



Static Program Analysis

Lecture 19: Interprocedural Dataflow Analysis II (Fixpoint Solution)

Winter Semester 2016/17

Thomas Noll

Software Modeling and Verification Group

RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ws-1617/spa/>

Recap: The CEGAR Refinement Step

Outline of Lecture 19

Recap: The CEGAR Refinement Step

Recap: Interprocedural Dataflow Analysis

The Interprocedural Fixpoint Solution

The Equation System

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples I

Lemma

If $\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$ is a spurious counterexample, there exist Boolean expressions b_0, \dots, b_k with $b_0 \equiv \text{true}$, $b_k \equiv \text{false}$, and

$$\forall i \in \{1, \dots, k\}, \sigma, \sigma' \in \Sigma : \sigma \models b_{i-1} \wedge \langle c_{i-1}, \sigma \rangle \rightarrow \langle c_i, \sigma' \rangle \implies \sigma' \models b_i$$

Proof (idea).

Inductive definition of b_i as **strongest postconditions**:

1. $b_0 := \text{true}$

2. for $i = 1, \dots, k$: definition of b_i depending on b_{i-1} and on (axiom) transition rule applied in

$\langle c_{i-1}, \cdot \rangle \Rightarrow \langle c_i, \cdot \rangle$:

– (skip) $b_i := b_{i-1}$

– (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$

(for $x := a$; x' = previous value of x)

– (if1) $b_i := b_{i-1} \wedge b$

– (if2) $b_i := b_{i-1} \wedge \neg b$

– (wh1) $b_i := b_{i-1} \wedge b$

– (wh2) $b_i := b_{i-1} \wedge \neg b$

(yields $b_k \equiv \text{false}$; by induction on k)

□

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- **Interesting property:** after termination, $x \neq y$, i.e., label 4 unreachable

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- **Interesting property:** after termination, $x \neq y$, i.e., label 4 unreachable
- **Initial abstraction:** $P = \emptyset$ ($\implies Abs(P) = \{true, false\}$)

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- **Interesting property:** after termination, $x \neq y$, i.e., label 4 unreachable
- **Initial abstraction:** $P = \emptyset$ ($\implies Abs(P) = \{\text{true}, \text{false}\}$)
- (Spurious) **counterexample:** $\langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle$

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- **Interesting property:** after termination, $x \neq y$, i.e., label 4 unreachable
- **Initial abstraction:** $P = \emptyset$ ($\implies Abs(P) = \{\text{true}, \text{false}\}$)
- (Spurious) **counterexample:** $\langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle$
- Forward construction of **strongest postconditions:**
 - $b_0 := \text{true}$

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- **Interesting property:** after termination, $x \neq y$, i.e., label 4 unreachable
- **Initial abstraction:** $P = \emptyset$ ($\implies Abs(P) = \{\text{true}, \text{false}\}$)
- (Spurious) **counterexample:** $\langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle$
- Forward construction of **strongest postconditions:**
 - $b_0 := \text{true}$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_1 := \exists x'. (b_0[x \mapsto x'] \wedge x = z[x \mapsto x']) \equiv (x = z)$

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- **Interesting property:** after termination, $x \neq y$, i.e., label 4 unreachable
- **Initial abstraction:** $P = \emptyset$ ($\implies Abs(P) = \{\text{true}, \text{false}\}$)
- (Spurious) **counterexample:** $\langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle$
- Forward construction of **strongest postconditions:**
 - $b_0 := \text{true}$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_1 := \exists x'. (b_0[x \mapsto x'] \wedge x = z[x \mapsto x']) \equiv (x = z)$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_2 := \exists z'. (b_1[z \mapsto z'] \wedge z = z + 1[z \mapsto z'])$
 $= \exists z'. (x = z' \wedge z = z' + 1) \equiv (x + 1 = z)$

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- **Interesting property:** after termination, $x \neq y$, i.e., label 4 unreachable
- **Initial abstraction:** $P = \emptyset$ ($\implies Abs(P) = \{\text{true}, \text{false}\}$)
- (Spurious) **counterexample:** $\langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle$
- Forward construction of **strongest postconditions:**
 - $b_0 := \text{true}$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_1 := \exists x'. (b_0[x \mapsto x'] \wedge x = z[x \mapsto x']) \equiv (x = z)$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_2 := \exists z'. (b_1[z \mapsto z'] \wedge z = z + 1[z \mapsto z'])$
 $= \exists z'. (x = z' \wedge z = z' + 1) \equiv (x + 1 = z)$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_3 := \exists y'. (b_2[y \mapsto y'] \wedge y = z[y \mapsto y']) \equiv (x + 1 = z \wedge y = z)$

Recap: The CEGAR Refinement Step

Elimination of Spurious Counterexamples II

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- **Interesting property:** after termination, $x \neq y$, i.e., label 4 unreachable
- **Initial abstraction:** $P = \emptyset$ ($\implies Abs(P) = \{\text{true}, \text{false}\}$)
- (Spurious) **counterexample:** $\langle 0, \text{true} \rangle \Rightarrow \langle 1, \text{true} \rangle \Rightarrow \langle 2, \text{true} \rangle \Rightarrow \langle 3, \text{true} \rangle \Rightarrow \langle 4, \text{true} \rangle$
- Forward construction of **strongest postconditions:**
 - $b_0 := \text{true}$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_1 := \exists x'. (b_0[x \mapsto x'] \wedge x = z[x \mapsto x']) \equiv (x = z)$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_2 := \exists z'. (b_1[z \mapsto z'] \wedge z = z + 1[z \mapsto z'])$
 $= \exists z'. (x = z' \wedge z = z' + 1) \equiv (x + 1 = z)$
 - (asgn) $b_i := \exists x'. (b_{i-1}[x \mapsto x'] \wedge x = a[x \mapsto x'])$
 $\implies b_3 := \exists y'. (b_2[y \mapsto y'] \wedge y = z[y \mapsto y']) \equiv (x + 1 = z \wedge y = z)$
 - (if1) $b_i := b_{i-1} \wedge b$
 $\implies b_4 := (b_3 \wedge x = y) \equiv (x + 1 = z \wedge y = z \wedge x = y) \equiv \text{false}$

Recap: The CEGAR Refinement Step

Abstraction Refinement

- Using b_1, \dots, b_{k-1} as computed before, let $P' := P \cup \{p_1, \dots, p_n\}$ where p_1, \dots, p_n are the **atomic conjuncts** occurring in b_1, \dots, b_{k-1}
- Refine $Abs(P)$ to $Abs(P')$

Lemma

After refinement, the spurious counterexample

$$\langle c_0, \text{true} \rangle \Rightarrow \langle c_1, Q_1 \rangle \Rightarrow \dots \Rightarrow \langle c_k, Q_k \rangle$$

with $Q_k \not\equiv \text{false}$ does not exist anymore.

Proof.

omitted □

Recap: The CEGAR Refinement Step

Abstract Semantics for Predicate Abstraction

Definition (Execution relation for predicate abstraction)

If $c \in \text{Cmd}$ and $Q \in \text{Abs}(P)$, then $\langle c, Q \rangle$ is called an **abstract configuration**. The **execution relation for predicate abstraction** is defined by the following rules:

$$\begin{array}{c} \text{(skip)} \frac{}{\langle \text{skip}, Q \rangle \Rightarrow \langle \downarrow, Q \rangle} \quad \text{(asgn)} \frac{}{\langle x := a, Q \rangle \Rightarrow \langle \downarrow, \bigsqcup \{ Q_{\sigma[x \mapsto \text{val}_{\sigma}(a)]} \mid \sigma \models Q \} \rangle} \\ \text{(seq1)} \frac{\langle c_1, Q \rangle \Rightarrow \langle c'_1, Q' \rangle \quad c'_1 \neq \downarrow}{\langle c_1; c_2, Q \rangle \Rightarrow \langle c'_1; c_2, Q' \rangle} \quad \text{(seq2)} \frac{\langle c_1, Q \rangle \Rightarrow \langle \downarrow, Q' \rangle}{\langle c_1; c_2, Q \rangle \Rightarrow \langle c_2, Q' \rangle} \\ \text{(if1)} \frac{}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end}, Q \rangle \Rightarrow \langle c_1, \overline{Q \wedge b} \rangle} \\ \text{(if2)} \frac{}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end}, Q \rangle \Rightarrow \langle c_2, \overline{Q \wedge \neg b} \rangle} \\ \text{(wh1)} \frac{}{\langle \text{while } b \text{ do } c \text{ end}, Q \rangle \Rightarrow \langle c; \text{while } b \text{ do } c \text{ end}, \overline{Q \wedge b} \rangle} \\ \text{(wh2)} \frac{}{\langle \text{while } b \text{ do } c \text{ end}, Q \rangle \Rightarrow \langle \downarrow, \overline{Q \wedge \neg b} \rangle} \end{array}$$

Recap: The CEGAR Refinement Step

A Simple Example

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- $P = \emptyset, P' = \{\underbrace{x = z}_{p_1}, \underbrace{x + 1 = z}_{p_2}, \underbrace{y = z}_{p_3}\}$
- Refined abstract transitions:
 $\langle 0, \text{true} \rangle$

Recap: The CEGAR Refinement Step

A Simple Example

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- $P = \emptyset, P' = \{\underbrace{x = z}_{p_1}, \underbrace{x + 1 = z}_{p_2}, \underbrace{y = z}_{p_3}\}$
- Refined abstract transitions:
$$\langle 0, \text{true} \rangle \Rightarrow \langle 1, p_1 \wedge \neg p_2 \rangle$$

Recap: The CEGAR Refinement Step

A Simple Example

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- $P = \emptyset, P' = \{\underbrace{x = z}_{p_1}, \underbrace{x + 1 = z}_{p_2}, \underbrace{y = z}_{p_3}\}$
- Refined abstract transitions:
$$\langle 0, \text{true} \rangle \Rightarrow \langle 1, p_1 \wedge \neg p_2 \rangle$$
$$\Rightarrow \langle 2, \neg p_1 \wedge p_2 \rangle$$

Recap: The CEGAR Refinement Step

A Simple Example

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- $P = \emptyset, P' = \{\underbrace{x = z}_{p_1}, \underbrace{x + 1 = z}_{p_2}, \underbrace{y = z}_{p_3}\}$

- Refined abstract transitions:

$$\begin{aligned}\langle 0, \text{true} \rangle &\Rightarrow \langle 1, p_1 \wedge \neg p_2 \rangle \\ &\Rightarrow \langle 2, \neg p_1 \wedge p_2 \rangle \\ &\Rightarrow \langle 3, \neg p_1 \wedge p_2 \wedge p_3 \rangle\end{aligned}$$

Recap: The CEGAR Refinement Step

A Simple Example

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- $P = \emptyset, P' = \{\underbrace{x = z}_{p_1}, \underbrace{x + 1 = z}_{p_2}, \underbrace{y = z}_{p_3}\}$

- Refined abstract transitions:

$$\begin{aligned}\langle 0, \text{true} \rangle &\Rightarrow \langle 1, p_1 \wedge \neg p_2 \rangle \\ &\Rightarrow \langle 2, \neg p_1 \wedge p_2 \rangle \\ &\Rightarrow \langle 3, \neg p_1 \wedge p_2 \wedge p_3 \rangle \\ &\Rightarrow \langle 4, \underbrace{\neg p_1 \wedge p_2 \wedge p_3 \wedge x=y}_{\equiv \text{false}} \rangle\end{aligned}$$

Recap: The CEGAR Refinement Step

A Simple Example

Example

- Let $c_0 := [x := z]^0; [z := z + 1]^1; [y := z]^2;$
if $[x = y]^3$ then $[skip]^4$ else $[skip]^5$ end
- $P = \emptyset, P' = \{\underbrace{x = z}_{p_1}, \underbrace{x + 1 = z}_{p_2}, \underbrace{y = z}_{p_3}\}$

- Refined abstract transitions:

$$\begin{aligned} \langle 0, \text{true} \rangle &\Rightarrow \langle 1, p_1 \wedge \neg p_2 \rangle \\ &\Rightarrow \langle 2, \neg p_1 \wedge p_2 \rangle \\ &\Rightarrow \langle 3, \neg p_1 \wedge p_2 \wedge p_3 \rangle \\ &\Rightarrow \langle 4, \underbrace{\neg p_1 \wedge p_2 \wedge p_3 \wedge x=y}_{\equiv \text{false}} \rangle \end{aligned}$$

Recap: The CEGAR Refinement Step

Another Example: Multiplication

Example

- Let $c_0 := [z := 0]^0$;
 while $[x > 0]^1$ do
 $[z := z + y]^2$;
 $[x := x - 1]^3$
 end;
 if $[z \bmod y = 0]^4$ then
 $[\text{skip}]^5$
 else
 $[\text{skip}]^6$
 end;
 - **Global assumption:** $y > 0$
 - **Interesting property:** label 6 unreachable (since z multiple of y)
 - **Initial abstraction:** $P = \emptyset$ ($\implies \text{Abs}(P) = \{\text{true}, \text{false}\}$)
 - **Abstraction refinement:** on the board
- $p_3 = (z \bmod y \neq 0)$ not required!

Recap: Interprocedural Dataflow Analysis

Outline of Lecture 19

Recap: The CEGAR Refinement Step

Recap: Interprocedural Dataflow Analysis

The Interprocedural Fixpoint Solution

The Equation System

Recap: Interprocedural Dataflow Analysis

Extending the Syntax

Syntactic categories:

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

Context-free grammar:

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

- All labels and procedure names in **program** p c 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** (input), second **call-by-result** (output)

Recap: Interprocedural Dataflow Analysis

Naive Formulation I

- **Attempt:** directly transfer **techniques from intraprocedural analysis**
⇒ 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 $[call\ P(a, x)]_{l_r}^{l_c}$:
transfer functions $\varphi_{l_c}, \varphi_{l_r} : D \rightarrow D$ (definition later)
- For each procedure declaration $proc\ [P(val\ x, res\ y)]^{l_n}\ is\ c\ [end]^{l_x}$:
transfer functions $\varphi_{l_n}, \varphi_{l_x} : D \rightarrow D$ (definition later)
- Induces **equation system**

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

- **Problem:** procedure calls $(l_c; l_n)$ and procedure returns $(l_x; l_r)$ treated like goto's
⇒ **nesting** of calls and returns ignored
⇒ too many **paths** considered
⇒ analysis information possibly **imprecise** (but still correct)

Recap: Interprocedural Dataflow Analysis

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;
  [y := y - 1]7
else
  [call P(2, y)]8;
  [y := y - 2]10
end;
[skip]11
```

Two “valid” and two “invalid” paths:

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

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

Recap: Interprocedural Dataflow Analysis

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

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

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

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

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

Recap: Interprocedural Dataflow Analysis

The MVP Solution II

Definition (MVP solution)

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

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

where, for every $l \in Lab$,

$$\text{mvp}(l) := \bigsqcup \{ \varphi_\pi(\iota) \mid \pi \in VPath(l) \}.$$

Corollary

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

Proof.

1. since $VPath(l) \subseteq Path(l)$ for every $l \in Lab$
2. as $\text{mvp}(S) = \text{mop}(S)$ in intraprocedural case and MOP solution undecidable (Thm. 7.1) \square

The Interprocedural Fixpoint Solution

Outline of Lecture 19

Recap: The CEGAR Refinement Step

Recap: Interprocedural Dataflow Analysis

The Interprocedural Fixpoint Solution

The Equation System

The Interprocedural Fixpoint Solution

Making Context Explicit

- **Goal:** adapt fixpoint solution to **avoid invalid paths**

The Interprocedural Fixpoint Solution

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)

The Interprocedural Fixpoint Solution

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**

The Interprocedural Fixpoint Solution

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

The Interprocedural Fixpoint Solution

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 Fixpoint Solution

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 $\varphi_{l_r} : D^2 \rightarrow D$ for each $(l_c, l_n, l_x, l_r) \in \text{iflow}$ (and $\varphi_l : D \rightarrow D$ otherwise).

The **interprocedural extension** of S is given by

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

where

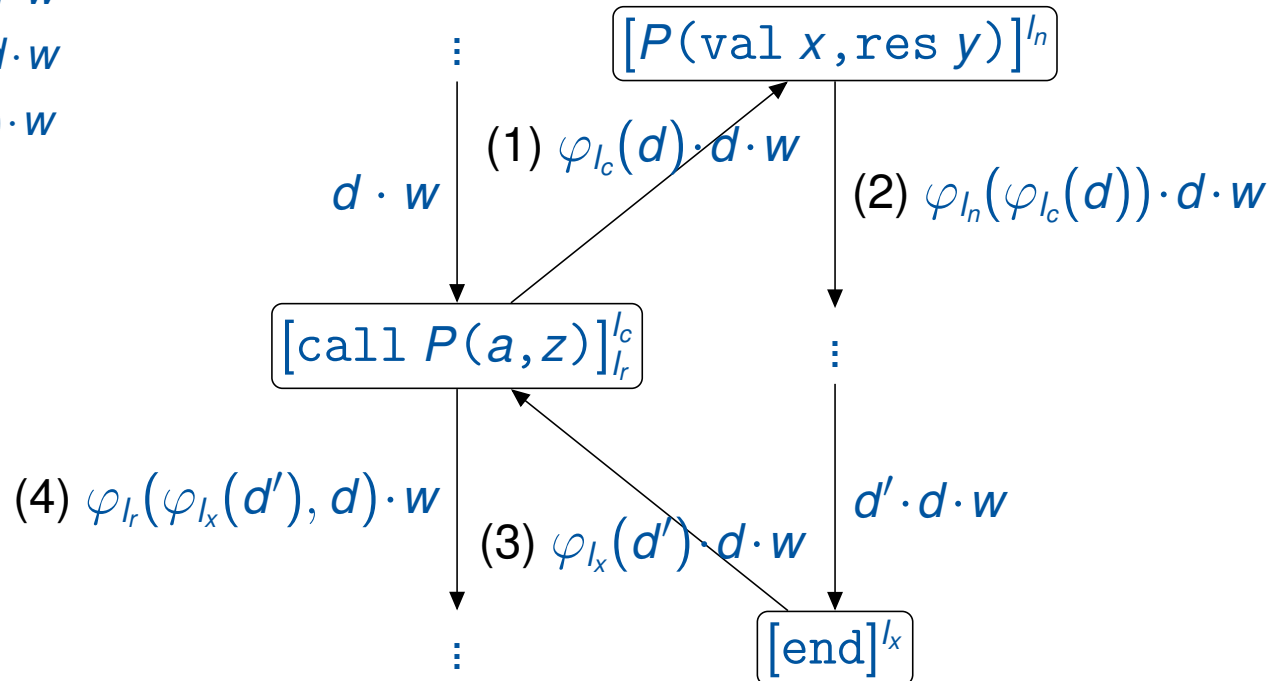
- $\hat{D} := D^+$
- $d_1 \dots d_n \hat{\sqsubseteq} d'_1 \dots d'_n$ iff $d_i \sqsubseteq d'_i$ for every $1 \leq i \leq n$
- $\hat{\iota} := \iota \in D^+$
- $\hat{\varphi}_l : D^+ \rightarrow D^+$ where
 - for each $l \in Lab \setminus \{l_c, l_r \mid (l_c, l_n, l_x, l_r) \in \text{iflow}\}$:
$$\hat{\varphi}_l(d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

The Interprocedural Extension II

Flow of information:

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 Fixpoint Solution

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

- $D := \{\delta \mid \delta : Var_c \rightarrow \mathbb{Z} \cup \{\perp, \top\}\}$ (constant/undefined/overdefined)
- $\perp \sqsubseteq z \sqsubseteq \top$ for every $z \in \mathbb{Z}$
- $\iota := \delta_{\top} \in D$
- For each $l \in Lab \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in \text{iflow}\}$,

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

The Interprocedural Fixpoint Solution

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

- $D := \{\delta \mid \delta : Var_c \rightarrow \mathbb{Z} \cup \{\perp, \top\}\}$ (constant/undefined/overdefined)
- $\perp \sqsubseteq z \sqsubseteq \top$ for every $z \in \mathbb{Z}$
- $\iota := \delta_{\top} \in D$
- For each $l \in Lab \setminus \{l_c, l_n, l_x, l_r \mid (l_c, l_n, l_x, l_r) \in \text{iflow}\}$,

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

- Whenever pc contains $[call\ P(a, z)]_{l_r}^{l_c}$ and $proc\ [P(val\ x, res\ y)]_{l_n}^{l_x}$ is $c\ [end]_{l_x}^{l_r}$,
 - **call/entry**: set input and 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 (x, y possible used in calling context) and set return value

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

The Equation System

Outline of Lecture 19

Recap: The CEGAR Refinement Step

Recap: Interprocedural Dataflow Analysis

The Interprocedural Fixpoint Solution

The Equation System

The Equation System

Types of Equations

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

$$AI_I = \begin{cases} \iota & \text{if } I \in E \\ \bigsqcup \{ \varphi_{I'}(AI_{I'}) \mid (I', I) \in F \} & \text{otherwise} \end{cases}$$

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

- for actual **dataflow information**: $AI_I \in \hat{D}$
 - counterpart of intraprocedural AI
- for **transfer functions of single nodes**: $f_I : \hat{D} \rightarrow \hat{D}$
 - extension of intraprocedural transfer functions by special handling of procedure calls
- for **transfer functions of complete procedures**: $F_I : \hat{D} \rightarrow \hat{D}$
 - $F_I(w)$ yields information at I if corresponding procedure is called with information w
 - thus complete procedure represented by F_{I_x} (“procedure summary”)

The Equation System

Formal Definition of Equation System

Dataflow equations:

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

The Equation System

Formal Definition of Equation System

Dataflow equations:

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

Node transfer functions (if l not an exit label):

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

The Equation System

Formal Definition of Equation System

Dataflow equations:

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

Node transfer functions (if l not an exit label):

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

Procedure transfer functions (if l occurs in some procedure):

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

The Equation System

Formal Definition of Equation System

Dataflow equations:

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

Node transfer functions (if l not an exit label):

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

Procedure transfer functions (if l occurs in some procedure):

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

As before: induces monotonic functional on lattice with ACC

⇒ least fixpoint effectively computable

The Equation System

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)]45;
[call P(z, z)]67;
[skip]8
```

The Equation System

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)]45;
[call P(z, z)]67;
[skip]8
```

Dataflow equations:

$$\begin{aligned} AI_1 &= f_4(AI_4) \sqcup f_6(AI_6) \\ AI_2 &= f_1(AI_1) \\ AI_3 &= f_2(AI_2) \\ AI_4 &= \iota = \top\top\top \\ AI_5 &= AI_4 \\ AI_6 &= f_5(AI_5) \\ AI_7 &= AI_6 \\ AI_8 &= f_7(AI_7) \end{aligned}$$

The Equation System

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)]45;
[call P(z, z)]67;
[skip]8
```

Dataflow equations:

$$\begin{aligned} Al_1 &= f_4(Al_4) \sqcup f_6(Al_6) \\ Al_2 &= f_1(Al_1) \\ Al_3 &= f_2(Al_2) \\ Al_4 &= \perp = \top\top\top \\ Al_5 &= Al_4 \\ Al_6 &= f_5(Al_5) \\ Al_7 &= Al_6 \\ Al_8 &= f_7(Al_7) \end{aligned}$$

Node transfer functions:

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

The Equation System

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)]45;
[call P(z, z)]67;
[skip]8
```

Dataflow equations:

$$\begin{aligned} Al_1 &= f_4(Al_4) \sqcup f_6(Al_6) \\ Al_2 &= f_1(Al_1) \\ Al_3 &= f_2(Al_2) \\ Al_4 &= \perp = \top\top\top \\ Al_5 &= Al_4 \\ Al_6 &= f_5(Al_5) \\ Al_7 &= Al_6 \\ Al_8 &= f_7(Al_7) \end{aligned}$$

Node transfer functions:

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

Procedure transfer functions:

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

The Equation System

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)]45;
[call P(z, z)]67;
[skip]8
```

Dataflow equations:

$$\begin{aligned} Al_1 &= f_4(Al_4) \sqcup f_6(Al_6) \\ Al_2 &= f_1(Al_1) \\ Al_3 &= f_2(Al_2) \\ Al_4 &= \perp = \top\top\top \\ Al_5 &= Al_4 \\ Al_6 &= f_5(Al_5) \\ Al_7 &= Al_6 \\ Al_8 &= f_7(Al_7) \end{aligned}$$

Fixpoint iteration:
on the board

Node transfer functions:

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

Procedure transfer functions:

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