Static Program Analysis

Lecture 19: Interprocedural Dataflow Analysis II (Fixpoint Solution)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

noll@cs.rwth-aachen.de

http://moves.rwth-aachen.de/teaching/ws-1415/spa/

Winter Semester 2014/15

Online Registration for Seminars and Practical Courses (Praktika) in Summer Term 2015

Who?

Students of:
- Master Courses
- Bachelor Informatik (~~Pro~~Seminar!)

Where?

www.graphics.rwth-aachen.de/apse

When?

14.01.2015 - 28.01.2015

Outline

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:

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

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

Naive Formulation I

- **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

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

- **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 imprecise (but still correct)

Naive Formulation II

Example (Impreciseness of constant propagation analysis)

```
proc [P(val x, res y)]^1 is
  [y := x]^2
[end]^3;
if [y=0]^4 then
  [call P(1, y)]^5_6;
  [y := y-1]^7
else
  [call P(2, y)]^8_9;
  [y := y-2]^10;
[skip]^11
```

Two "valid" and two "invalid" paths:

- Valid: $[4, 5, 1, 2, 3, 6, 7, 11]$
  $\Longrightarrow$ $y = 0$ at label 11
- Valid: $[4, 8, 1, 2, 3, 9, 10, 11]$
  $\Longrightarrow$ $y = 0$ at label 11
- Invalid: $[4, 5, 1, 2, 3, 9, 10, 11]$
  $\Longrightarrow$ $y = -1$ at label 11
- Invalid: $[4, 8, 1, 2, 3, 6, 7, 11]$
  $\Longrightarrow$ $y = 1$ at label 11

$\Longrightarrow$ actually always $y = 0$ at 11, but naive method yields $y = \top$

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

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

For a path $\pi = [l_1, \ldots, l_{k-1}] \in VPath(l)$, 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. since $\mathrm{mvp}(S) = \mathrm{mop}(S)$ in intraprocedural case, and by undecidability of MOP solution (cf. Theorem 7.4) □

Outline

Making Context Explicit

- **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 (= call-site) entry

The Interprocedural Extension I

Definition 19.1 (Interprocedural extension (forward analysis))

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system where $\varphi_{l_r} : D^2 \to D$ for each $(l_c, l_n, l_x, l_r) \in \text{iflow}$ (and $\varphi_l : D \to D$ otherwise).
The interprocedural extension of S is given by

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

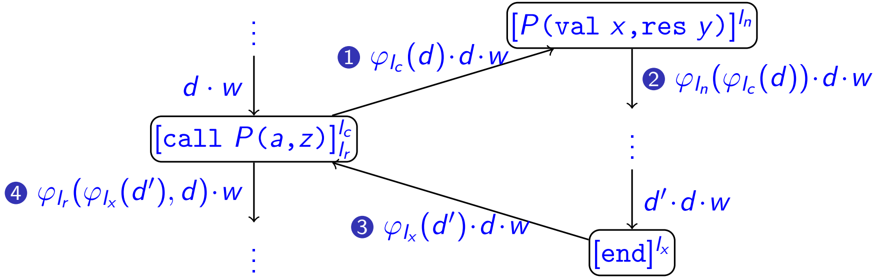
where

- $\hat{D} := D^+$
- $d_1 \dots d_n \mathrel{\hat{\sqsubseteq}} d'_1 \dots d'_n$ iff $d_i \sqsubseteq d'_i$ for every $1 \leq i \leq n$
- $\hat{\iota} := \iota \in D^+$
- $\hat{\varphi}_l : D^+ \to D^+$ where
  - for each $l \in Lab \setminus \{l_c, l_r \mid (l_c, l_n, l_x, l_r) \in \text{iflow}\}$:
    $$\hat{\varphi}_l(d \cdot w) := \varphi_l(d) \cdot w$$
  - for each $(l_c, l_n, l_x, l_r) \in \text{iflow}$:
    $$\hat{\varphi}_{l_c}(d \cdot w) := \varphi_{l_c}(d) \cdot d \cdot w$$
    $$\hat{\varphi}_{l_r}(d' \cdot d \cdot w) := \varphi_{l_r}(d', d) \cdot w$$

The Interprocedural Extension II

Visualization of

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



The Interprocedural Extension III

Example 19.2 (Constant Propagation; cf. Lecture 5/6)

$\hat{S} := (Lab, E, F, (\hat{D}, \hat{\sqsubseteq}), \hat{\iota}, \hat{\varphi})$ 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 $l \in Lab \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in \text{iflow}\}$,

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

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

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

  - exit/return: reset parameters/set return value

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

Outline

Types of Equations

For an interprocedural dataflow system $\hat{S} := (Lab, E, F, (\hat{D}, \hat{\sqsubseteq}), \hat{\iota}, \hat{\varphi})$, the intraprocedural equation system (cf. Definition 4.9)

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

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

- for actual dataflow information: $\mathsf{AI}_l \in \hat{D}$
  - counterpart of intraprocedural AI
- for transfer functions of single nodes: $f_l : \hat{D} \to \hat{D}$
  - extension of intraprocedural transfer functions by special handling of procedure calls
- for transfer functions of complete procedures: $F_l : \hat{D} \to \hat{D}$
  - $F_l(w)$ yields information at l if surrounding procedure is called with information w
  - thus complete procedure represented by F_{l_x} ("procedure summary")

Formal Definition of Equation System

Dataflow equations:

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

Node transfer functions:

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

(if l not an exit label)

Procedure transfer functions:

$$F_l(w) = \begin{cases} w & \text{if } l = l_n \\ & \text{for some } (l_c, l_n, l_x, l_r) \in \text{iflow} \\ \bigsqcup\{f_{l'}(F_{l'}(w)) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

(if l occurs in some procedure)

As before: induces monotonic functional on lattice with ACC

$\Longrightarrow$ least fixpoint effectively computable

Example of Equation System

Example 19.3 (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:

$$
\begin{aligned}
\mathrm{AI}_1 &= f_4(\mathrm{AI}_4) \sqcup f_6(\mathrm{AI}_6)\\
\mathrm{AI}_2 &= f_1(\mathrm{AI}_1)\\
\mathrm{AI}_3 &= f_2(\mathrm{AI}_2)\\
\mathrm{AI}_4 &= \iota = \top\top\top\\
\mathrm{AI}_5 &= \mathrm{AI}_4\\
\mathrm{AI}_6 &= f_5(\mathrm{AI}_5)\\
\mathrm{AI}_7 &= \mathrm{AI}_6\\
\mathrm{AI}_8 &= f_7(\mathrm{AI}_7)
\end{aligned}
$$

Fixpoint iteration:

on the board

Node transfer functions:

$$
\begin{aligned}
\hat{\varphi}_1(\delta w) &= \delta w\\
\hat{\varphi}_2(\delta w) &= \delta[\mathtt{y} \mapsto val_\delta(\mathtt{2*(x-1)})]w\\
\hat{\varphi}_3(\delta w) &= \delta w\\
\hat{\varphi}_4(\delta w) &= \delta[\mathtt{x} \mapsto 2, \mathtt{y} \mapsto \top]\delta w\\
\hat{\varphi}_5(\delta'\delta w) &= \delta'[\mathtt{x} \mapsto \delta(\mathtt{x}), \mathtt{y} \mapsto \delta(\mathtt{y}), \mathtt{z} \mapsto \delta'(\mathtt{y})]w\\
\hat{\varphi}_6(\delta w) &= \delta[\mathtt{x} \mapsto \delta(\mathtt{z}), \mathtt{y} \mapsto \top]\delta w\\
\hat{\varphi}_7(\delta'\delta w) &= \delta'[\mathtt{x} \mapsto \delta(\mathtt{x}), \mathtt{y} \mapsto \delta(\mathtt{y}), \mathtt{z} \mapsto \delta'(\mathtt{y})]w
\end{aligned}
$$

$$
\begin{aligned}
f_1(\delta w) &= \hat{\varphi}_1(\delta w) = \delta w\\
f_2(\delta w) &= \hat{\varphi}_2(\delta w) = \delta[\mathtt{y} \mapsto val_\delta(\mathtt{2*(x-1)})]w\\
f_4(\delta w) &= \hat{\varphi}_4(\delta w) = \delta[\mathtt{x} \mapsto 2, \mathtt{y} \mapsto \top]\delta w\\
f_5(\delta w) &= \hat{\varphi}_5(\hat{\varphi}_3(F_3(\hat{\varphi}_4(\delta w)))) = \hat{\varphi}_5(F_3(\hat{\varphi}_4(\delta w)))\\
f_6(\delta w) &= \hat{\varphi}_6(\delta w) = \delta[\mathtt{x} \mapsto \delta(\mathtt{z}), \mathtt{y} \mapsto \top]\delta w\\
f_7(\delta w) &= \hat{\varphi}_7(\hat{\varphi}_3(F_3(\hat{\varphi}_6(\delta w)))) = \hat{\varphi}_7(F_3(\hat{\varphi}_6(\delta w)))\\
f_8(\delta w) &= \hat{\varphi}_8(\delta w) = \delta w
\end{aligned}
$$

Procedure transfer functions:

$$
\begin{aligned}
F_1(\delta w) &= \delta w\\
F_2(\delta w) &= f_1(F_1(\delta w)) = \delta w\\
F_3(\delta w) &= f_2(F_2(\delta w)) = \delta[\mathtt{y} \mapsto val_\delta(\mathtt{2*(x-1)})]w
\end{aligned}
$$