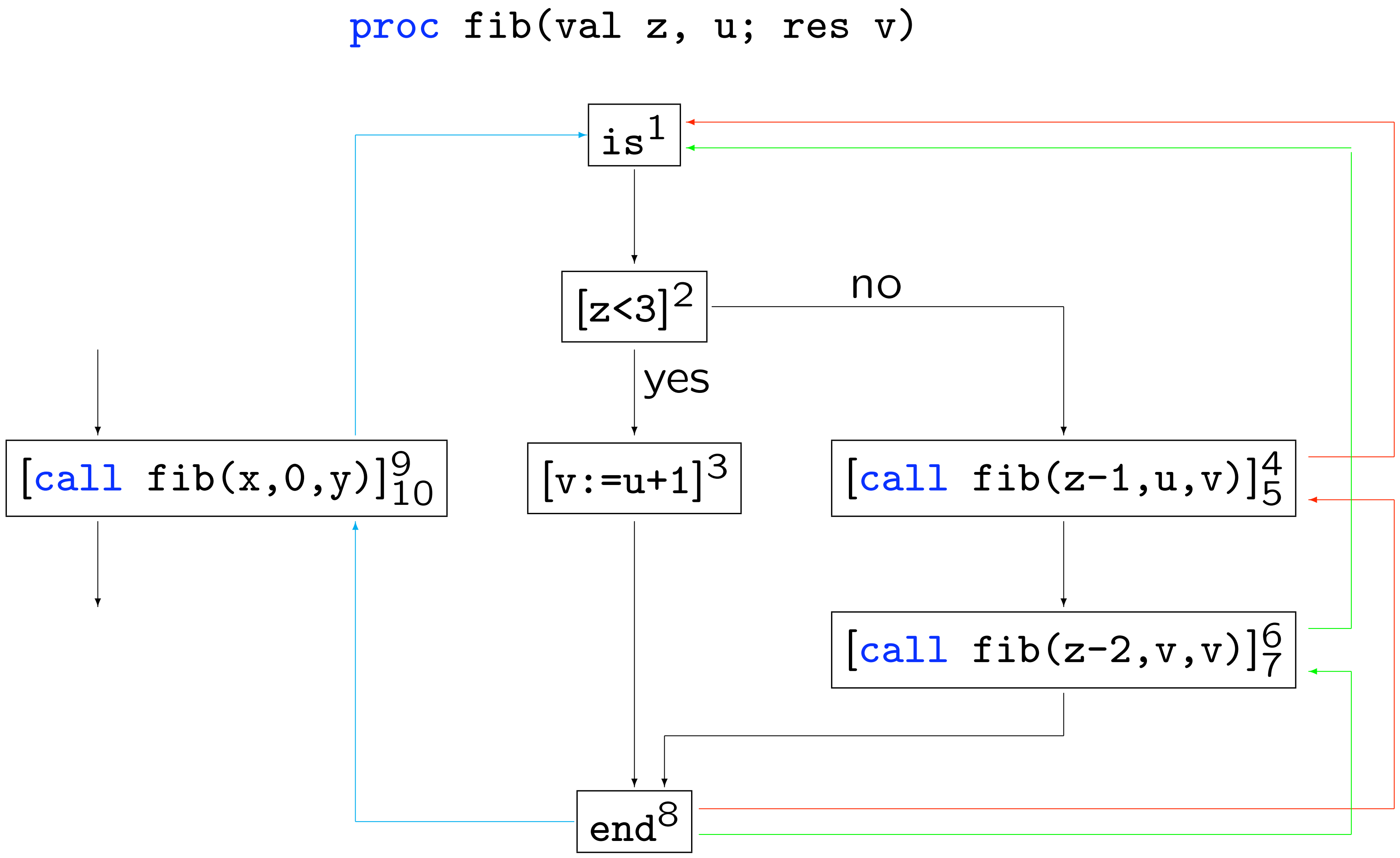
Interprocedural Analysis

- The problem
- MVP: "Meet" over Valid Paths
- Making context explicit
- Context based on call-strings
- Context based on assumption sets

(A restricted treatment; see the book for a more general treatment.)

The Problem: match entries with exits



Preliminaries

Syntax for procedures

Programs: $P_\star = \texttt{begin}\ D_\star\ S_\star\ \texttt{end}$

Declarations: $D ::= D; D \mid \texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}$

Statements: $S ::= \cdots \mid [\texttt{call}\ p(a, z)]^{\ell_c}_{\ell_r}$

Example:

```
begin  proc fib(val z, u; res v) is^1
              if [z<3]^2 then [v:=u+1]^3
              else ([call fib(z-1,u,v)]^4_5; [call fib(z-2,v,v)]^6_7)
       end^8;
       [call fib(x,0,y)]^9_10
end
```

Flow graphs for procedure calls

$$\begin{array}{rcl}
\mathit{init}([\texttt{call}\ p(a,z)]^{\ell_c}_{\ell_r}) & = & \ell_c \\
\mathit{final}([\texttt{call}\ p(a,z)]^{\ell_c}_{\ell_r}) & = & \{\ell_r\} \\
\mathit{blocks}([\texttt{call}\ p(a,z)]^{\ell_c}_{\ell_r}) & = & \{[\texttt{call}\ p(a,z)]^{\ell_c}_{\ell_r}\} \\
\mathit{labels}([\texttt{call}\ p(a,z)]^{\ell_c}_{\ell_r}) & = & \{\ell_c, \ell_r\} \\
\mathit{flow}([\texttt{call}\ p(a,z)]^{\ell_c}_{\ell_r}) & = & \{(\ell_c; \ell_n), (\ell_x; \ell_r)\} \\
 & \text{if} & \texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}\ \text{is in}\ D_\star
\end{array}$$

- $(\ell_c; \ell_n)$ is the flow corresponding to *calling* a procedure at ℓ_c and entering the procedure body at ℓ_n, and

- $(\ell_x; \ell_r)$ is the flow corresponding to exiting a procedure body at ℓ_x and *returning* to the call at ℓ_r.

Flow graphs for procedure declarations

For each procedure declaration $\texttt{proc}\ p(\texttt{val}\ x;\texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}$ of $D_\star$:

$$
\begin{array}{rcl}
\mathit{init}(p) & = & \ell_n \\
\mathit{final}(p) & = & \{\ell_x\} \\
\mathit{blocks}(p) & = & \{\texttt{is}^{\ell_n}, \texttt{end}^{\ell_x}\} \cup \mathit{blocks}(S) \\
\mathit{labels}(p) & = & \{\ell_n, \ell_x\} \cup \mathit{labels}(S) \\
\mathit{flow}(p) & = & \{(\ell_n, \mathit{init}(S))\} \cup \mathit{flow}(S) \cup \{(\ell, \ell_x) \mid \ell \in \mathit{final}(S)\}
\end{array}
$$

Flow graphs for programs

For the program $P_\star = \texttt{begin}\ D_\star\ S_\star\ \texttt{end}$:

$$
\begin{array}{rcl}
\mathit{init}_\star & = & \mathit{init}(S_\star) \\
\mathit{final}_\star & = & \mathit{final}(S_\star) \\
\mathit{blocks}_\star & = & \bigcup\{\mathit{blocks}(p) \mid \texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}\ \text{is in}\ D_\star\} \\
& & \cup\,\mathit{blocks}(S_\star) \\
\mathit{labels}_\star & = & \bigcup\{\mathit{labels}(p) \mid \texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}\ \text{is in}\ D_\star\} \\
& & \cup\,\mathit{labels}(S_\star) \\
\mathit{flow}_\star & = & \bigcup\{\mathit{flow}(p) \mid \texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}\ \text{is in}\ D_\star\} \\
& & \cup\,\mathit{flow}(S_\star) \\
\mathit{interflow}_\star & = & \{(\ell_c, \ell_n, \ell_x, \ell_r) \mid \texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}\ \text{is in}\ D_\star \\
& & \qquad\qquad \text{and}\ [\texttt{call}\ p(a, z)]^{\ell_c}_{\ell_r}\ \text{is in}\ S_\star\}
\end{array}
$$

Example:

```
begin  proc fib(val z, u; res v) is^1
               if [z<3]^2 then [v:=u+1]^3
               else ([call fib(z-1,u,v)]_5^4; [call fib(z-2,v,v)]_7^6)
       end^8;
       [call fib(x,0,y)]_10^9
end
```

We have

$$\begin{aligned}
\mathit{flow}_\star &= \{(1,2),(2,3),(3,8),\\
&\quad (2,4),(4;1),(8;5),(5,6),(6;1),(8;7),(7,8),\\
&\quad (9;1),(8;10)\}\\
\mathit{interflow}_\star &= \{(9,1,8,10),\ (4,1,8,5),\ (6,1,8,7)\}
\end{aligned}$$

and $\mathit{init}_\star = 9$ and $\mathit{final}_\star = \{10\}$.

A naive formulation

Treat the three kinds of flow in the same way:

flow	treat as
(ℓ_1, ℓ_2)	(ℓ_1, ℓ_2)
$(\ell_c; \ell_n)$	(ℓ_c, ℓ_n)
$(\ell_x; \ell_r)$	(ℓ_x, ℓ_r)

Equation system:

$$A_\bullet(\ell) = f_\ell(A_\circ(\ell))$$

$$A_\circ(\ell) = \bigsqcup\{A_\bullet(\ell') \mid (\ell', \ell) \in F \text{ or } (\ell', \ell) \in F \text{ or } (\ell', \ell) \in F\} \sqcup \iota_E^\ell$$

But there is no matching between entries and exits.

MVP: "Meet" over Valid Paths

Complete Paths

We need to match procedure entries and exits:

A *complete path* from ℓ_1 to ℓ_2 in $P_\star$ has proper nesting of procedure entries and exits; and a procedure returns to the point where it was called:

$$
\begin{array}{lll}
CP_{\ell_1,\ell_2} \longrightarrow \ell_1 & & \text{whenever } \ell_1 = \ell_2 \\
CP_{\ell_1,\ell_3} \longrightarrow \ell_1, CP_{\ell_2,\ell_3} & & \text{whenever } (\ell_1,\ell_2) \in \mathit{flow}_\star \\
CP_{\ell_c,\ell} \longrightarrow \ell_c, CP_{\ell_n,\ell_x}, CP_{\ell_r,\ell} & & \text{whenever } P_\star \text{ contains } [\mathtt{call}\ p(a,z)]_{\ell_r}^{\ell_c} \\
& & \text{and } \mathtt{proc}\ p(\mathtt{val}\ x; \mathtt{res}\ y)\ \mathtt{is}^{\ell_n}\ S\ \mathtt{end}^{\ell_x}
\end{array}
$$

More generally: whenever $(\ell_c, \ell_n, \ell_x, \ell_r)$ is an element of *interflow*$_\star$ (or *interflow*$_\star^R$ for backward analyses); see the book.

Valid Paths

A *valid path* starts at the entry node $init_\star$ of $P_\star$, all the procedure exits match the procedure entries but some procedures might be entered but not yet exited:

$VP_\star \longrightarrow VP_{init_\star,\ell}$	whenever $\ell \in \mathbf{Lab}_\star$
$VP_{\ell_1,\ell_2} \longrightarrow \ell_1$	whenever $\ell_1 = \ell_2$
$VP_{\ell_1,\ell_3} \longrightarrow \ell_1, VP_{\ell_2,\ell_3}$	whenever $(\ell_1, \ell_2) \in flow_\star$
$VP_{\ell_c,\ell} \longrightarrow \ell_c, CP_{\ell_n,\ell_x}, VP_{\ell_r,\ell}$	whenever $P_\star$ contains $[\texttt{call}\ p(a,z)]_{\ell_r}^{\ell_c}$ and $\texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}$
$VP_{\ell_c,\ell} \longrightarrow \ell_c, VP_{\ell_n,\ell}$	whenever $P_\star$ contains $[\texttt{call}\ p(a,z)]_{\ell_r}^{\ell_c}$ and $\texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}$

The MVP solution

$$MVP_\circ(\ell) = \bigsqcup\{f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in vpath_\circ(\ell)\}$$

$$MVP_\bullet(\ell) = \bigsqcup\{f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in vpath_\bullet(\ell)\}$$

where

$$vpath_\circ(\ell) = \{[\ell_1, \cdots, \ell_{n-1}] \mid n \geq 1 \wedge \ell_n = \ell \wedge [\ell_1, \cdots, \ell_n] \text{ is a valid path}\}$$

$$vpath_\bullet(\ell) = \{[\ell_1, \cdots, \ell_n] \mid n \geq 1 \wedge \ell_n = \ell \wedge [\ell_1, \cdots, \ell_n] \text{ is a valid path}\}$$

The MVP solution may be undecidable for lattices satisfying the Ascending Chain Condition, just as was the case for the MOP solution.

Making Context Explicit

Starting point: an instance $(L, \mathcal{F}, F, E, \iota, f_\cdot)$ of a Monotone Framework

- the analysis is forwards, i.e. $F = \mathit{flow}_\star$ and $E = \{\mathit{init}_\star\}$;

- the complete lattice is a powerset, i.e. $L = \mathcal{P}(D)$;

- the transfer functions in $\mathcal{F}$ are completely additive; and

- each f_ℓ is given by $f_\ell(Y) = \bigcup\{\phi_\ell(d) \mid d \in Y\}$ where $\phi_\ell : D \to \mathcal{P}(D)$.

(A restricted treatment; see the book for a more general treatment.)

An embellished monotone framework

- $L' = \mathcal{P}(\Delta \times D)$;

- the transfer functions in $\mathcal{F}'$ are completely additive; and

- each f'_ℓ is given by $f'_\ell(Z) = \bigcup\{\{\delta\} \times \phi_\ell(d) \mid (\delta, d) \in Z\}$.

Ignoring procedures, the data flow equations will take the form:

$$A_\bullet(\ell) = f'_\ell(A_\circ(\ell))$$

for all labels that do not label a procedure call

$$A_\circ(\ell) = \bigsqcup\{A_\bullet(\ell') \mid (\ell', \ell) \in F \text{ or } (\ell'; \ell) \in F\} \sqcup \iota'^{\ell}_E$$

for all labels (including those that label procedure calls)

Example:

Detection of Signs Analysis as a Monotone Framework:

$(L_{\mathsf{sign}}, \mathcal{F}_{\mathsf{sign}}, F, E, \iota_{\mathsf{sign}}, f_{\cdot}^{\mathsf{sign}})$ where $\mathbf{Sign} = \{\texttt{-}, \texttt{0}, \texttt{+}\}$ and

$$L_{\mathsf{sign}} = \mathcal{P}(\mathbf{Var}_\star \rightarrow \mathbf{Sign})$$

The transfer function f_ℓ^{sign} associated with the assignment $[x := a]^\ell$ is

$$f_\ell^{\mathsf{sign}}(Y) = \bigcup\{\phi_\ell^{\mathsf{sign}}(\sigma^{\mathsf{sign}}) \mid \sigma^{\mathsf{sign}} \in Y\}$$

where $Y \subseteq \mathbf{Var}_\star \rightarrow \mathbf{Sign}$ and

$$\phi_\ell^{\mathsf{sign}}(\sigma^{\mathsf{sign}}) = \{\sigma^{\mathsf{sign}}[x \mapsto s] \mid s \in \mathcal{A}_{\mathsf{sign}}[\![a]\!](\sigma^{\mathsf{sign}})\}$$

Example (cont.):

Detection of Signs Analysis as an embellished monotone framework

$$L'_{\mathsf{sign}} = \mathcal{P}(\boldsymbol{\Delta} \times \mathbf{(Var_\star \to Sign)})$$

The transfer function associated with $[x := a]^\ell$ will now be:

$$f_\ell^{\mathsf{sign}'}(Z) = \bigcup\{\{\delta\} \times \phi_\ell^{\mathsf{sign}}(\sigma^{\mathsf{sign}}) \mid (\delta, \sigma^{\mathsf{sign}}) \in Z\}$$

Transfer functions for procedure declarations

Procedure declarations

$$\texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x}$$

have two transfer functions, one for entry and one for exit:

$$f_{\ell_n}, f_{\ell_x} : \mathcal{P}(\Delta \times D) \to \mathcal{P}(\Delta \times D)$$

For simplicity we take both to be the identity function (thus incorporating procedure entry as part of procedure call, and procedure exit as part of procedure return).

Transfer functions for procedure calls

Procedure calls $[\texttt{call } p(a,z)]_{\ell_r}^{\ell_c}$ have two transfer functions:

For the *procedure call*

$$f_{\ell_c}^1 : \mathcal{P}(\Delta \times D) \to \mathcal{P}(\Delta \times D)$$

and it is used in the equation:

$$A_\bullet(\ell_c) \;=\; f_{\ell_c}^1(A_\circ(\ell_c)) \quad \text{for all procedure calls } [\texttt{call } p(a,z)]_{\ell_r}^{\ell_c}$$

For the *procedure return*

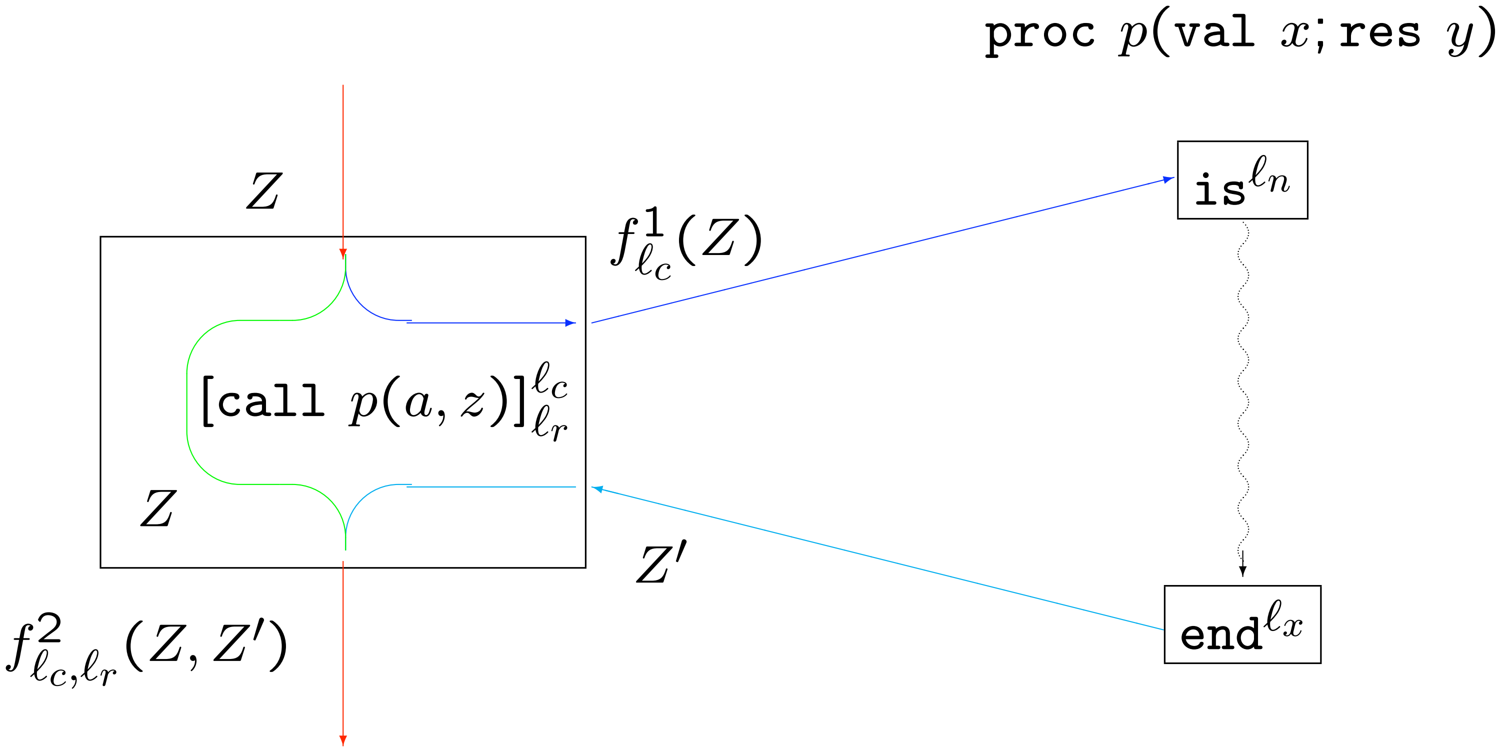
$$f_{\ell_c,\ell_r}^2 : \boxed{\mathcal{P}(\Delta \times D)} \times \mathcal{P}(\Delta \times D) \to \mathcal{P}(\Delta \times D)$$

and it is used in the equation:

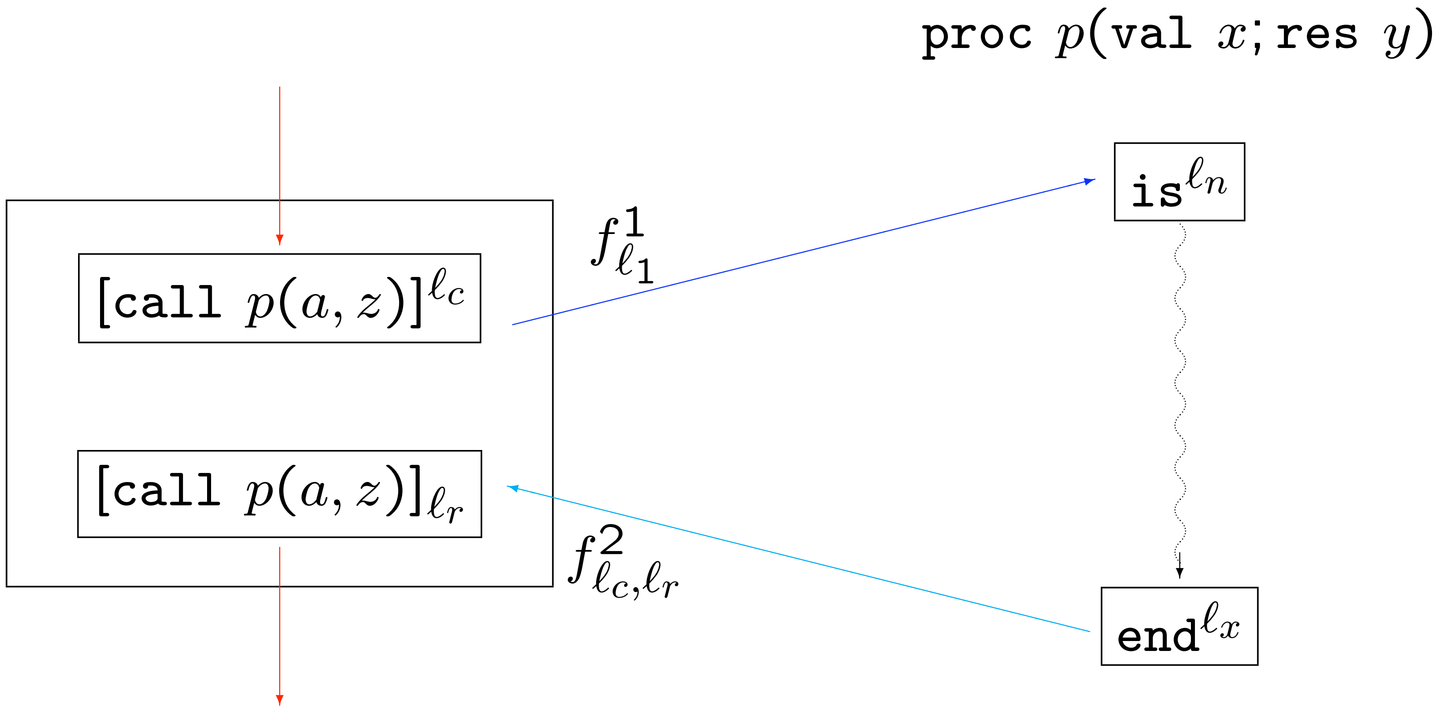
$$A_\bullet(\ell_r) \;=\; f_{\ell_c,\ell_r}^2(\boxed{A_\circ(\ell_c)}, A_\circ(\ell_r)) \quad \text{for all procedure calls } [\texttt{call } p(a,z)]_{\ell_r}^{\ell_c}$$

(Note that $A_\circ(\ell_r)$ will equal $A_\bullet(\ell_x)$ for the relevant procedure exit.)

Procedure calls and returns



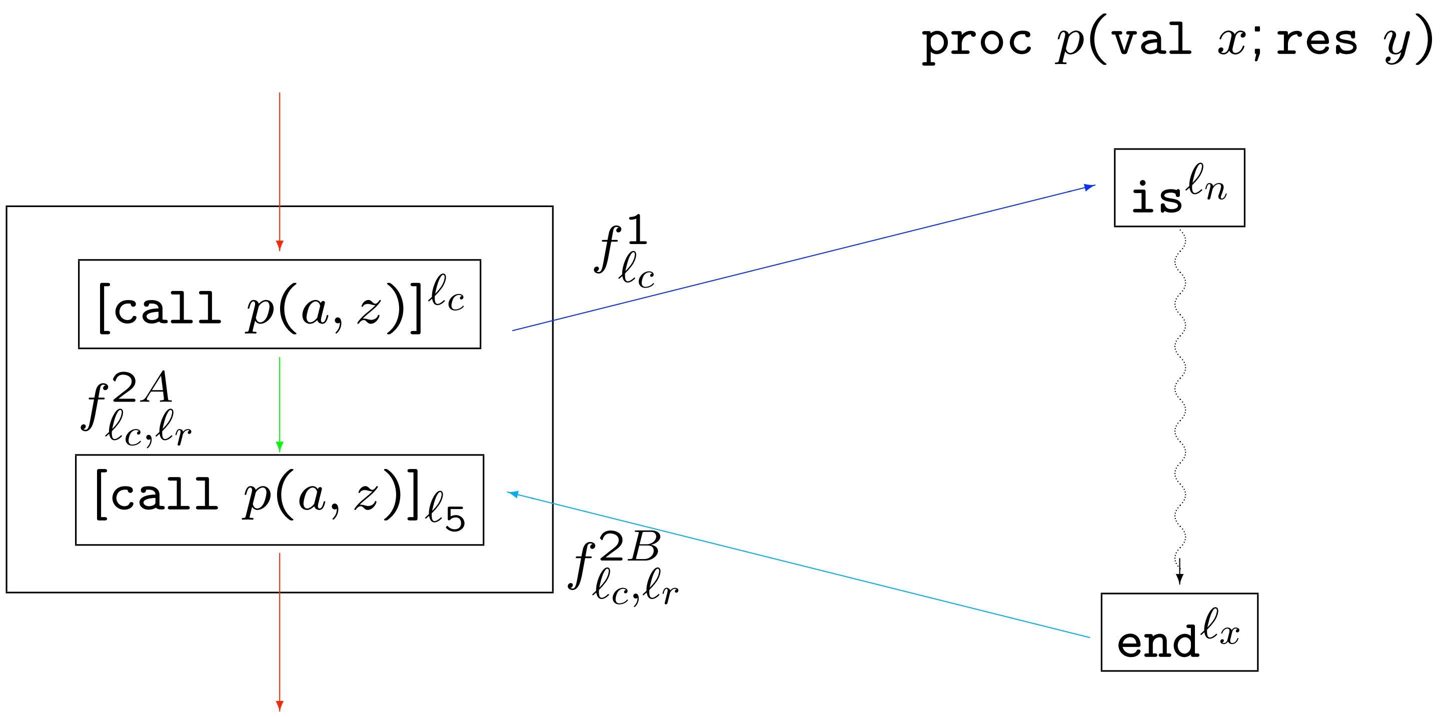
Variation 1: ignore calling context upon return



$$f^1_{\ell_c}(Z) = \bigcup \{ \{\delta'\} \times \phi^1_{\ell_c}(d) \mid \ (\delta, d) \in Z \ \wedge \delta' = \cdots \delta \cdots d \cdots Z \cdots \}$$

$$f^2_{\ell_c,\ell_r}(Z, Z') = f^2_{\ell_r}(Z')$$

Variation 2: joining contexts upon return



$$f^1_{\ell_c}(Z) = \bigcup\{\{\delta'\} \times \phi^1_{\ell_c}(d) \mid (\delta, d) \in Z \;\wedge \delta' = \cdots\delta\cdots d\cdots Z\cdots\}$$

$$f^2_{\ell_c,\ell_r}(Z, Z') = f^{2A}_{\ell_c,\ell_r}(Z) \sqcup f^{2B}_{\ell_c,\ell_r}(Z')$$

Different Kinds of Context

- Call Strings — contexts based on control
  - Call strings of unbounded length
  - Call strings of bounded length (k)
- Assumption Sets — contexts based on data
  - Large assumption sets $(k = 1)$
  - Small assumption sets $(k = 1)$

Call Strings of Unbounded Length

$$\boxed{\Delta} = \mathbf{Lab}^*$$

Transfer functions for procedure call

$$f^1_{\ell_c}(Z) = \bigcup\{\{\delta'\} \times \phi^1_{\ell_c}(d) \mid \begin{array}{l}(\delta, d) \in Z \ \wedge \\ \delta' = [\delta, \ell_c]\}\end{array}$$

$$f^2_{\ell_c,\ell_r}(Z, Z') = \bigcup\{\{\delta\} \times \phi^2_{\ell_c,\ell_r}(d, d') \mid \begin{array}{l}(\delta, d) \in Z \ \wedge \\ (\delta', d') \in Z' \ \wedge \ \delta' = [\delta, \ell_c]\}\end{array}$$

Example:

Recalling the statements:

$$\texttt{proc}\ p(\texttt{val}\ x; \texttt{res}\ y)\ \texttt{is}^{\ell_n}\ S\ \texttt{end}^{\ell_x} \qquad\qquad [\texttt{call}\ p(a,z)]^{\ell_c}_{\ell_r}$$

Detection of Signs Analysis:

$$\phi^{\mathsf{sign1}}_{\ell_c}(\sigma^{\mathsf{sign}}) = \{\sigma^{\mathsf{sign}}\overbrace{[x \mapsto s][y \mapsto s']}^{\text{initialise formals}} \mid s \in \mathcal{A}_{\mathsf{sign}}[\![a]\!](\sigma^{\mathsf{sign}}), s' \in \{\texttt{-}, 0, \texttt{+}\}\}$$

$$\phi^{\mathsf{sign2}}_{\ell_c,\ell_r}(\sigma^{\mathsf{sign}}_1, \sigma^{\mathsf{sign}}_2) = \{\sigma^{\mathsf{sign}}_2\underbrace{[x \mapsto \sigma^{\mathsf{sign}}_1(x)][y \mapsto \sigma^{\mathsf{sign}}_1(y)]}_{\text{restore formals}}\underbrace{[z \mapsto \sigma^{\mathsf{sign}}_2(y)]}_{\text{return result}}\}$$

Call Strings of Bounded Length

$$\boxed{\Delta} = \mathbf{Lab}^{\leq k}$$

Transfer functions for procedure call

$$f^1_{\ell_c}(Z) = \bigcup\{\{\delta'\} \times \phi^1_{\ell_c}(d) \mid \begin{array}{l}(\delta, d) \in Z \ \wedge \\ \delta' = \lceil \delta, \ell_c \rceil_k\}\end{array}$$

$$f^2_{\ell_c,\ell_r}(Z, Z') = \bigcup\{\{\delta\} \times \phi^2_{\ell_c,\ell_r}(d, d') \mid \begin{array}{l}(\delta, d) \in Z \ \wedge \\ (\delta', d') \in Z' \ \wedge \ \delta' = \lceil \delta, \ell_c \rceil_k\}\end{array}$$

A special case: call strings of length $k = 0$

$$\Delta = \{\Lambda\}$$

Note: this is equivalent to having no context information!

Specialising the transfer functions:

$$f^1_{\ell_c}(Y) = \bigcup\{\phi^1_{\ell_c}(d) \mid d \in Y\}$$

$$f^2_{\ell_c,\ell_r}(Y, Y') = \bigcup\{\phi^2_{\ell_c,\ell_r}(d, d') \mid d \in Y \;\wedge\; d' \in Y'\}$$

(We use that $\mathcal{P}(\Delta \times D)$ isomorphic to $\mathcal{P}(D)$.)

A special case: call strings of length $k = 1$

$$\Delta = \mathbf{Lab} \cup \{\Lambda\}$$

Specialising the transfer functions:

$$f^1_{\ell_c}(Z) = \bigcup \{\{\ell_c\} \times \phi^1_{\ell_c}(d) \mid (\delta, d) \in Z\}$$

$$f^2_{\ell_c,\ell_r}(Z, Z') = \bigcup \{\{\delta\} \times \phi^2_{\ell_c,\ell_r}(d, d') \mid (\delta, d) \in Z \ \wedge \ (\ell_c, d') \in Z'\}$$

Large Assumption Sets $(k = 1)$

$$\Delta = \mathcal{P}(D)$$

Transfer functions for procedure call

$$f^1_{\ell_c}(Z) = \bigcup\{\{\delta'\} \times \phi^1_{\ell_c}(d) \mid (\delta, d) \in Z \wedge \delta' = \{d'' \mid (\delta, d'') \in Z\}\}$$

$$f^2_{\ell_c,\ell_r}(Z, Z') = \bigcup\{\{\delta\} \times \phi^2_{\ell_c,\ell_r}(d, d') \mid (\delta, d) \in Z \wedge (\delta', d') \in Z' \wedge \delta' = \{d'' \mid (\delta, d'') \in Z\}\}$$

Small Assumption Sets $(k = 1)$

$$\boxed{\Delta} = D$$

Transfer function for procedure call

$$f^1_{\ell_c}(Z) = \bigcup\{\{\boxed{d}\} \times \phi^1_{\ell_c}(d) \mid (\delta, \boxed{d}) \in Z\}$$

$$f^2_{\ell_c,\ell_r}(Z, Z') = \bigcup\{\{\delta\} \times \phi^2_{\ell_c,\ell_r}(d, d') \mid \begin{array}{l}(\delta, d) \in Z \;\wedge \\ (d, d') \in Z'\}\end{array}$$