Semantics and Verification of Software

Lecture 21: Dataflow Analysis

Thomas Noll

Lehrstuhl für Informatik 2
RWTH Aachen University
noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw/

Summer semester 2007

Outline

Overview

- So far: only intraprocedural analyses (i.e., without user–defined functions or procedures)
- Now: interprocedural dataflow analysis
- Complications:
  - correct matching between calls and returns
  - parameter passing (aliasing effects)
- Here: simple setting
  - only top–level declarations, no blocks or nested declarations
  - mutually recursive procedures
  - one call–by–value and one call–by–result parameter
  - extension to multiple and call–by–value–result parameters straightforward

Extending the Syntax

Syntactic categories:

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

Context–free grammar:

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

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

An Example

(extension: multiple call-by–value parameters)

Example 21.1 (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_5;
    [call Fib(x-2, z, z)]^6_7;
[end]^8;
[call Fib(5, 0, v)]^9_10
```

Procedure Flow Graphs

Definition 21.2 (Procedure flow graphs)

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

$$\begin{aligned}
\mathsf{init}([\texttt{call } P(a,x)]^{l_c}_{l_r}) &:= l_c \\
\mathsf{final}([\texttt{call } P(a,x)]^{l_c}_{l_r}) &:= \{l_r\} \\
\mathsf{flow}([\texttt{call } P(a,x)]^{l_c}_{l_r}) &:= \{(l_c, l_n), (l_x, l_r)\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{init}(\texttt{proc } [P(\texttt{val } x, \texttt{res } y)]^{l_n} \texttt{ is } c \, [\texttt{end}]^{l_x}) &:= l_n \\
\mathsf{final}(\texttt{proc } [P(\texttt{val } x, \texttt{res } y)]^{l_n} \texttt{ is } c \, [\texttt{end}]^{l_x}) &:= \{l_x\} \\
\mathsf{flow}(\texttt{proc } [P(\texttt{val } x, \texttt{res } y)]^{l_n} \texttt{ is } c \, [\texttt{end}]^{l_x}) &:= \{(l_n, \mathsf{init}(c))\} \cup \\
&\quad \{(l, l_x) \mid l \in \mathsf{final}(c)\}
\end{aligned}$$

if $\texttt{proc } [P(\texttt{val } x, \texttt{res } y)]^{l_n} \texttt{ is } c \, [\texttt{end}]^{l_x}$ is in p.
Moreover the interprocural flow of a program $p\,c$ is defined by

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

Procedure Flow Graphs

Example 21.3 (Fibonacci numbers)

Flow graph of

```
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_5;
    [call Fib(x-2, z, z)]^6_7;
[end]^8;
[call Fib(5, 0, v)]^9_10
```

(on the board)

Here $IF = \{(9,1,8,10),(4,1,8,5),(6,1,8,7)\}$

Outline

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

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

- **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
  $\Longrightarrow$ analysis information imprecise (but still correct)

Naive Formulation II

Example 21.4 (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_5;
    [call Fib(x-2, z, z)]^6_7;
[end]^8;
[call Fib(5, 0, v)]^9_10
```

- "Valid" path: $[9, 1, 2, 3, 8, 10]$
- "Invalid" path: $[9, 1, 2, 4, 1, 2, 3, 8, 10]$

Outline

Valid Paths I

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

Definition 21.5 (Valid paths I)

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 productions:

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

Valid Paths II

Example 21.6 (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_5;
    [call Fib(x-2, z, z)]^6_7;
[end]^8;
[call Fib(5, 0, v)]^9_10
```

Thus

$$[9, 1, 2, 3, 8, 10] \in L(P[10, 11])$$

but

$$[9, 1, 2, 4, 1, 2, 3, 8, 10] \notin L(P[10, 11])$$

Grammar for $P[10, 11]$:
on the board

Definition 21.7 (Valid paths II)

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

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

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

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

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

The MVP Solution II

Definition 21.8 (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

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

where, for every $l \in Lab$,

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

Corollary 21.9

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

Proof.

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

□