CSE 231 Midterm

Thursday February 16th, 2017

Name:

Student ID:

Instructions: This is a open-book midterm on which you will work individually. "open-book" means that you have access to all the materials from the CSE 231 class web page.

Power-of-2 Analysis

In this problem, your task will be to design a power-of-2 analysis. At each program point, the analysis computes for each variable X if it's a power of 2.

To start, we will consider a simple language without pointers, and then we will add pointers. In particular, consider the language where statements are taken from the following grammar:

$$\begin{array}{lll} S & ::= & X := C \mid X := Y \mid X := Y * Z \mid X := Y + Z \\ X, Y, Z & ::= & \text{Variable names} \\ C & ::= & \text{Integer constant literals} \end{array}$$

In addition to the above statements, the language also supports regular C-style control constructs, like `if` and `while`, which in our CFG will be represented using:

- `branch` nodes that have one incoming edge and two outgoing edges.
- `merge` nodes that have two incoming edges and one outgoing edge.

To narrow things down, we will make the domain D be $Powerset(Vars \times Z^+)$ where $Vars$ is the set of variables, and Z^+ is the set of non-negative integers. For example, an element of this lattice would look like this:

$$\{(\mathtt{x}, 2), (\mathtt{y}, 3)\}$$

If the above set is computed at a program point, then this would state that $x = 2^2$ and $y = 2^3$ at that program point.

To give you a sense for how the analysis should work, consider the following code snippet, in which we assume the entry dataflow fact is the empty set. The computed information is shown for every program point as an inline comment.

```
void foo(c) {
  // {}
  a := 2
  // {(a,1)}
  b := 4
  // {(a,1), (b,2)}
  if (c == 2) {
    // {(a,1), (b,2)}
    b := b + b
    // {(a,1), (b,3)}
    a := a * b
    // {(a,4), (b,3)}
  } else {
    // {(a,1), (b,2)}
    b := a * b
    // {(a,1), (b,3)}
    b := a + b
    // {(a,1)}
  }
  // {}
}
```

As a sanity check, make sure your analysis computes the above information correctly!

1. Define the lattice over which the dataflow analysis will run. The set D has already been defined above; you should define $\bot$ and $\top$, and then go on to define $\sqcup, \sqcap, \sqsubseteq$.

$\bot =$

$\top =$

$\sqcup =$

$\sqcap =$

$\sqsubseteq =$

Solution

$\bot = \mathit{Vars} \times Z^+ \equiv \{(v, n) \mid v \in \mathit{Vars} \wedge n \in Z^+\}$
$\top = \varnothing$
$\sqcup = \cap$
$\sqcap = \cup$
$\sqsubseteq = \supseteq$

2. How large is your lattice (ie: what is the size of the D set), and what is the height of your lattice? Express your answer in terms of the size of Z^+, written as $|Z^+|$.

Size: ---

Height: ---

Solution

$$\texttt{Size} = 2^{|\mathit{Vars}|\cdot|Z^+|}$$
$$\texttt{Height} = |\mathit{Vars}| \cdot |Z^+|$$

This needs to be consistent with the lattice described above.

Now you will write the flow functions for all the statement forms in the language (including branches and merge statements).

As a hint, here is the definition for the flow function for $X := Y$:

$$F_{X:=Y}(S) \quad = \quad S - \{(X, *)\} \cup \{(X, n) \mid (Y, n) \in S\}$$

Now fill in the definitions for the remaining flow functions:

3. $X := C$:

Solution

$$F_{X:=C}(S) \quad = \quad S - \{(X, *)\} \cup \{(X, n) \mid C = 2^n\}$$

4. $X := Y * Z$:

Solution

$$F_{X:=Y*Z}(IN) = IN - \{(X, *)\} \cup \{(X, n + m) \mid (Y, n) \in IN \wedge (Z, m) \in IN\}$$

5. $X := Y + Z$:

   Hint: make sure that $\{(a, 2)\}\, b := a + a$ gives $(b, 3)$ on output.

Solution

$$F_{X:=Y+Z}(IN) = IN - \{(X, *)\} \cup \{(X, n+1) \mid (Y, n) \in IN \wedge (Z, n) \in IN\}$$

6. `merge` :

Solution

$$F_{\texttt{merge}}(IN_1, IN_2) = IN_1 \cap IN_2$$

7. `branch`:

   Hint: The flow function should have form $F_{\texttt{branch}(\ldots)} = (out_T, out_F)$, where out_T and out_F are the facts computed for the outgoing true and false sides of the branch, respectively.

Solution

$$F_{\texttt{branch}}(IN) = (IN, IN)$$

8. We want to translate multiplication by a power of 2 into a left shift. Fill in the blanks in the following transformation rule:

   If there is a statement of the form $X := Y * Z$, where the incoming set *IN* contains the fact ____(a)____, then transform that statement to ____(b)____.

   Fill in the blanks below:

a	b
----------	----------

Solution

a	b
(Z, n)	$X := Y << n$
(Y, n)	$X := Z << n$
$(Y, n), (Z, m)$	$X := 1 << (n + m)$

We will now add pointers to the original language by extending its grammar as follows:

$$S \quad ::= \quad \ldots \mid X := \&Y \mid X := *Y \mid *X := Y$$

Assume that a pointer is never a power of two. Thus, the flow function for address assignment statements $X := \&Y$ is the following:

$$F_{X:=\&Y}(S) = S - \{(X, *)\}$$

Furthermore, assume that you are given the result of a prior may-point-to analysis. That is, for each program point, you are provided a map *mayPointTo* that maps each variable to the set of variables to which it may point. For example, if $mayPointTo(\texttt{a}) = \{\texttt{a,b}\}$ then `a` may point to `a` or `b`.

Write flow functions for these new statements.

9. $X := *Y$:

Solution

$$F_{X:=*Y}(S) \quad = \quad S - \{(X, *)\} \quad \cup \quad \{(X, n) \mid \forall Z \in mayPointTo(Y).(Z, n) \in S\}$$

10. $*X := Y$:

Solution

$$\begin{array}{rcll} F_{*X:=Y}(S) & = & S & - \quad \{(Z,*) \mid Z \in mayPointTo(X)\} \\ & & & \cup \quad \{(Z,n) \mid (Z,n) \in S \wedge (Y,n) \in S\} \end{array}$$

11. Let's go back to the domain of the analysis. Recall the size you computed for the domain. This is a very large domain. If $|Z^+|$ is infinite, then the domain is infinite. If $|Z^+|$ only represents 64-bit numbers, the domain is finite, but extremely large. Design a new domain for this analysis so that the height is finite and relatively small.

Solution

Version 1 Define a domain for each variable $D_v = \{\bot, \top\} \cup Z^+$. This gives a height of 2. The size of this domain is bounded by the size of $|Z^+|$, which is now finite.

The total domain D can be defined as the mapping: $D \equiv \mathit{Vars} \rightarrow D_v$ or a tuple domain: $D \equiv (D_{v_1}, \ldots D_{v_k})$, where $v_1, \ldots, v_k \in \mathit{Vars}$. The height of the lattice is $2 \cdot |\mathit{Vars}|$.

Version 2 (Partial Credit)

Retrict the exponent domain to $Z_{pow} \equiv \{n \mid n, c \in Z^+ \wedge n = 2^c\}$, whose size is now just 64.