

Semantics and Verification of Software

Lecture 22: Dataflow Analysis IX (Interprocedural Fixpoint Solution)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

<http://www-i2.informatik.rwth-aachen.de/i2/svsw08/>

Winter semester 2008/09

- 1 Repetition: Interprocedural Dataflow Analysis
- 2 The Interprocedural Fixpoint Solution
- 3 The Equation System

Extending the Syntax

Syntactic categories:

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

Context-free grammar:

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

- All labels and procedure names in $program$ pc 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 , second call-by-result

Procedure Flow Graphs

Definition (Procedure flow graphs)

The auxiliary functions `init`, `final`, and `flow` are extended as follows:

$$\text{init}([\text{call } P(a, x)]_{l_r}^{l_c}) := l_c$$

$$\text{final}([\text{call } P(a, x)]_{l_r}^{l_c}) := \{l_r\}$$

$$\text{flow}([\text{call } P(a, x)]_{l_r}^{l_c}) := \{(l_c; l_n), (l_x; l_r)\}$$

$$\text{init}(\text{proc } [P(\text{val } x, \text{res } y)]_{l_n}^{l_x} \text{ is } c [\text{end}]^{l_x}) := l_n$$

$$\text{final}(\text{proc } [P(\text{val } x, \text{res } y)]_{l_n}^{l_x} \text{ is } c [\text{end}]^{l_x}) := \{l_x\}$$

$$\text{flow}(\text{proc } [P(\text{val } x, \text{res } y)]_{l_n}^{l_x} \text{ is } c [\text{end}]^{l_x}) := \{(l_n, \text{init}(c))\} \cup \{(l, l_x) \mid l \in \text{final}(c)\}$$

if `proc` $[P(\text{val } x, \text{res } y)]_{l_n}^{l_x}$ `is` c `end` l_x is in p .

Moreover the **interprocedural flow** of a program pc is defined by

$$IF := \{(l_c, l_n, l_x, l_r) \mid pc \text{ contains } [\text{call } P(a, x)]_{l_r}^{l_c} \text{ and } \text{proc } [P(\text{val } x, \text{res } y)]_{l_n}^{l_x} \text{ is } c [\text{end}]^{l_x}\} \subseteq L^4$$

Example (Fibonacci numbers)

```
proc [Fib(val x, y, res z)]1 is
  if [x<2]2 then
    [z := y+1]3
  else
    [call Fib(x-1, y, z)]4;
    [call Fib(x-2, z, z)]6;
  [end]8;
  [call Fib(5, 0, v)]910
```

- “Valid” path:
[9, 1, 2, 3, 8, 10]
- “Invalid” path:
[9, 1, 2, 4, 1, 2, 3, 8, 10]

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

Definition (Valid paths I)

Given a dataflow system $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ and $l_1, l_2 \in L$, 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 productions:

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

Definition (Valid paths II)

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

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

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

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

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

Definition (MVP solution)

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

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

where, for every $l \in L$,

$$\mathbf{mvp}(l) := \bigsqcup \{\varphi_p(\iota) \mid p \in VPath(l)\}.$$

Corollary

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

Proof.

- ① since $VPath(l) \subseteq Path(l)$ for every $l \in L$
- ② by undecidability of MOP solution



- 1 Repetition: Interprocedural Dataflow Analysis
- 2 The Interprocedural Fixpoint Solution
- 3 The Equation System

- **Goal:** adapt fixpoint solution to **avoid invalid paths**
- **Approach:** encode call history into data flow properties (use **stacks** D^+ as dataflow version of runtime stack)
- Non-procedural constructs (**skip**, assignments, tests): operate only on **topmost element**
- **call:** computes **new topmost entry** from current and pushes it
- **return:** **removes topmost entry** and combines it with underlying entry

The Interprocedural Extension I

Definition 22.1 (Interprocedural extension (forward analysis))

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. The **interprocedural extension** of S is given by

$$\hat{S} := (L, E, F, (\hat{D}, \hat{\sqsubseteq}), \hat{\iota}, \hat{\varphi})$$

where

- $\hat{D} := D^+$
- $d_1 \dots d_n \hat{\sqsubseteq} d'_1 \dots d'_n$ iff $d_i \sqsubseteq d'_i$ for every $1 \leq i \leq n$
- $\hat{\iota} := \iota \in D^+$
- for each $l \in L \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in IF\}$,
 $\hat{\varphi}_l : D^+ \rightarrow D^+$ is given by $\hat{\varphi}_l(dw) := \varphi_l(d)w$
- for each $(l_c, l_n, l_x, l_r) \in IF$, $\hat{\varphi}_l : D^+ \rightarrow D^+$ is given by
 - $\hat{\varphi}_{l_c}(dw) := \varphi_{l_c}(d)dw$
 - $\hat{\varphi}_{l_n}(w) := w$
 - $\hat{\varphi}_{l_x}(w) := w$
 - $\hat{\varphi}_{l_r}(d'dw) := \varphi_{l_r}(d', d)w$

Remark: the schema

- ① $\hat{\varphi}_{l_c}(dw) := \varphi_{l_c}(d)dw$
- ② $\hat{\varphi}_{l_n}(w) := w$
- ③ $\hat{\varphi}_{l_x}(w) := w$
- ④ $\hat{\varphi}_{l_r}(d'dw) := \varphi_{l_r}(d', d)w$

can be generalized by allowing a modification of the topmost entry in
2. and 3. (local variables, ...)

Example 22.2 (Constant Propagation (cf. Lecture 19))

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

- $D := \{\delta \mid \delta : Var_c \rightarrow \mathbb{Z} \cup \{\perp, \top\}\}$
- $\perp \sqsubseteq z \sqsubseteq \top$
- $\iota := \delta_{\top} \in D$
- for each $l \in L \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in IF\}$,
$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \text{skip or } B^l \in BExp \\ \delta[x \mapsto \mathfrak{A}[a]\delta] & \text{if } B^l = (x := a) \end{cases}$$
- whenever pc contains $[\text{call } P(a, z)]_{l_r}^{l_c}$ and
 $\text{proc } [P(\text{val } x, \text{res } y)]^{l_n} \text{ is } c \text{ [end]}^{l_x}$,
 - **call:** set input parameter and reset output parameter
 $\varphi_{l_c}(\delta) := \delta[x \mapsto \mathfrak{A}[a]\delta, y \mapsto \top]$
 - **return:** propagate output parameter to caller by overwriting old value
 $\varphi_{l_r}(\delta', \delta) := \delta[z \mapsto \delta'(y)]$

- 1 Repetition: Interprocedural Dataflow Analysis
- 2 The Interprocedural Fixpoint Solution
- 3 The Equation System

The Equation System I

For an interprocedural dataflow system $\hat{S} := (L, E, F, (\hat{D}, \hat{\sqsubseteq}), \hat{\iota}, \hat{\varphi})$, the intraprocedural equation system

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

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

- for actual **dataflow information**: $\text{AI}_l \in D^+$
(extension of intraprocedural AI)
- for **transfer functions of single nodes**: $f_l : D^+ \rightarrow D^+$
(extension of intraprocedural transfer functions)
- for **transfer functions of complete procedures**: $F_l : D^+ \rightarrow D^+$
($F_l(w)$ yields information at l if surrounding procedure is called
with information $w \implies$ full procedure represented by F_{l_x})

The Equation System II

Formal definition:

$$\text{Al}_l = \begin{cases} \uplus \{\hat{\varphi}_{l_c}(\text{Al}_{l_c}) \mid (l_c, l_n, l_x, l_r) \in IF\} & \text{if } l \in E \\ \uplus \{\text{fl}'(\text{Al}_{l'}) \mid (l', l) \in F\} & \text{if } l = l_n \\ & \text{for some } (l_c, l_n, l_x, l_r) \in IF \\ & \text{otherwise} \end{cases}$$

(if l not an exit label)

$$\text{fl}(w) = \begin{cases} \hat{\varphi}_{l_r}(\text{Fl}_x(\hat{\varphi}_{l_c}(w))) & \text{if } l = l_c \text{ for some } (l_c, l_n, l_x, l_r) \in IF \\ \hat{\varphi}_l(w) & \text{otherwise} \end{cases}$$

$$\text{Fl}_l(w) = \begin{cases} w & \text{if } l = l_n \\ \uplus \{\text{fl}'(\text{Fl}'(w)) \mid (l', l) \in F\} & \text{for some } (l_c, l_n, l_x, l_r) \in IF \\ & \text{otherwise} \end{cases}$$

(if l occurs in procedure)

As before: induces monotonic functional on lattice with ACC
⇒ least fixpoint effectively computable

Example 22.3 (Constant Propagation)

```
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
```