InterProcedure-I

Thursday, May 19, 2022 2:55 PM

⇒ CMU : Claire Le Goues

interprocedural control-flow analysis.

⇒ example: flow, in function call & return:

$flow(x \leftarrow g(y))\sigma = [x \rightarrow L_r]\sigma$, where $\sigma(y) \sqsubseteq L_a$.

$flow(return)\sigma = \sigma$. where $\sigma(x) \sqsubseteq L_r$.

if $L_a = L_r = T$:

fun: g(x) : int

y = 10/x.

return y.

z = 5.

w = g(z).

$\sigma : [w \rightarrow L_r]\sigma_0$ where $\sigma_0(z) \sqsubseteq L_a$.

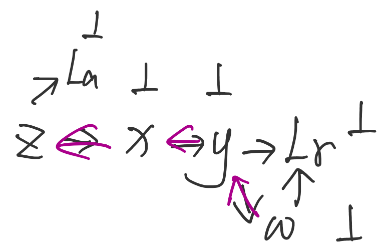
⇒ ⇒ $\sigma_0 = \{z \rightarrow L_a\} \sigma_1^*$

$[return\ y]\sigma_1 = \sigma_1$: $\sigma_1(y) \sqsubseteq L_r$.

⇒ $\sigma_1 = \{y \rightarrow L_r\}$.

⇒ $\sigma = \{w \rightarrow L_r,\ z \rightarrow L_a,\ y \rightarrow L_r,$ $z \leftarrow x,\ x \leftarrow y.\}$

$y \leftarrow w$.



⇒ Harvard. Stephen Chong:

⇒ call graph: ⇒ 1 big CFG with call graph.

⇒ interprocedural CFG. ⇒ treat argument, return value as assignments.

⇒ problem: all different calls of a function, its flow paths are merged

⇒ inline: copy a new function's CFG every time it is called.

⇒ problem: no recursive call

~~no~~ size increase exponentially.

⇒ context Sensitive analysis.

Syntactic Context:

⇒ only produce 1 copy if the nested call or multiple call of a function isn't explicitly inlining in program.

eg: main() → p() q()

1: p() 3: q().

2: p()

context 0 → context 1 → context 3

call p Enter p Enter q

call q

call p → context 2

Enter p

call q

Call-Site Stack Context Sensitive

context 0 → context 0:1 → context 0:1:3

→ context 0:2 → context 0:2:3.

similar for nested call.

others: caller stack. less precise.