



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/>

Recap: Interprocedural Dataflow Analysis

Extending the Syntax

Syntactic categories:

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

Context-free grammar

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

- All labels and procedure names in **program** p c distinct
- In $\text{proc } [P(\text{val } x, \text{res } y)]^{l_n} \text{ is } c \text{ [end]}^{l_x}$, l_n / l_x refers to the **entry** / **exit** of P
- In $[\text{call } P(a, x)]_{l_r}^{l_c}$, 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**
 - ⇒ 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 $[call\ P(a, x)]_{l_r}^{l_c}$:
transfer functions $\varphi_{l_c}, \varphi_{l_r} : D \rightarrow D$ (definition later)
- For each procedure declaration $proc\ [P(val\ x, res\ y)]^{l_n}\ is\ c\ [end]^{l_x}$:
transfer functions $\varphi_{l_n}, \varphi_{l_x} : D \rightarrow D$ (definition later)
- Induces **equation system**

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

- **Problem:** procedure calls $(l_c; l_n)$ and procedure returns $(l_x; l_r)$ treated like goto's
 - ⇒ **nesting** of calls and returns ignored
 - ⇒ **too many paths** considered
 - ⇒ 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:

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

Recap: Interprocedural Dataflow Analysis

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

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

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

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

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

Recap: Interprocedural Dataflow Analysis

The MVP Solution II

Definition (MVP solution)

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

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

where, for every $l \in Lab$,

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

Corollary

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

Proof.

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

The Interprocedural Fixpoint Solution

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)

⇒ Two types of functions:

- **Node transfer functions** φ_I :
 - for non-procedural constructs (**skip**, assignments, tests): as before (specific to analysis)
 - for procedure call (I_c): parameter passing (specific to analysis)
 - for procedure entry (I_n): initialisation (specific to analysis)
 - for procedure exit (I_x): always identity
 - for procedure return (I_r): combine call context with procedure result obtained from summary function

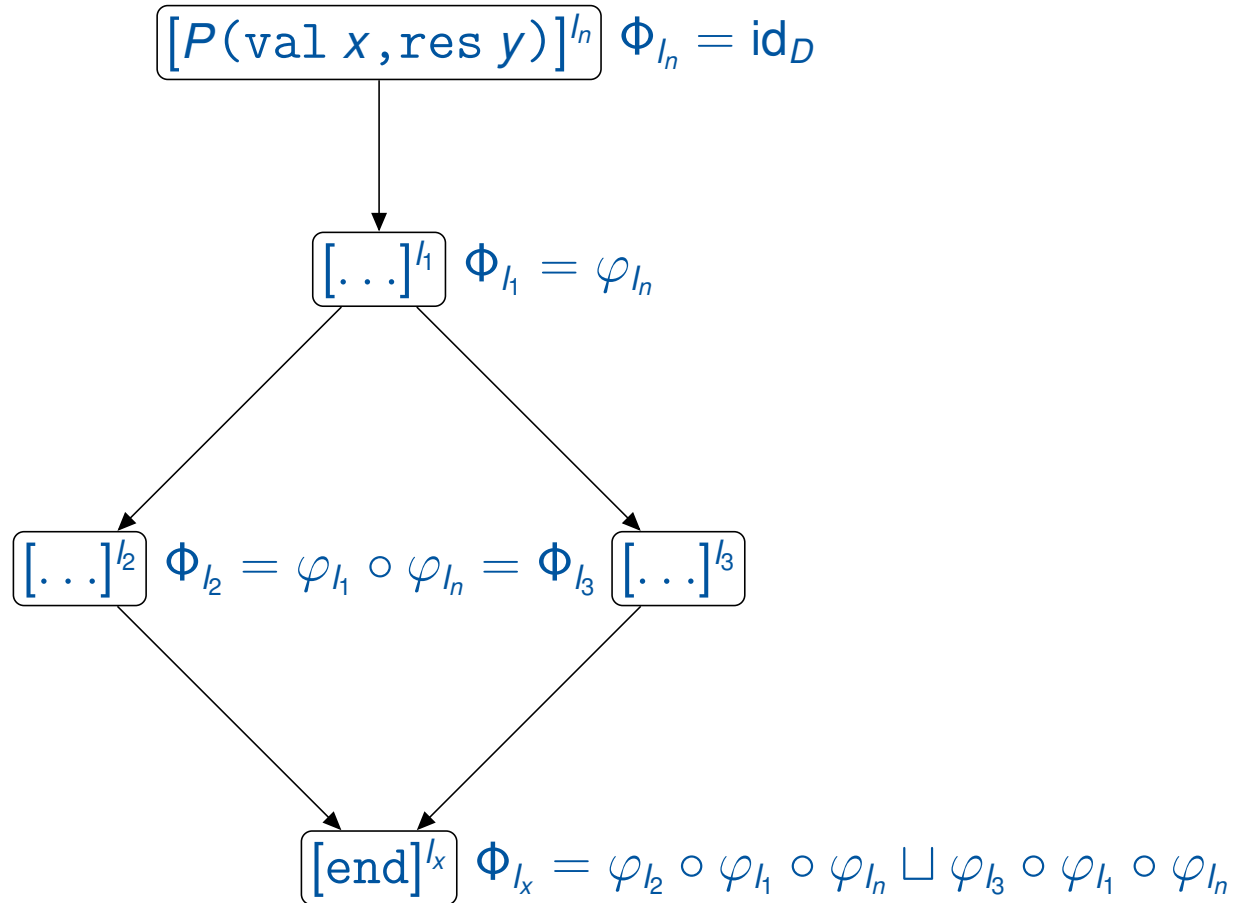
$$\varphi_{I_r}(d) = \text{combine}(d, \Phi_{I_x}(\varphi_{I_c}(d)))$$

(**combine** specific to analysis)

- **Procedure summary functions** Φ_I :
 - for procedure entry (I_n): identity
 - for procedure exit (I_x): full effect
 - inductively defined by stepwise composition of node transfer functions

The Interprocedural Fixpoint Solution

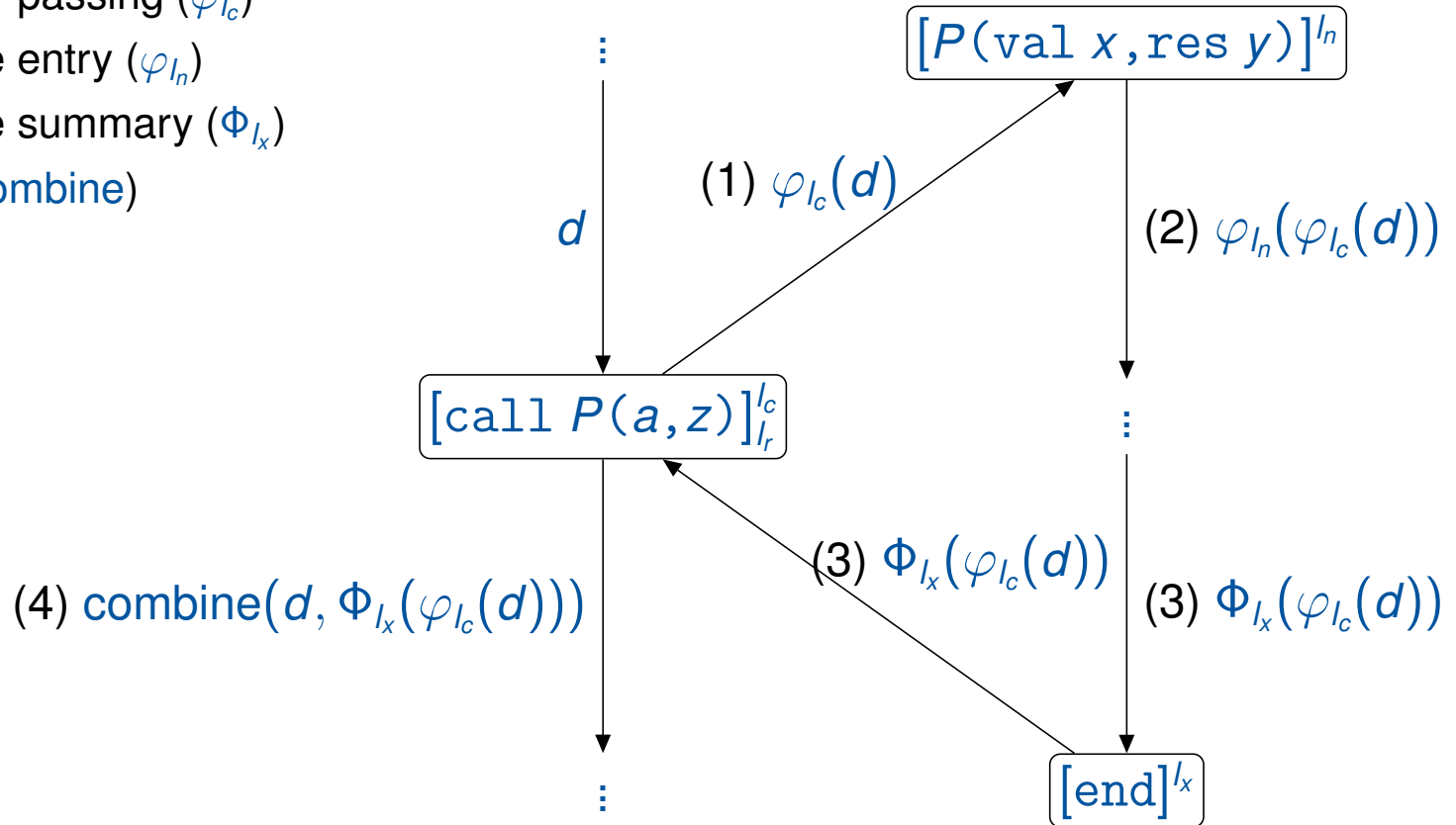
Procedure Summaries



The Interprocedural Fixpoint Solution

Information Flow

1. Parameter passing (φ_{I_c})
2. Procedure entry (φ_{I_n})
3. Procedure summary (Φ_{I_x})
4. Return (**combine**)



The Interprocedural Fixpoint Solution

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

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

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

- Whenever pc contains $[call\ P(a, z)]_{l_r}^{l_c}$ and $proc\ [P(val\ x, res\ y)]_{l_n}^{l_x}$ is $c\ [end]_{l_x}^{l_c}$:
 - **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

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

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

The Equation System

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)

$$AI_I = \begin{cases} \iota & \text{if } I \in E \\ \bigsqcup \{ \varphi_{I'}(AI_{I'}) \mid (I', I) \in F \} & \text{otherwise} \end{cases}$$

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

- for actual **dataflow information**: $AI_I \in D$
 - counterpart of intraprocedural AI
- for **node transfer functions**: $\varphi_I : D \rightarrow D$
 - extension of intraprocedural transfer functions by handling of procedure calls
- for **procedure summary functions of complete procedures**: $\Phi_I : D \rightarrow D$
 - $\Phi_I(d)$ yields information at (entry of) I if corresponding procedure is called with information d
 - thus $\Phi_{I_n}(d) = d$ and complete procedure effect represented by $\Phi_{I_x}(d)$

The Equation System

Formal Definition of Equation System

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

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

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 $D \rightarrow D$)

The Equation System

Solving the Equation System

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

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

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

- Thus least solution effectively computable by **fixpoint iteration** (Theorem 3.15):

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

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

- Moreover:
 - $(D \rightarrow_{\text{mon}} D, \sqsubseteq)$ satisfies **ACC** if (D, \sqsubseteq) satisfies ACC
 - Φ_l **distributive** if φ_l distributive
- **Problem: effective and efficient representation** of procedure summaries
 - D finite $\implies D \rightarrow_{\text{mon}} D$ 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)

The Equation System

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;
[call P(z, z)]6;
[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:

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

Procedure summary functions:

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

Fixpoint iteration: on the board

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 $\text{mvp}(S)$ (Def. 17.9) and fixpoint solution $\text{fix}(\Psi_{\hat{S}}) = (d_1, \dots, d_n, f_1, \dots, f_m)$. Then:

1. (cf. Theorem 6.2)

$$\text{mvp}(S) \sqsubseteq^n (d_1, \dots, d_n)$$

2. (cf. Theorem 6.5)

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

Proof.

see J. Knoop, B. Steffen: *The Interprocedural Coincidence Theorem*, Proc. CC '92, LNCS 641, Springer, 1992, 125–140¹ □

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