Program Optimization
Exercise Sheet 5
23.11., due 28.11. 12:30

Exercise 1: Interval Analysis
Consider the following program:

1. Perform interval analysis with accelerated widening and narrowing. Make use of the more precise transfer functions $\text{Pos}(e)$ and $\text{Neg}(e)$ which are defined in the slides (slides 329, 330). Use the program point 4 as a loop separator point. Solve using RR-iteration and present the tables.

2. Show that a bounds check can be removed.

Solution:

1. First perform interval analysis with accelerated widening (apply at the pp.4).

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\bot$</td>
<td>${i \mapsto \top, x \mapsto \top}$</td>
<td>${i \mapsto [19, 20], x \mapsto \top}$</td>
<td>${i \mapsto [-\infty, 20], x \mapsto \top}$</td>
</tr>
<tr>
<td>2</td>
<td>$\bot$</td>
<td>${i \mapsto [20, 20], x \mapsto \top}$</td>
<td>${i \mapsto [19, 20], x \mapsto \top}$</td>
<td>${i \mapsto [1, 20], x \mapsto \top}$</td>
</tr>
<tr>
<td>3</td>
<td>$\bot$</td>
<td>${i \mapsto [20, 20], x \mapsto \top}$</td>
<td>${i \mapsto [-\infty, 20], x \mapsto [-\infty, 40]}$</td>
<td>${i \mapsto [-\infty, 20], x \mapsto [-\infty, 40]}$</td>
</tr>
<tr>
<td>4</td>
<td>$\bot$</td>
<td>${i \mapsto [20, 20], x \mapsto [40, 40]}$</td>
<td>${i \mapsto [-\infty, 20], x \mapsto [0, 40]}$</td>
<td>${i \mapsto [-\infty, 20], x \mapsto [0, 40]}$</td>
</tr>
<tr>
<td>5</td>
<td>$\bot$</td>
<td>${i \mapsto [20, 20], x \mapsto [40, 40]}$</td>
<td>${i \mapsto [-\infty, 20], x \mapsto [-\infty, 40]}$</td>
<td>${i \mapsto [-\infty, 20], x \mapsto [-\infty, 40]}$</td>
</tr>
<tr>
<td>6</td>
<td>$\bot$</td>
<td>${i \mapsto [20, 20], x \mapsto [40, 40]}$</td>
<td>$\bot$</td>
<td>${i \mapsto [-\infty, 0], x \mapsto \top}$</td>
</tr>
<tr>
<td>7</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
</tbody>
</table>

Use the fixpoint computed at the previous step as initial state for the iteration with accelerated narrowing.
2. At point 4 we know that $x \in [2, 40]$, so the positive branch will always be taken.

**Exercise 2: Points-to Analysis (alias analysis idea 1)**

Consider the following program:

$$
\begin{array}{c}
\text{0} & \text{x=new();} \\
\text{1} & \text{y=new(); y[0]=3;} \\
\text{2} & \text{x[0]=3;} \\
\text{3} & \text{y[0]=4;} \\
\text{4} & \text{z=x[0];} \\
\text{5} & \text{z=new();}
\end{array}
$$

1. Write down the constraint system for the points-to analysis (slide 374). Use subscripts to access tuple components, e.g., if $P[5] = (D, M)$, then $P[5]_1 = D$ and $P[5]_2 = M$. Start with:

$$
P[0] \equiv (\emptyset, \emptyset)$$

$$
P[1] \equiv (P[0]_1(x \mapsto \{(0, 1)\}), P[0]_2)
$$

2. Define $\sqsubseteq$ for the domain $(\text{Vars} \to 2^{\text{Addr}^\sharp}) \times (\text{Addr}^\sharp \to 2^{\text{Addr}^\sharp})$.

3. Give the least solution to the constraint system.

4. Assume that the analysis for constant propagation was extended in order to deal with pointers and memory. Then, for this particular program, does the pointer analysis help us in order to perform constant propagation? Give a precise answer why or why not pointer analysis supports constant propagation.

**Solution:**

1. Write down the constraint system for the points-to analysis (slide 374). Use subscripts to access tuple components, e.g., if $P[5] = (D, M)$, then $P[5]_1 = D$ and $P[5]_2 = M$. Start with:

$$
P[0] \equiv (\emptyset, \emptyset)$$

$$
P[1] \equiv (P[0]_1(x \mapsto \{(0, 1)\}), P[0]_2)
$$

2. Define $\sqsubseteq$ for the domain $(\text{Vars} \to 2^{\text{Addr}^\sharp}) \times (\text{Addr}^\sharp \to 2^{\text{Addr}^\sharp})$.

3. Give the least solution to the constraint system.

4. Assume that the analysis for constant propagation was extended in order to deal with pointers and memory. Then, for this particular program, does the pointer analysis help us in order to perform constant propagation? Give a precise answer why or why not pointer analysis supports constant propagation.

**Solution:**

1. Write down the constraint system for the points-to analysis (slide 374). Use subscripts to access tuple components, e.g., if $P[5] = (D, M)$, then $P[5]_1 = D$ and $P[5]_2 = M$. Start with:

$$
P[0] \equiv (\emptyset, \emptyset)$$

$$
P[1] \equiv (P[0]_1(x \mapsto \{(0, 1)\}), P[0]_2)
$$

2. $A \sqsubseteq B \iff \forall x \in \text{Var}, a \in \text{Addr}^\sharp. A_1(x) \subseteq B_1(x) \land A_2(a) \subseteq B_2(a)$.}

2
3.

\[ P[0] = (\emptyset, \emptyset) \]
\[ P[1] = (\{x \mapsto \{(0, 1)\}\}, \emptyset) \]
\[ P[2] = (\{x \mapsto \{(0, 1)\}, y \mapsto \{(1, 2)\}\}, \emptyset) = P[3] = P[4] \]
\[ P[5] = (\{x \mapsto \{(0, 1)\}, y \mapsto \{(1, 2)\}, z \mapsto \emptyset\}, \emptyset) \]

4. With the help of the pointer analysis we might find out, that two variables do not alias. In such a case the analysis of constant propagation can exploit this information. For the particular example above we find out that \(x\) and \(y\) do not alias each other. Therefore, while performing constant propagation we might safely set \(z\) to 3.

Exercise 3: Flow insensitive alias analysis (alias analysis idea 2)
Consider the following program.

1. Write down the constraint system for the flow insensitive alias analysis (slide 378).
2. Give the least solution to the constraint system.

Solution:

1.

\[ P[a] \supseteq \{(1, 2)\} \]
\[ P[xs] \supseteq \{(2, 3)\} \]
\[ P[ys] \supseteq \{(3, 4)\} \]
\[ P[t] \supseteq \bigcup \{P[f] \mid f \in P[a]\} \]
\[ P[f] \supseteq (f \in P[ys]) ? P[t] : \emptyset \quad \forall f \in Addr^2 \]
\[ P[f] \supseteq (f \in P[ys]) ? P[t] : \emptyset \quad \forall f \in Addr^2 \]

2.

\[ P[a] = \{(1, 2)\} \]
\[ P[xs] = \{(2, 3)\} \]
\[ P[ys] = \{(3, 4)\} \]
\[ P[t] = \emptyset \]
\[ P[(1, 2)] = \emptyset \]
\[ P[(2, 3)] = \emptyset \]
\[ P[(3, 4)] = \emptyset \]
Exercise 4: Partitioning alias analysis (alias analysis idea 3)

Consider the following program (same as in the previous exercise).

1. Perform the analysis (slide 383) to find out aliasing information.
2. Using your results, show that the second load from \( a[i] \) is superfluous.

Solution:

1. For any other edge \((_, lab, _)_\) the following holds \(J_{lab}^\pi = \pi\).

2. The interesting part from the code is \( t = a[i]; \ x = a[i]; \ t = a[i]; \).

   The second load would not be superfluous in the case that \( t \) would not contain the value \( a[i] \) (as it is not dead).

   One possibility is that the index is different. This is not the case, as we can syntactically verify that there is no writing to the variable \( i \).

   The other possibility to contain some other value (in the semantics, defined in the lecture) would be if the assignment \( x = a[i] \) would modify \( a[i] \).

   This is not the case, however, as \( a \) and \( xs \) are not aliases (they are not in the same partition in the solution).