02/15/19 Recitation Notes

17-355/17-665/17-819: Program Analysis (Spring 2019)
Jenna Wise
jlwise@andrew.cmu.edu

1 Reminders

- Homework 5 is due next **Thursday, February 21, 2019 at 11:59pm**. Instructions can be found on the course website

2 Program Analysis Correctness

2.1 Soundness

For an analysis of a WHILE3ADDR program to be correct, intuitively, we would like the program analysis results to correctly describe every actual execution of the program.

To formalize correctness, we start by formalizing a program execution as a trace:

Program Trace	A trace T of a program P is a potentially infinite sequence $\{c_0, c_1, ...\}$ of program configurations, where $c_0 = E_0, 1$ is called the initial configuration, and for every $i \geqslant 0$ we have $P \vdash c_i \rightsquigarrow c_{i+1}$

.

Given this definition, we can formally define soundness (correctness):

Dataflow Analysis Soundness	The result $\{\sigma_n \mid n \in P\}$ of a program analysis running on program P is sound iff, for all traces T of P, for all i such that $0 \leqslant i < \mathit{length}(T)$, $\alpha(c_i) \sqsubseteq \sigma_{n_i}$

In this definition, just as c_i is the program configuration immediately before executing instruction n_i as the ith program step, σ_{n_i} is the dataflow analysis information immediately before instruction n_i.

2.1.1 Showing Unsoundness of Parity Analysis

We can show that (incorrect) flow functions are unsound for *parity analysis* as we did in class for unsound flow functions for *zero analysis*. The idea is to find an example program and concrete trace that illustrates that the flow function is unsound. Make sure to show how the program and concrete trace illustrate the soundness. We will not practice showing global unsoundness of an incorrect flow function in this recitation, because we had enough practice with it in lecture and instead we will practice showing local unsoundness.

2.1.2 Local Soundness to Global Soundness

To prove global soundness (as formalized previously) of our dataflow analyses it is enough to show that our flow functions are monotonic and locally sound (formalized in Sections 2.2 & 2.3), thanks to the following:

Formal definition of the state of a dataflow analysis at a fixed point:

Fixed Point	A dataflow analysis result $\{\sigma_i \mid i \in P\}$ is a fixed point iff $\sigma_0 \sqsubseteq \sigma_1$ where σ_0 is the initial analysis information and σ_1 is the information before the first instruction, and for each instruction i we have $\bigsqcup_{j \in \texttt{preds}(i)} f[\![P[j]]\!](\sigma_j) \sqsubseteq \sigma_i$.

The lecture notes on Program Analysis Correctness, develop proofs that the worklist algorithm, which we use to run our data analyses, terminates with analysis results that reach a fixed point so long as our flow functions are monotonic and our lattice has finite height.

Finally, the lecture notes on Program Analysis Correctness also develop a proof of the following theorem:

Theorem 1 (A fixed point of a locally sound analysis is globally sound)**.** *If a dataflow analysis's flow function f is monotonic and locally sound, and for all traces T we have $\alpha(c_0) \sqsubseteq \sigma_0$ where σ_0 is the initial analysis information, then any fixed point $\{\sigma_n \mid n \in P\}$ of the analysis is sound.*

Proof. See the lecture notes on Program Analysis Correctness for the proof.

□

2.2 Monotonicity

Monotonicity of flow functions is important for both termination and correctness of program analyses.

Monotonicity	A function f is *monotonic* iff $\sigma_1 \sqsubseteq \sigma_2$ implies $f(\sigma_1) \sqsubseteq f(\sigma_2)$

2.2.1 Proving Monotonicity of Parity Analysis

We can formally show that *parity analysis* is monotone. We will only show one interesting case of the flow function for the multiplication of two variables assigned to a third variable, ie. $x := y * z$, and leave the rest as an exercise to the reader (you can find some other cases for other program statements for *zero analysis* in the lecture notes on Program Analysis Correctness):

The parity analysis flow function for the multiplication of two variables assigned to a third variable:

$$f_P[\![x := y * z]\!](\sigma) = \begin{cases} \sigma[x \mapsto \bot] & \text{if } \sigma(y) = \bot \vee \sigma(z) = \bot \\ \sigma[x \mapsto e] & \text{if } (\sigma(y) = e \wedge \sigma(z) \neq \bot) \vee (\sigma(z) = e \wedge \sigma(y) \neq \bot) \\ \sigma[x \mapsto o] & \text{if } \sigma(y) = o \wedge \sigma(z) = o \\ \sigma[x \mapsto \top] & \text{if } (\sigma(y) = \top \wedge \sigma(z) \neq e, \bot) \vee (\sigma(z) = \top \wedge \sigma(y) \neq e, \bot) \end{cases}$$

Proof. of monotonicity of the above flow function

Assume $\sigma_1 \sqsubseteq \sigma_2$

Since $\sqsubseteq$ is defined point-wise, $\sigma_1(y) \sqsubseteq_{simple} \sigma_2(y)$ and $\sigma_1(z) \sqsubseteq_{simple} \sigma_2(z)$

Case $(\sigma_1(y) = e \wedge \sigma_1(z) \neq \bot) \vee (\sigma_1(z) = e \wedge \sigma_1(y) \neq \bot)$

Since $\sigma_1(y) \sqsubseteq_{simple} \sigma_2(y)$ and $\sigma_1(z) \sqsubseteq_{simple} \sigma_2(z)$, we get

$(\sigma_2(y) = e, \top \wedge \sigma_2(z) \neq \bot) \vee (\sigma_2(z) = e, \top \wedge \sigma_2(y) \neq \bot)$

$$\therefore f_P[\![x := y * z]\!](\sigma_2) = \begin{cases} \sigma_2[x \mapsto e] & \text{if } (\sigma_2(y) = e \wedge \sigma_2(z) \neq \bot) \vee \\ & \quad (\sigma_2(z) = e \wedge \sigma_2(y) \neq \bot) \\ \sigma_2[x \mapsto \top] & \text{otherwise} \end{cases}$$

Since $\sqsubseteq$ is defined point-wise, $e \sqsubseteq_{simple} e$, $e \sqsubseteq_{simple} \top$, and $\sigma_1 \sqsubseteq \sigma_2$, we get

$f_P[\![x := y * z]\!](\sigma_1) = \sigma_1[x \mapsto e] \sqsubseteq f_P[\![x := y * z]\!](\sigma_2)$

[The other cases of this proof (casing on the flow function cases for σ_1) are left as an exercise] □

2.3 Local Soundness

The key to designing a sound analysis is to make sure that the flow functions are locally sound. We can formalize this as a *local soundness* property:

Local Soundness A flow function f is *locally sound* iff $P \vdash c_i \rightsquigarrow c_{i+1}$ and $\alpha(c_i) \sqsubseteq \sigma_{n_i}$ and $f[\![P[n_i]]\!](\sigma_{n_i}) = \sigma_{n_{i+1}}$ implies $\alpha(c_{i+1}) \sqsubseteq \sigma_{n_{i+1}}$

In English: if we take any concrete execution of a program instruction, map the input machine state to the abstract domain using the abstraction function, find that the abstracted input state is described by the analysis input information, and apply the flow function, we should get a result that correctly accounts for what happens if we map the actual concrete output machine state to the abstract domain.

Another way of saying this is that the manipulation of the abstract state done by the flow function of the analysis should reflect the manipulation of the concrete machine state done by the executing instruction.

2.3.1 Proving Local Unsoundness of Parity Analysis

We will specify an input state c_i for the unsound flow function below and show using that input state that the flow function is not locally sound:

$$f_P[\![x := y * z]\!](\sigma) = \sigma[x \mapsto e]$$

Proof. of local unsoundness of the above flow function

Consider $c_i = E, n = \{x \mapsto 0, y \mapsto 3, z \mapsto 5\}, n$

where $P(n) = I = x := y * z$

Then $\sigma_{n_i} = \alpha(E) = \{x \mapsto e, y \mapsto o, z \mapsto o\}$

and $c_{i+1} = E', n' = \{x \mapsto 15, y \mapsto 3, z \mapsto 5\}, n + 1$ by step-arith

Also, $\sigma_{n_{i+1}} = f_P[\![x := y * z]\!](\sigma_{n_i}) = \sigma_{n_i}[x \mapsto e] = \{x \mapsto e, y \mapsto o, z \mapsto o\}$

and $\alpha(c_{i+1}) = \alpha(E') = \{x \mapsto o, y \mapsto o, z \mapsto o\}$

Therefore, since $\sqsubseteq$ is defined point-wise and $o \not\sqsubseteq_{simple} e$, we get

$\alpha(c_{i+1}) = \{x \mapsto o, y \mapsto o, z \mapsto o\} \not\sqsubseteq \{x \mapsto e, y \mapsto o, z \mapsto o\} = \sigma_{n_{i+1}}$ □

2.3.2 Proving Local Soundness of Parity Analysis

We can show that the flow functions for *parity analysis* are locally sound. For brevity, we only show an interesting local soundness proof case for the flow function for the multiplication of two variables assigned to another variables, ie. $x := y * z$; the rest are analogous:

Here is the multiplication flow function for parity analysis:

$$f_P[\![x := y * z]\!](\sigma) = \begin{cases} \sigma[x \mapsto \bot] & \text{if } \sigma(y) = \bot \vee \sigma(z) = \bot \\ \sigma[x \mapsto e] & \text{if } (\sigma(y) = e \wedge \sigma(z) \neq \bot) \vee (\sigma(z) = e \wedge \sigma(y) \neq \bot) \\ \sigma[x \mapsto o] & \text{if } \sigma(y) = o \wedge \sigma(z) = o \\ \sigma[x \mapsto \top] & \text{if } (\sigma(y) = \top \wedge \sigma(z) \neq e, \bot) \vee (\sigma(z) = \top \wedge \sigma(y) \neq e, \bot) \end{cases}$$

Proof. of local soundness of the above flow function
Assume $c_i = E, n$ and $\alpha(E) \sqsubseteq \sigma_{n_i}$
Then $c_{i+1} = E[x \mapsto k], n + 1$ for some k such that $E(y) * E(z) = k$ by rule *step-arith*
Now $\alpha(c_{i+1}) = \alpha(E[x \mapsto k]) = \alpha(E)[x \mapsto \alpha_{simple}(k)]$ by the definitions of α and α_{simple}

Case $k \bmod 2 = 0$
Then $\alpha_{simple}(k) = e$ and $E(y) \bmod 2 = 0 \vee E(z) \bmod 2 = 0$
Thus $(\alpha_{simple}(E(y)) = e \wedge \alpha_{simple}(E(z)) \neq \bot) \vee$
$(\alpha_{simple}(E(z)) = e \wedge \alpha_{simple}(E(y)) \neq \bot)$

Since $\alpha(E) \sqsubseteq \sigma_{n_i}$, we get
$(\alpha_{simple}(E(y)) = e \sqsubseteq_{simple} \sigma_{n_i}(y) \wedge \alpha_{simple}(E(z)) \neq \bot \sqsubseteq_{simple} \sigma_{n_i}(z)) \vee$
$(\alpha_{simple}(E(z)) = e \sqsubseteq_{simple} \sigma_{n_i}(z) \wedge \alpha_{simple}(E(y)) \neq \bot \sqsubseteq_{simple} \sigma_{n_i}(y))$

From this we get $(\sigma_{n_i}(y) = e, \top \wedge \sigma_{n_i}(z) \neq \bot) \vee (\sigma_{n_i}(z) = e, \top \wedge \sigma_{n_i}(y) \neq \bot)$

$$\therefore \sigma_{n_{i+1}} = f_P[\![x := y * z]\!](\sigma_{n_i}) = \begin{cases} \sigma_{n_i}[x \mapsto e] & \text{if } (\sigma_{n_i}(y) = e \wedge \sigma_{n_i}(z) \neq \bot) \vee \\ & \quad (\sigma_{n_i}(z) = e \wedge \sigma_{n_i}(y) \neq \bot) \\ \sigma_{n_i}[x \mapsto \top] & \text{otherwise} \end{cases}$$

Since $\sqsubseteq$ is defined point-wise, $\alpha(E) \sqsubseteq \sigma_{n_i}$, $e \sqsubseteq_{simple} e$, and $e \sqsubseteq_{simple} \top$, we get
$\alpha(c_{i+1}) = \alpha(E)[x \mapsto e] \sqsubseteq \sigma_{n_{i+1}}$

Case $k \bmod 2 \neq 0$
[Similar to the previous case and left up to the reader to prove as an exercise]

□