Static Program Analysis

Lecture 18: Interprocedural Dataflow Analysis II (Fixpoint Solution)

Summer Semester 2018

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ss-18/spa/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Outline of Lecture 18

Recap: Interprocedural Dataflow Analysis

Extending the Syntax

Syntactic categories:

Category	Domain	Meta variable
Procedure identifiers	$Pid = \{\mathtt{P}, \mathtt{Q}, \ldots\}$	P
Procedure declarations	$PDec$	p
Commands (statements)	Cmd	c

Context-free grammar

$$p ::= \mathtt{proc}\ [P(\mathtt{val}\ x, \mathtt{res}\ y)]^{l_n}\ \mathtt{is}\ c\ [\mathtt{end}]^{l_x}; p \mid \varepsilon \in PDec$$
$$c ::= [\mathtt{skip}]^{l} \mid [x := a]^{l} \mid c_1; c_2 \mid \mathtt{if}\ [b]^{l}\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\ \mathtt{end} \mid$$
$$\mathtt{while}\ [b]^{l}\ \mathtt{do}\ c\ \mathtt{end} \mid [\mathtt{call}\ P(a, x)]^{l_c}_{l_r} \in Cmd$$

- All labels and procedure names in program $p\,c$ distinct
- In $\mathtt{proc}\ [P(\mathtt{val}\ x, \mathtt{res}\ y)]^{l_n}\ \mathtt{is}\ c\ [\mathtt{end}]^{l_x}$, l_n / l_x refers to the entry / exit of P
- In $[\mathtt{call}\ P(a, x)]^{l_c}_{l_r}$, l_c / l_r refers to the call of / return from P
- First parameter call-by-value (input), second call-by-result (output)

Recap: Interprocedural Dataflow Analysis

Naive Formulation

- **Attempt:** directly transfer techniques from intraprocedural analysis
  $\Longrightarrow$ treat $(l_c; l_n)$ like (l_c, l_n) and $(l_x; l_r)$ like (l_x, l_r)
- Given: dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$
- For each procedure call $[\texttt{call } P(a,x)]^{l_c}_{l_r}$:
  transfer functions $\varphi_{l_c}, \varphi_{l_r} : D \to D$ (definition later)
- For each procedure declaration $\texttt{proc } [P(\texttt{val } x, \texttt{res } y)]^{l_n} \texttt{ is } c\ [\texttt{end}]^{l_x}$:
  transfer functions $\varphi_{l_n}, \varphi_{l_x} : D \to D$ (definition later)
- Induces equation system

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{\varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F \text{ or } (l'; l) \in F\} & \text{otherwise} \end{cases}$$

- **Problem:** procedure calls $(l_c; l_n)$ and procedure returns $(l_x; l_r)$ treated like goto's
  $\Longrightarrow$ nesting of calls and returns ignored
  $\Longrightarrow$ too many paths considered
  $\Longrightarrow$ analysis information possibly imprecise (but still sound as all "valid" paths covered)

Recap: Interprocedural Dataflow Analysis

Valid Paths

- Consider only paths with correct nesting of procedure calls and returns
- Yields MVP solution (Meet over all Valid Paths)

Definition (Valid path fragments)

Given a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ and $l_1, l_2 \in Lab$, the set of valid paths from l_1 to l_2 is generated by the nonterminal symbol $P[l_1, l_2]$ according to the following context-free grammar:

$$
\begin{aligned}
P[l_1, l_2] &\to l_1 && \text{whenever } l_1 = l_2 \\
P[l_1, l_3] &\to l_1, P[l_2, l_3] && \text{whenever } (l_1, l_2) \in F \\
P[l_c, l] &\to l_c, P[l_n, l_x], P[l_r, l] && \text{whenever } (l_c, l_n, l_x, l_r) \in \text{iflow}
\end{aligned}
$$

The MVP Solution I

Definition (Complete valid paths)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. For every $l \in Lab$, the set of valid paths up to l is given by

$$VPath(l) := \{[l_1, \ldots, l_{k-1}] \mid k \geq 1, l_1 \in E, l_k = l, [l_1, \ldots, l_k] \text{ valid path from } l_1 \text{ to } l_k\}.$$

For $\pi = [l_1, \ldots, l_{k-1}] \in VPath(l)$, we define the transfer function $\varphi_\pi : D \to D$ by

$$\varphi_\pi := \varphi_{l_{k-1}} \circ \ldots \circ \varphi_{l_1} \circ \mathrm{id}_D$$

(so that $\varphi_{[]} = \mathrm{id}_D$).

The MVP Solution II

Definition (MVP solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system where $Lab = \{l_1, \ldots, l_n\}$. The MVP solution for S is determined by

$$\mathrm{mvp}(S) := (\mathrm{mvp}(l_1), \ldots, \mathrm{mvp}(l_n)) \in D^n$$

where, for every $l \in Lab$,

$$\mathrm{mvp}(l) := \bigsqcup \{\varphi_\pi(\iota) \mid \pi \in VPath(l)\}.$$

Corollary

1. $\mathrm{mvp}(S) \sqsubseteq \mathrm{mop}(S)$
2. *The MVP solution is undecidable.*

Proof.

1. since $VPath(l) \subseteq Path(l)$ for every $l \in Lab$
2. as $\mathrm{mvp}(S) = \mathrm{mop}(S)$ in intraprocedural case and MOP solution undecidable (Thm. 6.6) □

Outline of Lecture 18

The Interprocedural Fixpoint Solution

- **Goal:** adapt fixpoint solution to avoid invalid paths
- For simplification: consider only forward problems

The Interprocedural Fixpoint Solution

- **Goal:** adapt fixpoint solution to avoid invalid paths
- For simplification: consider only forward problems
- **Approach:** represent combined effect of procedure execution by procedure summaries (to be used in node transfer function of return label)

The Interprocedural Fixpoint Solution

- **Goal:** adapt fixpoint solution to avoid invalid paths
- For simplification: consider only forward problems
- **Approach:** represent combined effect of procedure execution by procedure summaries (to be used in node transfer function of return label)

$\Rightarrow$ Two types of functions:

- Node transfer functions φ_l:
  - for non-procedural constructs (`skip`, assignments, tests): as before (specific to analysis)
  - for procedure call (l_c): parameter passing (specific to analysis)
  - for procedure entry (l_n): initialisation (specific to analysis)
  - for procedure exit (l_x): always identity
  - for procedure return (l_r): combine call context with procedure result obtained from summary function

$$\varphi_{l_r}(d) = \text{combine}(d, \Phi_{l_x}(\varphi_{l_c}(d)))$$

(combine specific to analysis)

The Interprocedural Fixpoint Solution

- **Goal:** adapt fixpoint solution to avoid invalid paths
- For simplification: consider only forward problems
- **Approach:** represent combined effect of procedure execution by procedure summaries (to be used in node transfer function of return label)

$\Rightarrow$ Two types of functions:

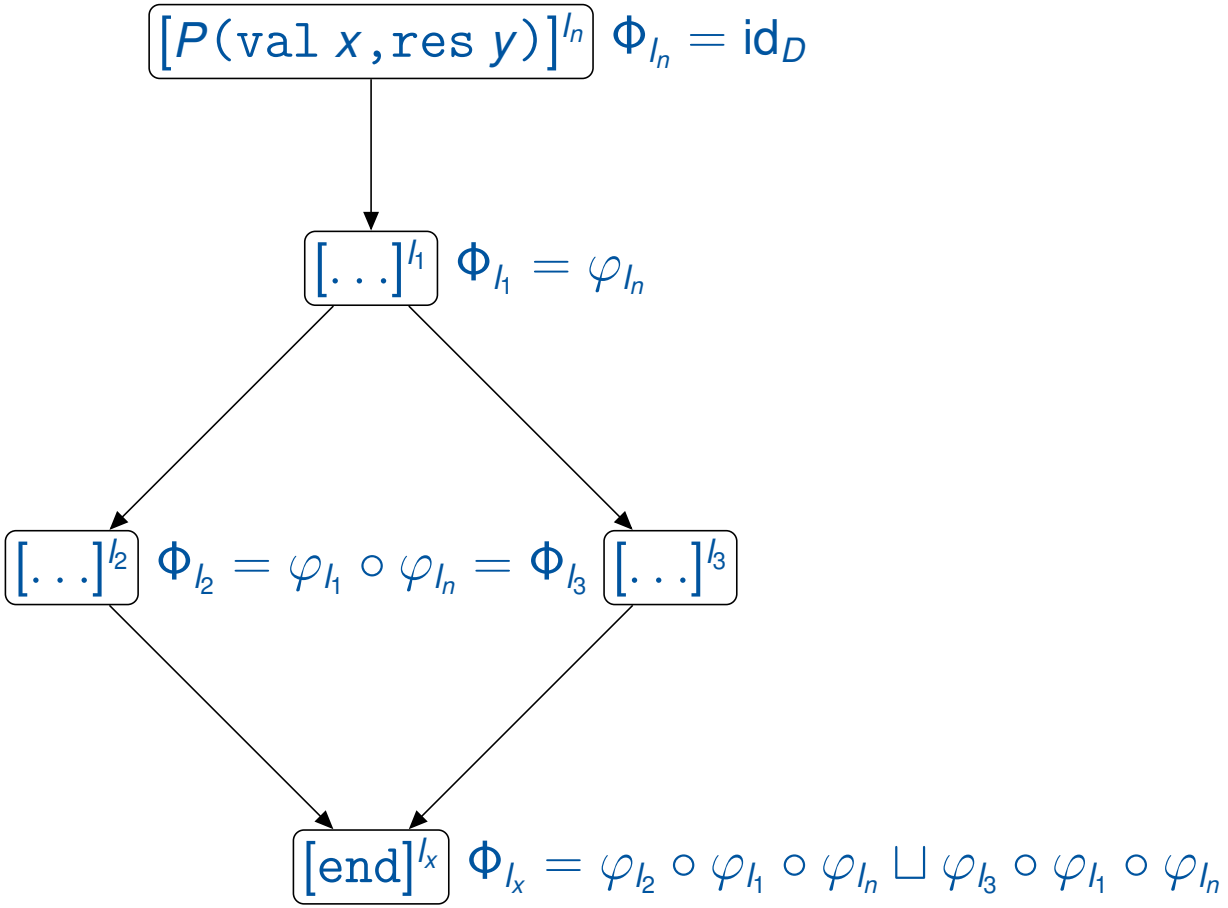
- Node transfer functions φ_l:
  - for non-procedural constructs (`skip`, assignments, tests): as before (specific to analysis)
  - for procedure call (l_c): parameter passing (specific to analysis)
  - for procedure entry (l_n): initialisation (specific to analysis)
  - for procedure exit (l_x): always identity
  - for procedure return (l_r): combine call context with procedure result obtained from summary function

$$\varphi_{l_r}(d) = \text{combine}(d, \Phi_{l_x}(\varphi_{l_c}(d)))$$

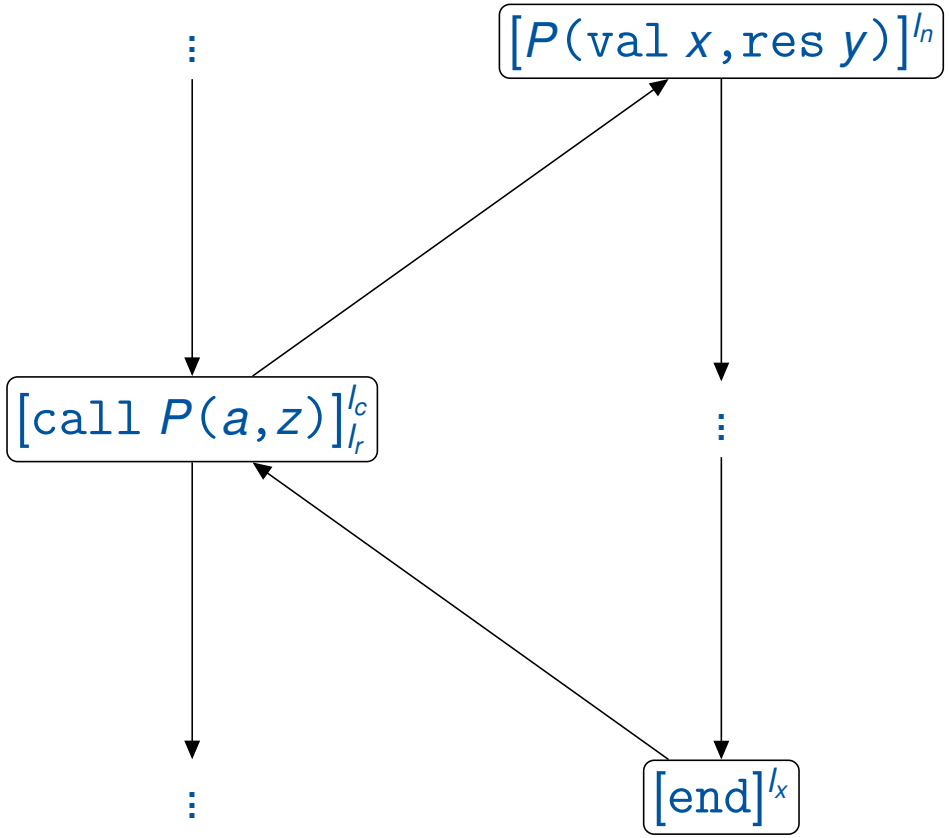
  (combine specific to analysis)
- Procedure summary functions Φ_l:
  - for procedure entry (l_n): identity
  - for procedure exit (l_x): full effect
  - inductively defined by stepwise composition of node transfer functions

The Interprocedural Fixpoint Solution

Procedure Summaries

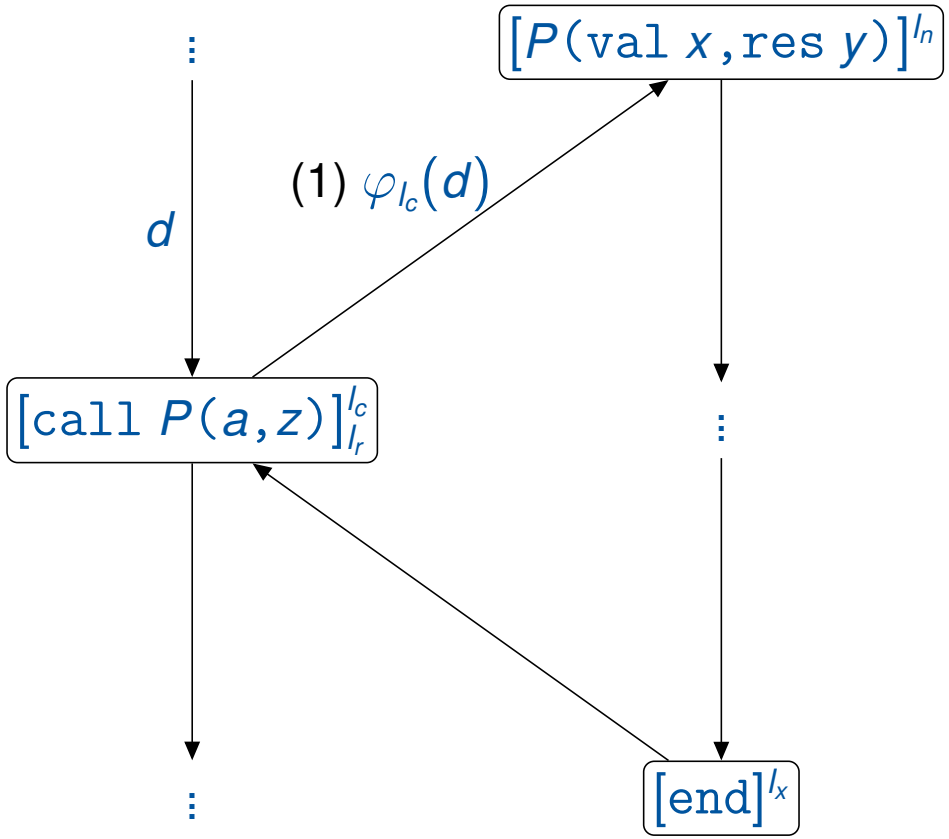


Information Flow



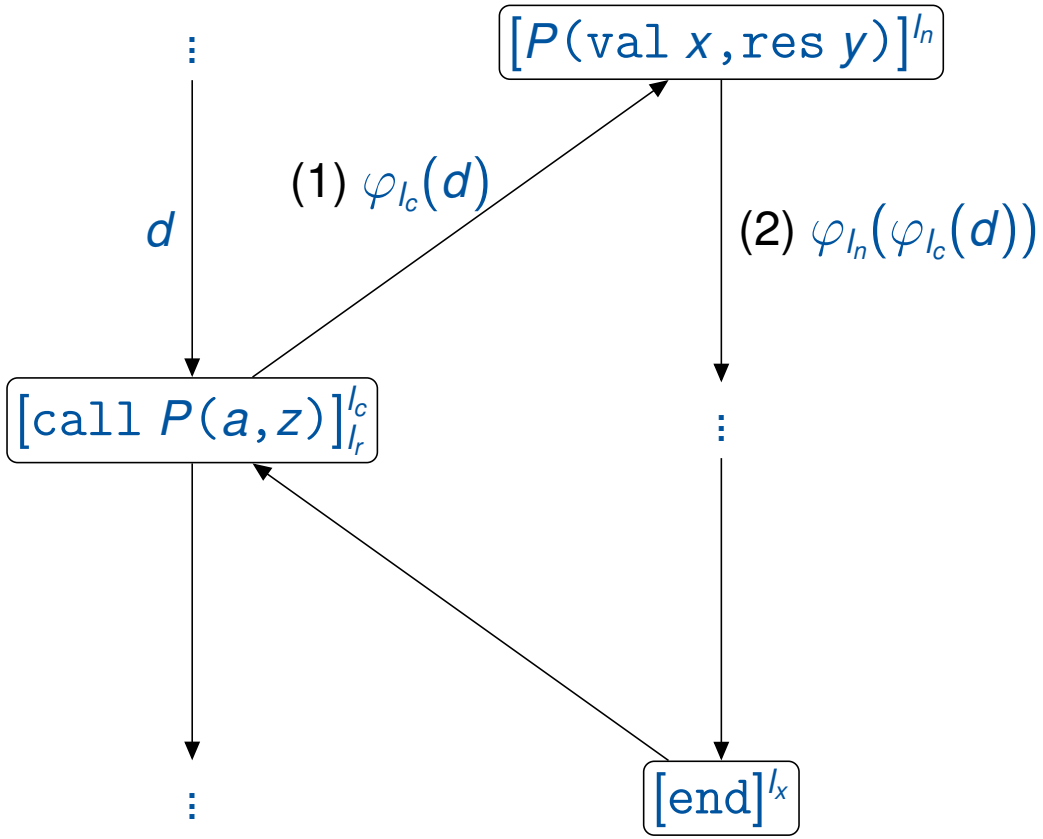
Information Flow

1. Parameter passing (φ_{l_c})



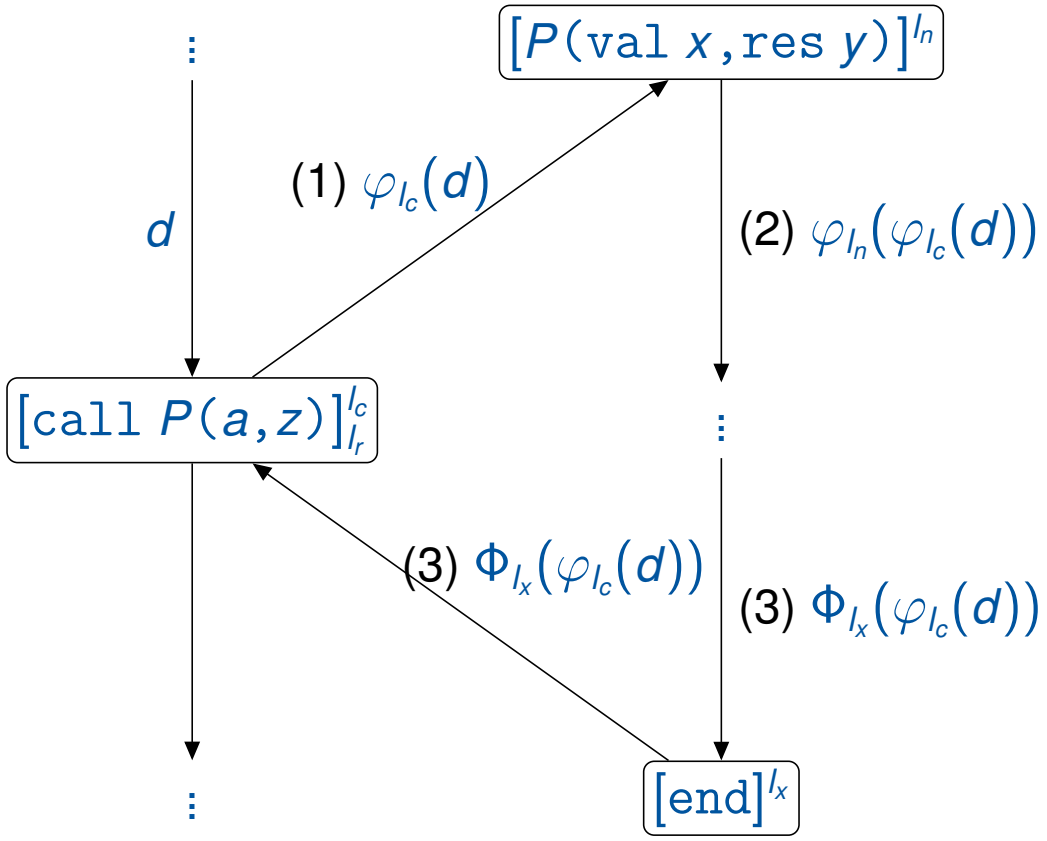
Information Flow

1. Parameter passing (φ_{l_c})
2. Procedure entry (φ_{l_n})



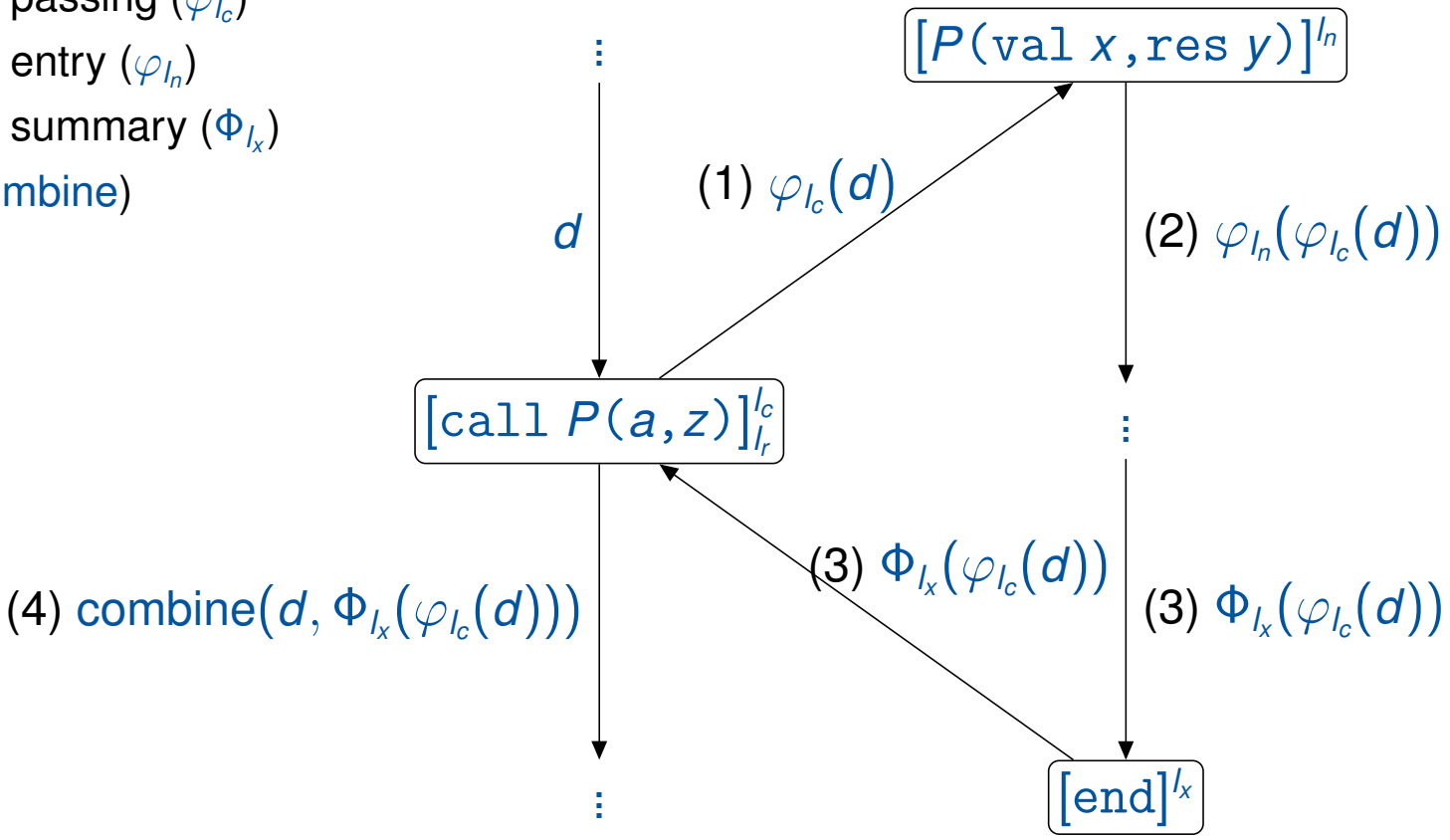
Information Flow

1. Parameter passing (φ_{l_c})
2. Procedure entry (φ_{l_n})
3. Procedure summary (Φ_{l_x})



Information Flow

1. Parameter passing (φ_{l_c})
2. Procedure entry (φ_{l_n})
3. Procedure summary (Φ_{l_x})
4. Return (combine)



Example: Constant Propagation

Example 18.1 (Constant Propagation; cf. Lecture 5/6)

$\hat{S} := (Lab, E, F, (D, \sqsubseteq), \iota, \varphi, \Phi)$ is determined by

- $D := \{\delta \mid \delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}\}$ (constant/undefined/overdefined)
- $\bot \sqsubseteq z \sqsubseteq \top$ for every $z \in \mathbb{Z}$
- $\iota := \delta_\top \in D$
- For each $l \in Lab \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in \text{iflow}\}$,

$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \texttt{skip} \text{ or } B^l \in BExp \\ \delta[x \mapsto val_\delta(a)] & \text{if } B^l = (x := a) \end{cases}$$

Example: Constant Propagation

Example 18.1 (Constant Propagation; cf. Lecture 5/6)

$\hat{S} := (Lab, E, F, (D, \sqsubseteq), \iota, \varphi, \Phi)$ is determined by

- $D := \{\delta \mid \delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}\}$ (constant/undefined/overdefined)
- $\bot \sqsubseteq z \sqsubseteq \top$ for every $z \in \mathbb{Z}$
- $\iota := \delta_\top \in D$
- For each $l \in Lab \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in \text{iflow}\}$,

$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \texttt{skip} \text{ or } B^l \in BExp \\ \delta[x \mapsto val_\delta(a)] & \text{if } B^l = (x := a) \end{cases}$$

- Whenever $p\,c$ contains $[\texttt{call } P(a,z)]^{l_c}_{l_r}$ and $\texttt{proc } [P(\texttt{val } x, \texttt{res } y)]^{l_n} \texttt{ is } c\ [\texttt{end}]^{l_x}$:
  - call/entry: set input and reset output parameter

$$\varphi_{l_c}(\delta) := \delta[x \mapsto val_\delta(a), y \mapsto \top], \quad \varphi_{l_n}(\delta) := \delta$$

  - return: reset parameters (x, y possibly used in calling context) and set return value

$$\varphi_{l_r}(\delta) := \text{combine}(\delta, \Phi_{l_x}(\varphi_{l_c}(\delta))), \quad \text{combine}(\delta, \delta') := \delta'[x \mapsto \delta(x), y \mapsto \delta(y), z \mapsto \delta'(y)]$$

(note: order of δ' updates important in case $z \in \{x, y\}$; definition of Φ: later)

Outline of Lecture 18

Types of Equations

For an interprocedural dataflow system $\hat{S} := (Lab, E, F, (D, \sqsubseteq), \iota, \varphi, \Phi)$ the intraprocedural equation system (cf. Definition 4.3)

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{\varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

is extended to a system with three kinds of equations (for $l \in Lab$):

Types of Equations

For an interprocedural dataflow system $\hat{S} := (Lab, E, F, (D, \sqsubseteq), \iota, \varphi, \Phi)$ the intraprocedural equation system (cf. Definition 4.3)

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{\varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

is extended to a system with three kinds of equations (for $l \in Lab$):

- for actual dataflow information: $\mathsf{AI}_l \in D$
  - counterpart of intraprocedural AI

Types of Equations

For an interprocedural dataflow system $\hat{S} := (Lab, E, F, (D, \sqsubseteq), \iota, \varphi, \Phi)$ the intraprocedural equation system (cf. Definition 4.3)

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{\varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

is extended to a system with three kinds of equations (for $l \in Lab$):

- for actual dataflow information: $\mathsf{AI}_l \in D$
  - counterpart of intraprocedural AI
- for node transfer functions: $\varphi_l : D \to D$
  - extension of intraprocedural transfer functions by handling of procedure calls

Types of Equations

For an interprocedural dataflow system $\hat{S} := (Lab, E, F, (D, \sqsubseteq), \iota, \varphi, \Phi)$ the intraprocedural equation system (cf. Definition 4.3)

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{\varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

is extended to a system with three kinds of equations (for $l \in Lab$):

- for actual dataflow information: $\mathsf{AI}_l \in D$
  - counterpart of intraprocedural AI
- for node transfer functions: $\varphi_l : D \to D$
  - extension of intraprocedural transfer functions by handling of procedure calls
- for procedure summary functions of complete procedures: $\Phi_l : D \to D$
  - $\Phi_l(d)$ yields information at (entry of) l if corresponding procedure is called with information d
  - thus $\Phi_{l_n}(d) = d$ and complete procedure effect represented by $\Phi_{l_x}(d)$

The Equation System

Formal Definition of Equation System

Dataflow equations (for each $l \in Lab$):

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \mathsf{AI}_{l_c} & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \bigsqcup \{ \varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F \text{ or } (l'; l) \in F \} & \text{otherwise} \end{cases}$$

Formal Definition of Equation System

Dataflow equations (for each $l \in Lab$):

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \mathsf{AI}_{l_c} & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \bigsqcup \{ \varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F \text{ or } (l'; l) \in F \} & \text{otherwise} \end{cases}$$

Node transfer functions (for each $l \in Lab \setminus \{ l_x \mid (l_c, l_n, l_x, l_r) \in \text{iflow} \}$):

$$\varphi_l(d) = \begin{cases} \text{combine}(d, \Phi_{l_x}(\varphi_{l_c}(d))) & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \text{specific to analysis} & \text{otherwise} \end{cases}$$

Formal Definition of Equation System

Dataflow equations (for each $l \in Lab$):

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \mathsf{AI}_{l_c} & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \bigsqcup \{ \varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F \text{ or } (l'; l) \in F \} & \text{otherwise} \end{cases}$$

Node transfer functions (for each $l \in Lab \setminus \{ l_x \mid (l_c, l_n, l_x, l_r) \in \text{iflow} \}$):

$$\varphi_l(d) = \begin{cases} \text{combine}(d, \Phi_{l_x}(\varphi_{l_c}(d))) & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \text{specific to analysis} & \text{otherwise} \end{cases}$$

Procedure summary functions (for each $l \in Lab$ occurring in some procedure):

$$\Phi_l(d) = \begin{cases} d & \text{if } l = l_n \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \Phi_{l_c}(d) & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \bigsqcup \{ \varphi_{l'}(\Phi_{l'}(d)) \mid (l', l) \in F \} & \text{otherwise} \end{cases}$$

Formal Definition of Equation System

Dataflow equations (for each $l \in Lab$):

$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \mathsf{AI}_{l_c} & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \bigsqcup \{ \varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F \text{ or } (l'; l) \in F \} & \text{otherwise} \end{cases}$$

Node transfer functions (for each $l \in Lab \setminus \{ l_x \mid (l_c, l_n, l_x, l_r) \in \text{iflow} \}$):

$$\varphi_l(d) = \begin{cases} \text{combine}(d, \Phi_{l_x}(\varphi_{l_c}(d))) & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \text{specific to analysis} & \text{otherwise} \end{cases}$$

Procedure summary functions (for each $l \in Lab$ occurring in some procedure):

$$\Phi_l(d) = \begin{cases} d & \text{if } l = l_n \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \Phi_{l_c}(d) & \text{if } l = l_r \text{ for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \bigsqcup \{ \varphi_{l'}(\Phi_{l'}(d)) \mid (l', l) \in F \} & \text{otherwise} \end{cases}$$

Note: introduces recursive equations for AI_l (of type D) and Φ_l (of type $D \to D$)

Solving the Equation System

- Monotonicity of φ_l implies monotonicity of $\Phi_{l'}$ ($\Phi_{l'} : D \to_{\text{mon}} D$)

Solving the Equation System

- Monotonicity of φ_l implies monotonicity of $\Phi_{l'}$ ($\Phi_{l'} : D \to_{\text{mon}} D$)
- Complete lattice property of $(D, \sqsubseteq)$ ensures that $(D \to_{\text{mon}} D, \dot{\sqsubseteq})$ is complete lattice
  - $\Phi_1 \dot{\sqsubseteq} \Phi_2$ iff $\forall d \in D : \Phi_1(d) \sqsubseteq \Phi_2(d)$ (pointwise extension)

Solving the Equation System

- Monotonicity of φ_l implies monotonicity of $\Phi_{l'}$ ($\Phi_{l'} : D \to_{\text{mon}} D$)
- Complete lattice property of $(D, \sqsubseteq)$ ensures that $(D \to_{\text{mon}} D, \dot{\sqsubseteq})$ is complete lattice
  - $\Phi_1 \dot{\sqsubseteq} \Phi_2$ iff $\forall d \in D : \Phi_1(d) \sqsubseteq \Phi_2(d)$ (pointwise extension)
- Equation system induces monotonic functional on complete lattice

$$\Psi_{\hat{S}} : \underbrace{D^n}_{\text{AI}} \times \underbrace{(D \to_{\text{mon}} D)^m}_{\Phi} \to D^n \times (D \to_{\text{mon}} D)^m$$

where $n = |Lab|$ and $m \leq n$ (procedure labels)

Solving the Equation System

- Monotonicity of φ_l implies monotonicity of $\Phi_{l'}$ ($\Phi_{l'} : D \to_{\text{mon}} D$)
- Complete lattice property of $(D, \sqsubseteq)$ ensures that $(D \to_{\text{mon}} D, \dot{\sqsubseteq})$ is complete lattice
  - $\Phi_1 \dot{\sqsubseteq} \Phi_2$ iff $\forall d \in D : \Phi_1(d) \sqsubseteq \Phi_2(d)$ (pointwise extension)
- Equation system induces monotonic functional on complete lattice

$$\Psi_{\hat{S}} : \underbrace{D^n}_{\text{AI}} \times \underbrace{(D \to_{\text{mon}} D)^m}_{\Phi} \to D^n \times (D \to_{\text{mon}} D)^m$$

  where $n = |Lab|$ and $m \leq n$ (procedure labels)
- Thus least solution effectively computable by fixpoint iteration (Theorem 3.15):

$$\text{fix}(\Psi_{\hat{S}}) = \bigsqcup \{\Psi^k_{\hat{S}}(\bot) \mid k \in \mathbb{N}\} \in D^n \times (D \to_{\text{mon}} D)^m$$

  where $\bot = (\bot_D^n, [d \mapsto \bot_D \mid d \in D]^m)$

Solving the Equation System

- Monotonicity of φ_l implies monotonicity of $\Phi_{l'}$ ($\Phi_{l'} : D \to_{\text{mon}} D$)
- Complete lattice property of $(D, \sqsubseteq)$ ensures that $(D \to_{\text{mon}} D, \dot{\sqsubseteq})$ is complete lattice
  - $\Phi_1 \dot{\sqsubseteq} \Phi_2$ iff $\forall d \in D : \Phi_1(d) \sqsubseteq \Phi_2(d)$ (pointwise extension)
- Equation system induces monotonic functional on complete lattice

$$\Psi_{\hat{S}} : \underbrace{D^n}_{\text{AI}} \times \underbrace{(D \to_{\text{mon}} D)^m}_{\Phi} \to D^n \times (D \to_{\text{mon}} D)^m$$

  where $n = |Lab|$ and $m \leq n$ (procedure labels)
- Thus least solution effectively computable by fixpoint iteration (Theorem 3.15):

$$\text{fix}(\Psi_{\hat{S}}) = \bigsqcup \{\Psi^k_{\hat{S}}(\bot) \mid k \in \mathbb{N}\} \in D^n \times (D \to_{\text{mon}} D)^m$$

  where $\bot = (\bot_D^n, [d \mapsto \bot_D \mid d \in D]^m)$
- Moreover:
  - $(D \to_{\text{mon}} D, \dot{\sqsubseteq})$ satisfies ACC if $(D, \sqsubseteq)$ satisfies ACC
  - $\Phi_{l'}$ distributive if φ_l distributive

Solving the Equation System

- Monotonicity of φ_l implies monotonicity of $\Phi_{l'}$ ($\Phi_{l'} : D \to_{\text{mon}} D$)
- Complete lattice property of $(D, \sqsubseteq)$ ensures that $(D \to_{\text{mon}} D, \dot{\sqsubseteq})$ is complete lattice
  - $\Phi_1 \dot{\sqsubseteq} \Phi_2$ iff $\forall d \in D : \Phi_1(d) \sqsubseteq \Phi_2(d)$ (pointwise extension)
- Equation system induces monotonic functional on complete lattice

$$\Psi_{\hat{S}} : \underbrace{D^n}_{\text{AI}} \times \underbrace{(D \to_{\text{mon}} D)^m}_{\Phi} \to D^n \times (D \to_{\text{mon}} D)^m$$

  where $n = |Lab|$ and $m \leq n$ (procedure labels)
- Thus least solution effectively computable by fixpoint iteration (Theorem 3.15):

$$\text{fix}(\Psi_{\hat{S}}) = \bigsqcup \{\Psi^k_{\hat{S}}(\bot) \mid k \in \mathbb{N}\} \in D^n \times (D \to_{\text{mon}} D)^m$$

  where $\bot = (\bot_D^n, [d \mapsto \bot_D \mid d \in D]^m)$
- Moreover:
  - $(D \to_{\text{mon}} D, \dot{\sqsubseteq})$ satisfies ACC if $(D, \sqsubseteq)$ satisfies ACC
  - $\Phi_{l'}$ distributive if φ_l distributive
- **Problem:** effective and efficient representation of procedure summaries
  - D finite $\implies$ $D \to_{\text{mon}} D$ finite $\implies$ can use (compact representation of) value tables
  - often: demand-driven approach (restrict analysis to values occurring in calls)
  - example: Constant Propagation (see following slide)

Example of Equation System

Example 18.2 (Constant Propagation)

Program:

```
proc [P(val x, res y)]^1 is
  [y := 2*(x-1)]^2;
[end]^3;
[call P(2, z)]^4_5;
[call P(z, z)]^6_7;
[skip]^8
```

Example of Equation System

Example 18.2 (Constant Propagation)

Program:

```
proc [P(val x, res y)]^1 is
  [y := 2*(x-1)]^2;
[end]^3;
[call P(2, z)]^4_5;
[call P(z, z)]^6_7;
[skip]^8
```

Dataflow equations:

$$
\begin{aligned}
AI_1 &= \varphi_4(AI_4) \sqcup \varphi_6(AI_6) \\
AI_2 &= \varphi_1(AI_1) \\
AI_3 &= \varphi_2(AI_2) \\
AI_4 &= \iota = [x, y, z \mapsto \top] \\
AI_5 &= AI_4 \\
AI_6 &= \varphi_5(AI_5) \\
AI_7 &= AI_6 \\
AI_8 &= \varphi_7(AI_7)
\end{aligned}
$$

Example of Equation System

Example 18.2 (Constant Propagation)

Program:

```
proc [P(val x, res y)]^1 is
  [y := 2*(x-1)]^2;
[end]^3;
[call P(2, z)]^4_5;
[call P(z, z)]^6_7;
[skip]^8
```

Dataflow equations:

$$
\begin{aligned}
AI_1 &= \varphi_4(AI_4) \sqcup \varphi_6(AI_6)\\
AI_2 &= \varphi_1(AI_1)\\
AI_3 &= \varphi_2(AI_2)\\
AI_4 &= \iota = [x, y, z \mapsto \top]\\
AI_5 &= AI_4\\
AI_6 &= \varphi_5(AI_5)\\
AI_7 &= AI_6\\
AI_8 &= \varphi_7(AI_7)
\end{aligned}
$$

Node transfer functions:

$$
\begin{aligned}
\varphi_1(\delta) &= \delta\\
\varphi_2(\delta) &= \delta[y \mapsto \mathit{val}_\delta(2*(x-1))]\\
\varphi_4(\delta) &= \delta[x \mapsto 2, y \mapsto \top]\\
\varphi_5(\delta) &= \mathrm{combine}(\delta, \Phi_3(\varphi_4(\delta)))\\
\varphi_6(\delta) &= \delta[x \mapsto \delta(z), y \mapsto \top]\\
\varphi_7(\delta) &= \mathrm{combine}(\delta, \Phi_3(\varphi_6(\delta)))\\
\mathrm{combine}(\delta, \delta') &= \delta'[x \mapsto \delta(x), y \mapsto \delta(y), z \mapsto \delta'(y)]
\end{aligned}
$$

Example of Equation System

Example 18.2 (Constant Propagation)

Program:

```
proc [P(val x, res y)]^1 is
  [y := 2*(x-1)]^2;
[end]^3;
[call P(2, z)]^4_5;
[call P(z, z)]^6_7;
[skip]^8
```

Dataflow equations:

$$
\begin{aligned}
AI_1 &= \varphi_4(AI_4) \sqcup \varphi_6(AI_6)\\
AI_2 &= \varphi_1(AI_1)\\
AI_3 &= \varphi_2(AI_2)\\
AI_4 &= \iota = [x, y, z \mapsto \top]\\
AI_5 &= AI_4\\
AI_6 &= \varphi_5(AI_5)\\
AI_7 &= AI_6\\
AI_8 &= \varphi_7(AI_7)
\end{aligned}
$$

Node transfer functions:

$$
\begin{aligned}
\varphi_1(\delta) &= \delta\\
\varphi_2(\delta) &= \delta[y \mapsto \mathit{val}_\delta(2*(x-1))]\\
\varphi_4(\delta) &= \delta[x \mapsto 2, y \mapsto \top]\\
\varphi_5(\delta) &= \mathrm{combine}(\delta, \Phi_3(\varphi_4(\delta)))\\
\varphi_6(\delta) &= \delta[x \mapsto \delta(z), y \mapsto \top]\\
\varphi_7(\delta) &= \mathrm{combine}(\delta, \Phi_3(\varphi_6(\delta)))\\
\mathrm{combine}(\delta, \delta') &= \delta'[x \mapsto \delta(x), y \mapsto \delta(y), z \mapsto \delta'(y)]
\end{aligned}
$$

Procedure summary functions:

$$
\begin{aligned}
\Phi_1(\delta) &= \delta\\
\Phi_2(\delta) &= \varphi_1(\Phi_1(\delta)) = \delta\\
\Phi_3(\delta) &= \varphi_2(\Phi_2(\delta)) = \delta[y \mapsto \mathit{val}_\delta(2*(x-1))]
\end{aligned}
$$

Example of Equation System

Example 18.2 (Constant Propagation)

Program:

```
proc [P(val x, res y)]^1 is
  [y := 2*(x-1)]^2;
[end]^3;
[call P(2, z)]^4_5;
[call P(z, z)]^6_7;
[skip]^8
```

Dataflow equations:

$$\begin{aligned}
AI_1 &= \varphi_4(AI_4) \sqcup \varphi_6(AI_6)\\
AI_2 &= \varphi_1(AI_1)\\
AI_3 &= \varphi_2(AI_2)\\
AI_4 &= \iota = [x, y, z \mapsto \top]\\
AI_5 &= AI_4\\
AI_6 &= \varphi_5(AI_5)\\
AI_7 &= AI_6\\
AI_8 &= \varphi_7(AI_7)
\end{aligned}$$

Node transfer functions:

$$\begin{aligned}
\varphi_1(\delta) &= \delta\\
\varphi_2(\delta) &= \delta[y \mapsto \mathit{val}_\delta(2*(x-1))]\\
\varphi_4(\delta) &= \delta[x \mapsto 2, y \mapsto \top]\\
\varphi_5(\delta) &= \mathrm{combine}(\delta, \Phi_3(\varphi_4(\delta)))\\
\varphi_6(\delta) &= \delta[x \mapsto \delta(z), y \mapsto \top]\\
\varphi_7(\delta) &= \mathrm{combine}(\delta, \Phi_3(\varphi_6(\delta)))\\
\mathrm{combine}(\delta, \delta') &= \delta'[x \mapsto \delta(x), y \mapsto \delta(y), z \mapsto \delta'(y)]
\end{aligned}$$

Procedure summary functions:

$$\begin{aligned}
\Phi_1(\delta) &= \delta\\
\Phi_2(\delta) &= \varphi_1(\Phi_1(\delta)) = \delta\\
\Phi_3(\delta) &= \varphi_2(\Phi_2(\delta)) = \delta[y \mapsto \mathit{val}_\delta(2*(x-1))]
\end{aligned}$$

Fixpoint iteration: on the board

Outline of Lecture 18

Correctness of Fixpoint Solution

Soundness and Completeness

The following results carry over from the intraprocedural case:

Theorem 18.3

Let $S := (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ *be a dataflow system with interprocedural extension* $\hat{S} := (Lab, E, F, (D, \sqsubseteq), \iota, \varphi, \Phi)$ *with MVP solution* $\mathrm{mvp}(S)$ *(Def. 17.9) and fixpoint solution* $\mathrm{fix}(\Psi_{\hat{S}}) = (d_1, \ldots, d_n, f_1, \ldots, f_m)$*. Then:*

1. *(cf. Theorem 6.2)*

$$\mathrm{mvp}(S) \sqsubseteq^n (d_1, \ldots, d_n)$$

2. *(cf. Theorem 6.5)*

$$\mathrm{mvp}(S) = (d_1, \ldots, d_n) \text{ if all } \varphi_l \text{ are distributive}$$

Proof.

see J. Knoop, B. Steffen: *The Interprocedural Coincidence Theorem*, Proc. CC '92, LNCS 641, Springer, 1992, 125–140[1] □

[1] https://link.springer.com/chapter/10.1007/3-540-55984-1_13