Program Analysis
Lecture 14

Rayna Dimitrova

WS 2016/2017
Recap: A simple programming languages with procedures

\[ p ::= \text{proc } [P(\text{val } x, \text{res } y)]^{ln} \text{ is } c \text{ [end]}^{lx}; p \mid \varepsilon \]

\[ c ::= \text{[skip]}^l \mid [x := a]^l \mid [\text{read}(x)]^l \mid [\text{print}(a)]^l \mid c_1; c_2 \]

\[ \mid \text{if } [b]^l \text{ then } \{c_1\} \text{ else } \{c_2\} \]

\[ \mid \text{while } [b]^l \text{ do } \{c\} \]

\[ \mid [\text{call } P(a, z)]^{lc}_{lr} \]
Recap: Dataflow analysis for programs with procedures

We are given a dataflow system $S = (G, (D, \leq), i, \{f_l\}_{l \in \text{Labels}})$

- $G = (B, E, F, \text{iflow})$ is the control flow graph
- $(D, \leq)$ is a complete lattice that satisfies (ACC)
- $i \in D$ is an initial value for extremal blocks in $E_c$
- $\{f_l\}_{l \in \text{Labels}}$, is a family of monotonic functions $f_l : D \rightarrow D$, one for each of the labels in the program $l \in \text{Labels}$

For a block $\text{[call } P(a, z)\text{]}_{l_r}$, we have two functions $f_{l_c}$ and $f_r$.

- $f_{l_c} : D \rightarrow D$ modifies the dataflow properties as required for passing to the procedure entry (set input, reset output parameter).
- $f_{l_r} : D \times D \rightarrow D$ modifies the dataflow properties as required for coming back from the procedure (reset parameters, set return value).
Recap: System of equations for procedure summaries

Consider $S = (G, (D, \leq), i, \{f_l\}_{l \in \text{Labels}})$ with $G = (B, E, F, \text{iflow})$.

We define the following equations

$\triangleright$ $Z_P = Y_{lx}$, where $\text{proc} [P(\text{val } x, \text{res } y)]^{ln}$ is $c'$ $\text{end}^{lx}$ is in $p$

$\triangleright$ for block $b$ in the definition of some procedure

$$Y_b = \begin{cases} id & \text{if } b = [P(\text{val } x, \text{res } y)]^{ln}, \\ \bigcup \{g_{b'} \circ Y_{b'} \mid b' \in \text{pred}(b)\} & \text{otherwise} \end{cases}$$

Function $g_{b'}$ for a block $b'$.

$\triangleright$ for blocks $[\ldots]^l$ in the intraprocedural fragment, and for blocks of the form $[P(\text{val } x, \text{res } y)]^l$ and $\text{end}^l$, $g_b = f_l$ is given in $S$

$\triangleright$ for blocks $b' = [\text{call } P(a, z)]^{lc}_{lr}$

$$g_{b'} = \lambda d. f_{lr}(d, Z_P \circ f_{lc}(d))$$

$Z_P \circ f_{lc}$ passes the dataflow information to the call of $Z_P$

$f_{lr}(d, d')$ combines the result $d'$ with the prior information $d$
Consider $S = (G, (D, \leq), i, \{f_l\}_{l \in \text{Labels}})$ with $G = (B, E, F, \text{iflow})$.

Consider the induced system of procedure-summary equations.

**Theorem**

If $(h_1, \ldots, h_{|B|}, f_1, \ldots, f_{|P|}) \in \hat{D}^{|B|+|P|}$ is the least solution (w.r.t. $\sqsubseteq^{|B|+|P|}$) of the system of procedure-summary equations, then,

- if $b$ is a block in a procedure $P$, and $h_b$ is the function computed for $b$, then $h_b \sqsupseteq f_\pi$ for every valid path $\pi$ from the entry of the procedure $P$ to the entry point of block $b$.
- if $f_P$ is the function computed for a procedure $P$, then $f_P \sqsupseteq f_\pi$ for every valid path $\pi$ from the entry to the exit of $P$. 

Least fixpoint solution of the procedure-summary equations
Dataflow analysis using procedure summaries

Consider $S = (G, (D, \leq), i, \{f_l\}_{l \in \text{Labels}})$ with $G = (B, E, F, \text{iflow})$.

Suppose we have computed $(h_1, \ldots, h_{|B|}, f_1, \ldots, f_{|p|}) \in \hat{D}^{B|+|p|}$, the least solution of the system of procedure-summary equations.

We define the system of dataflow equations over variables $X_b$

$$X_b = \begin{cases} i & \text{if } b \in E_c \\ \bigsqcup \{f_{l_c}(X_{b'}) \mid (l_c, l_n, \ldots) \in \text{iflow}, b' = [...]^{l_c}_{l_r}\} & \text{if } b = [...]^{l_n} \in E \setminus E_c \\ \bigsqcup \{f_{b'}(X_{b'}) \mid (b', b) \in F\} & \text{otherwise} \end{cases}$$

where the transfer function $f_b$ for a block $b$ is

- given in $S$ for blocks of the form $[...]^l$
- defined as $f_{b'} = \lambda d. f_{l_r}(d, f_P(f_{l_c}(d)))$ for $b' = [\text{call } P(a, z)]^{l_c}_{l_r}$
Example: analysis with procedure summaries

```
proc [Zero(val x, res y)] is
  if [¬(x = 0)] then {
    [call Zero(0, y)]
  }
  else {
    [assert(x = 0)];
    [y := x]
  }
[end];
[call Zero(42, z);
[print(z)]
```

\[ D = (Vars \rightarrow \{0, ?\}) \cup \{\bot\} \]

\[ \top = (?, ?, ?, \ldots) \]

determine if variable is constant 0

\[ X_{8,9} = i \]
\[ X_{10} = f_9(X_{8,9}, f_{Zero}(f_8(X_{8,9}))) \]
\[ X_1 = f_8(X_{8,9}) \sqcup f_3(X_{3,4}) \]
\[ X_2 = X_1 \]
\[ X_{3,4} = X_2 \]
\[ X_5 = X_2 \]
\[ X_6 = f_5(X_5) \]
\[ X_7 = f_4(X_3, f_{Zero}(f_3(X_3))) \sqcup f_6(X_6) \]
Dataflow analysis using procedure summaries

Theorem
Consider $S = (G, (D, \leq), i, \{f_l\}_{l \in \text{Labels}})$ with $G = (B, E, F, \text{iflow})$, and $(f_P)_{P \in p}$ be (approximations of) the procedure summaries. If $(X_1^{LFP}, \ldots, X_{|B|}^{LFP})$ is the least fixpoint solution of the associated system of dataflow equations and $(X_1^{JOVP}, \ldots, X_{|B|}^{JOVP})$ is the $JOVP$ solution of $S$, then the following properties hold

1. $X_b^{JOVP} \leq X_b^{LFP}$ for all blocks $b$.
2. If all transfer functions are distributive, then $X_b^{JOVP} = X_b^{LFP}$ for all blocks $b$. 
control-state reachability problem for Boolean programs

Given: recursive program \( p; c \) with Boolean variables, block \( b \) in \( p; c \)

Problem: determine whether there exists an execution reaching \( b \)

Theorem The control-state reachability problem for recursive Boolean programs is decidable.

Proof sketch The problem can be formulated a Reachable Values analysis question: block \( b \) is reachable if and only if \( X_b^{JOVP} \neq \emptyset \). Since the transfer functions for the Reachable Values analysis are distributive the \( JOVP \) solution can be computed by computing the least fixpoint solution and used to check control state reachability.
Methods for dataflow analysis

Goal: approximate the $JOVP$ solution

Approach: compute least fixpoint solution where we avoid including too many invalid paths. We consider two methods:

1. procedure summaries
   - compute transfer functions for complete procedures
   - perform dataflow analysis using the procedure summaries

   Procedure summary computation ignores the call context.

2. including explicit call context
   - encode the call history into the dataflow information
   - formulate dataflow equations corresponding to procedures
Explicit context

Consider a dataflow system $S = (G, (D, \leq), i, \{f_l\}_{l \in \text{Labels}})$.

We will define a corresponding dataflow system that includes explicit context information in the dataflow information.

For a given set $\Delta$, where $\delta \in \Delta$ is context information the constructed dataflow system will be over $\tilde{D} = \delta \rightarrow D$.

Intuition: the dataflow information at a given program point depends on the context information.

We will consider a specific kind of context information that encodes the taken path, as a sequence of procedure calls.
Call strings as context

A call string encodes the sequence of pending procedure calls.

Let $\Delta = Labels^*$ and define $\tilde{D} = \Delta \rightarrow D$ and $\tilde{d}_1 \leq \tilde{d}_2$ iff

$$\tilde{d}_1(\delta) \leq \tilde{d}_2(\delta) \text{ for all } \delta.$$  

Intuition: the rightmost element of $\delta \in \Delta$ is the most recent call.
Example: call strings

```plaintext
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)]5;
   }
[end]8;
[call Fib(5, 0, v)]9
```

The following call strings, corresponding to valid paths from the initial block, are of interest:

ε (the empty string, 0 pending calls),
[9] (1 pending call),
[9, 4], [9, 6] (2 pending calls),
[9, 4, 4], [9, 4, 6], [9, 6, 4], [9, 6, 6] . . .
Call-strings dataflow system

For $S = (G, (D, \leq), i, \{f_l\}_{l \in \text{Labels}})$ with $G = (B, E, F, \text{iflow})$ we define $\tilde{S} = (G, (\tilde{D}, \leq), \tilde{i}, \{\tilde{f}_l\}_{l \in \text{Labels}})$ as follows.

- The initial value is
  
  $$\tilde{i}(\delta) = \begin{cases} 
i & \text{if } \delta = \varepsilon \\
\bot & \text{otherwise,} \end{cases}$$

  where $\varepsilon$ is the empty string (no pending procedure calls).

- For a block of the form $[\ldots]^l$ we define
  
  $$\tilde{f}_l(\tilde{d})(\delta) = f_l(\tilde{d}(\delta)).$$

  **Intuition:** The function $f_l$ given in $S$ is applied pointwise.
Call-strings dataflow system

For $\textbf{call} \ P(a, z)_{l_c}$, we define the transfer functions

$$\tilde{f}_{l_c}(\tilde{d})(\delta) = \begin{cases} f_{l_c}(\tilde{d}(\delta')) & \text{if } \delta = [\delta', l_c] \\ \bot & \text{otherwise} \end{cases}$$

where $\delta = [\delta', l_c]$ is the path obtained from $\delta'$ by appending $l_c$;

$$\tilde{f}_{l_r}(\tilde{d}_1, \tilde{d}_2)(\delta) = f_{l_r}(\tilde{d}_1(\delta), \tilde{d}_2([\delta, l_c])).$$

Note: the function $\tilde{f}_{l_r}$ combines dataflow information from before and after the call at $l_c$ only from the same context $\delta$. 
Example: transfer functions

Recall the detection of signs analysis where

\[ D_{\text{sign}} = \mathcal{P}(\text{Vars} \rightarrow \{-, 0, +\}). \]

\[ \Delta \rightarrow D_{\text{sign}} \] is isomorphic to \( \tilde{D} = \mathcal{P}(\Delta \times (\text{Vars} \rightarrow \{-, 0, +\})). \)

Consider a call \([\text{call } P(a, z)]_{lr}^{lc}\) for a procedure \(P\) defined as \(\text{proc } [P(\text{val } x, \text{res } y)]_{ln}^{l_n} \text{ is } c' [\text{end}]_{lx}\). We define the functions:

\[ \tilde{f}_{lc}(\tilde{d}) = \bigcup \{\{\delta'\} \times \varphi_{lc}(\sigma) \mid (\delta, \sigma) \in \tilde{d} \land \delta' = [\delta, l_c]\}, \text{ where} \]

\[ \varphi_{lc}(\sigma) = \{\sigma[x \rightarrow s, y \rightarrow s'] \mid s \in \mathcal{A}[a](\sigma), s' \in \{-, 0, +\}\} \]

Intuition: \(\tilde{f}_{lc}\) appends \(l_c\) to the call contexts in \(\tilde{d}\)
Example: transfer functions

Recall the detection of signs analysis where

\[ D_{\text{sign}} = \mathcal{P}(\text{Vars} \rightarrow \{-, 0, +\}) \]

\[ \Delta \rightarrow D_{\text{sign}} \] is isomorphic to \( \widetilde{D} = \mathcal{P}(\Delta \times (\text{Vars} \rightarrow \{-, 0, +\})) \).

Consider a call \([[\text{call } P(a, z)]_{lc}] \) for a procedure \( P \) defined as \( \text{proc } [P(\text{val } x, \text{res } y)]_{ln} \text{ is } c' \) \([\text{end}]_{lx} \). We define the functions:

\[ \tilde{f}_{lr}(\tilde{d}_1, \tilde{d}_2) = \bigcup \left\{ \{\delta\} \times \varphi_{lr}(\sigma_1, \sigma_2) \mid (\delta, \sigma_1) \in \tilde{d}_1 \land ([\delta, l_c], \sigma_2) \in \tilde{d}_2 \right\} \]

\[ \varphi_{lr}(\sigma_1, \sigma_2) = \{ \sigma_2[x \rightarrow \sigma_1(x), y \rightarrow \sigma_1(y), z \rightarrow \sigma_2(y)] \} \]

Intuition: \( \tilde{f}_{lr} \) combines information with the same context

Remark: all variables except for the formal parameters \( x, y \) and the actual parameter \( z \) are global and can be modified by the call
Fixpoint solution

Given \( S = (G, (D, \leq), i, \{ f_l \}_{l \in \text{Labels}}) \) with \( G = (B, E, F, \text{iflow}) \) the call-string dataflow system \( \tilde{S} = (G, (\tilde{D}, \leq), \tilde{i}, \{ \tilde{f}_l \}_{l \in \text{Labels}}) \), induces the system of equations over variables \( X_l \) for \( l \in \text{Labels} \):

\[
X_l = \tilde{i} \text{ if } l \in E_c
\]

\[
X_l = \bigsqcup \{ \tilde{f}_{l'}(X_{l'}) \mid (l', l) \in F \}
\]
\[
\bigsqcup \{ \tilde{f}_{l_c}(X_{l_c}) \mid (l_c, l, l_x, l_r) \in \text{iflow} \}
\]
\[
\bigsqcup \{ f_{l}(X_{l_c}, X_{l_x}) \mid (l_c, l_n, l_x, l) \in \text{iflow} \}
\]

Theorem The least fixpoint solution overapproximates \( JOVP(S) \).
Call strings of bounded length

Let $\Delta^{\leq k}$ be the set of call strings of length up to $k \geq 0$.

Note: for $k = 0$, we have $\Delta = \{\varepsilon\}$, i.e., no context information

Idea: truncate call strings from the left when bound is exceeded:

$$
\tilde{f}_{lc}(\tilde{d})(\delta') = \bigsqcup \{f_{lc}(\tilde{d}(\delta)) \mid \delta' = [[\delta, l_c]]_k \}
$$

$$
\tilde{f}_{lr}(\tilde{d}_1, \tilde{d}_2)(\delta) = f_{lr}(\tilde{d}_1(\delta), \tilde{d}_2([[\delta, l_c]]_k))
$$

where $[l'_1, \ldots, l'_m, l_1, \ldots, l_k] = l_1, \ldots, l_k$

Note: the function $\tilde{f}_{lc}$ takes the least upper bound over all possible contexts $\delta$ that can be mapped to $[[\delta, l_c]]_k$
Example: call strings of bounded length

\[
\text{proc [Fib(val } x, y, \text{ res } z)\text{]}^1 \text{ is }
\]
\[
\text{if } [x < 2]^2 \text{ then } \{
\]
\[
[z := y + 1]^3
\]
\[
\}
\]
\[
\text{else } \{
\]
\[
[\text{call Fib}(x - 1, y, z)]^4; \quad [\text{call Fib}(x - 2, z, z)]^6
\]
\[
\}
\]
\[
[end]^8;
\]
\[
[\text{call Fib}(5, 0, v)]^9
\]

\[k = 1: \text{ record only last call }\]
\[\varepsilon, [9], [4], [6]\]

\[k = 2: \text{ more precise analysis with }\]
\[\varepsilon, [9, 4], [9, 6], [4, 4], [4, 6], [6, 4], [6, 6]\]
Example: transfer functions

Recall the detection of signs analysis with

\[ D_{\text{sign}} = \mathcal{P}(\text{Vars} \rightarrow \{-, 0, +\}). \]

\[ \Delta \rightarrow D_{\text{sign}} \] is isomorphic to \( \mathcal{P}(\Delta \times (\text{Vars} \rightarrow D_{\text{sign}})) \).

Consider \([\text{call } P(a, z)]_{l_c}^{l_r} \) for \( \text{proc } [P(\text{val} x, \text{res} y)]_{l_n}^{l_x} \) is \( c' [\text{end}]_{l_x}^{l_x} \).

\( k=0: \tilde{d}, \tilde{d}' \in \mathcal{P}(\text{Vars} \rightarrow \{-, 0, +\}) \)

\[ \tilde{f}_{l_c}(\tilde{d}) = \bigcup \{ \varphi_{l_c}(\sigma) \mid \sigma \in \tilde{d} \}, \]

\[ \tilde{f}_{l_r}(\tilde{d}, \tilde{d}') = \bigcup \{ \varphi_{l_r}(\sigma_1, \sigma_2) \mid \sigma_1 \in \tilde{d} \wedge \sigma_2 \in \tilde{d}' \} \]

Note: context insensitive analysis
Example: transfer functions

Recall the detection of signs analysis with

\[ D_{\text{sign}} = \mathcal{P}(\text{Vars} \to \{-, 0, +\}) \].

\( \Delta \to D_{\text{sign}} \) is isomorphic to \( \mathcal{P}(\Delta \times (\text{Vars} \to D_{\text{sign}})) \).

Consider \( \text{call } P(a, z) \) for \( \text{proc } P(\text{val } x, \text{res } y) \) is \( c' \) \( \text{end} \).

\( k=1: \tilde{d}, \tilde{d}' \in \mathcal{P}((\text{Labels} \cup \{\varepsilon\}) \times (\text{Vars} \to \{-, 0, +\})) \)

\[ \tilde{f}_{l_c}(\tilde{d}) = \bigcup \left\{ \{l_c\} \times \varphi_{l_c}(\sigma) \mid (\delta, \sigma) \in \tilde{d} \right\}, \]

\[ \tilde{f}_{l_r}(\tilde{d}, \tilde{d}') = \bigcup \left\{ \{\delta\} \times \varphi_{l_r}(\sigma_1, \sigma_2) \mid (\delta, \sigma_1) \in \tilde{d} \land (l_c, \sigma_2) \in \tilde{d}' \right\} \]

Note: the function \( \tilde{f}_{l_r} \) selects the pairs \( (l_c, \sigma_2) \) that are relevant for the current call and combines them with the pairs before the call.
What topics we did not cover in the lecture

- pointer analysis,
- heap structures,
- and many more ...
Oral exam

Program Analysis + Complexity Theory exam: 17 March 2017

Program Analysis exam: 23 March 2017

Precise schedule and room will be announced later.