Homework 2: Overview of Program Analysis

Due: problems 1–2 and 4–7, January 30, 2008; remaining problems, February 11, 2008.

In this homework we will get a bit more into the different kinds of program analysis. You will also learn how to do calculational style proofs.

Turn in (on WebCT) your answers as either plain text files with suffix `.txt`, MS word files with suffix `.doc`, or PDF files with suffix `.pdf`. (We prefer to have a PDF file if that is easy for you.) Please don't put any spaces in your file names!

If you wish, you can work in groups, although I don't recommend trying to do proofs in groups. Note that if you do work together, you must carefully follow the course grading policies.

Sections 1.1-1.3: Reaching Definitions and Data Flow

Read sections 1.1-1.3 of our textbook: *Principles of Programming Analysis* by Flemming Nielson, Hanne Riis Nielson, and Chris Hankin [5].

1. (10 points) [Concepts]

   Describe a question that an optimizing compiler might want to answer, and a way to answer it using information from the Reaching Definitions analysis. An example mentioned in class is the following question. What is the set of variables that may be assigned in a program $S_\star$? This can be answered using the set of reaching definitions from the exits of all final labels (see section 2.1 of the text) in a program $S_\star$, as follows: $\{x \mid (x,\ell) \in \mathsf{RD}_{exit}(\ell'), \ell' \in final(S_\star)), \ell \neq ?\}$. (This might be useful for scheduling memory writes, or for reordering statements.)

   Your task is to give another example.

2. (10 points) [Concepts] [Soundness]

   Consider again your answer to the previous problem. Let us call the question described in your previous answer Q. Give one example of a statement (program) S and an answer to question Q that is incorrect for S. Explain how such an incorrect answer for Q could only be derived from an unsafe answer to the Reaching Definitions analysis for your statement S.

   For example, if Q is the example described in the previous problem, consider the statement: $[\mathtt{y} := 5021]^1$. If the answer to Q was, incorrectly, that the set of variables assigned in that statement is $\{\}$, it could only be from an unsafe answer to the Reaching Definitions analysis, since the set $\{x \mid (x,\ell) \in \mathsf{RD}_{exit}(\ell'), \ell' \in final(S_\star)), \ell \neq ?\}$ could only be empty if $\mathsf{RD}_{exit}(1) = \{(\mathtt{y}, ?)\}$, which is unsafe because $(\mathtt{y}, 1)$ can reach the exit of statement 1.

   Your task is to explain the same kind of reasoning for your answer to the previous question.

3. (suggested practice) Do exercise 1.1 in the text.

4. (15 points; extra credit) [Concepts] What if we added arrays to the language? Precisely describe how that would change the Reaching Definitions and give an example.

5. (15 points; extra credit) [Concepts] [Quality]

   Give an example of a single language change that would make the Reaching Definitions analysis less precise? Describe the syntax and semantics of this language change briefly, and then describe how it affects the Reaching Definitions analysis, showing how it makes it less precise.

Section 1.4: Constraint Based Analysis

Read sections 1.4 of our textbook: *Principles of Programming Analysis* [5].

6. (20 points; extra credit) [Concepts] [Soundness]

   Do the first part of exercise 1.2, showing that the solution in section 1.4 is a solution to the constraints. (For another 10 points extra credit, you can show that it's also a least solution.)

   Use the calculational style for your demonstrations, as explained in the next section below.

Section 1.5: Abstract Interpretation

Read section 1.5 of our textbook: *Principles of Programming Analysis* [5].

For the proofs in this section and in the rest of the course, you must use the "calculational" style [1, 2, 4, 3] that we will demonstrate. (Gries's article can be obtained on campus at the URL: `http://doi.acm.org/10.1145/102868.102870`.) Macros that I use to typeset such calculations in LaTeX are available from `http://www.eecs.ucf.edu/~leavens/COP5021/latex/calculation.tex`.

You are required to use this style for the proofs in this homework. Note that you don't have to have a beautifully typeset proof. However, this is *much* easier to do on a computer than by hand, so you are urged to develop your proofs on a computer, so that you can copy unchanged parts of a formula to the next step.

To explain this better, here are a few examples of this style. A simple example that proves an approximation by use of definitions is the following.

Lemma 1 *Suppose F_j is monotonic, $\vec{RD} = (RD_1, \ldots, RD_j, \ldots, RD_{12})$, and that $\vec{RD'}$ is defined as $(RD_1, \ldots, F_j(\vec{RD}), \ldots, RD_{12})$. Then $\vec{RD} \sqsubseteq \vec{RD'}$.*

Proof: Assuming the hypothesis, we calculate as follows.

$$
\begin{array}{ll}
 & \vec{RD} \\
= & \quad \langle \text{by definition of } \vec{RD} \rangle \\
 & (RD_1, \ldots, RD_j, \ldots, RD_{12}) \\
\sqsubseteq & \quad \langle \text{by } F_j \text{ is monotonic} \rangle \\
 & (RD_1, \ldots, F_j(\vec{RD}), \ldots, RD_{12}) \\
= & \quad \langle \text{by definition of } \vec{RD'} \rangle \\
 & \vec{RD'} \ \blacksquare
\end{array}
$$

The following is a slightly more advanced example of the calculation proof style. It features a calculation done using reverse implications, so as to simplify from the more complex side in the direction of the proof (and avoid "pulling rabbits out of the hat" [2]). It also indicates what is to be changed, by enclosing that subformula in "quotation marks", as an aid to the reader.

Lemma 2 *Suppose $\vec{RD} = (RD_1, \ldots, RD_j, \ldots, RD_{12})$, $\vec{RD'} = (RD_1, \ldots, F_j(\vec{RD}), \ldots, RD_{12})$, and F is monotonic. If $F^n(\vec{\emptyset}) = F^{n+1}(\vec{\emptyset})$ and $F(\vec{RD}) \sqsubseteq F^n(\vec{\emptyset})$, then $F(\vec{RD'}) \sqsubseteq F^n(\vec{\emptyset})$.*

Proof: Assume the hypothesis. Then we calculate, starting from the desired result, as follows.

$$
\begin{array}{ll}
 & F(\vec{RD'}) \sqsubseteq \text{“}F^n(\vec{\emptyset})\text{”} \\
= & \quad \langle \text{by assumption } F^n(\vec{\emptyset}) = F^{n+1}(\vec{\emptyset}) \rangle \\
 & F(\vec{RD'}) \sqsubseteq \text{“}F^{n+1}(\vec{\emptyset})\text{”} \\
= & \quad \langle \text{by definition of iteration of a function} \rangle \\
 & F(\vec{RD'}) \sqsubseteq F(F^n(\vec{\emptyset})) \\
\Leftarrow & \quad \langle \text{by monotonicity of } F \rangle \\
 & \vec{RD'} \sqsubseteq F^n(\vec{\emptyset})
\end{array}
$$

$\Leftarrow$ $\langle$by transitivity$\rangle$
$\vec{RD'} \sqsubseteq F(\vec{RD}) \wedge F(\vec{RD}) \sqsubseteq F^n(\vec{\emptyset})$
$=$ $\langle$by assumption $F(\vec{RD}) \sqsubseteq F^n(\vec{\emptyset})\rangle$
$\vec{RD'} \sqsubseteq F(\vec{RD})$
$\Leftarrow$ $\langle$by Lemma 7 (not shown here)$\rangle$
$\vec{RD} \sqsubseteq F(\vec{RD})$
$=$ $\langle$by F is monotonic$\rangle$
$true$ ∎

7. (15 points) [Concepts]

   Do exercise 1.3 in the text. Note that α and γ are considered to be functions from sets to sets, as in section 1.5.

8. (80 points) [Concepts]

   Do exercise 1.4 in the text; however to avoid extra work, for the first part of this exercise, you only have to show that

   $$\alpha(G_j(\gamma(\mathsf{RD}_1), \ldots, \gamma(\mathsf{RD}_{12}))) \subseteq F_j(\mathsf{RD}_1, \ldots, \mathsf{RD}_{12})$$

   for $j = 4, 5, 6$, i.e., for $G_4 = G_{\mathit{exit}(2)}$, $G_5 = G_{\mathit{entry}(3)}$, and $G_6 = G_{\mathit{exit}(3)}$.

For the sake of clarity, note the following definitions.

F is defined by $F(\vec{\mathsf{RD}}) = (F_1(\vec{\mathsf{RD}}), \ldots, F_{12}(\vec{\mathsf{RD}}))$, where $\vec{\mathsf{RD}} = (\mathsf{RD}_1, \ldots, \mathsf{RD}_{12})$ and the F_j are defined, as in Section 1.3, as follows.

$$
\begin{array}{lcll}
F_{\mathit{entry}(1)}(\vec{\mathsf{RD}}) & = & F_1(\vec{\mathsf{RD}}) & = \{(\mathtt{x}, ?), (\mathtt{y}, ?), (\mathtt{z}, ?)\} \\
F_{\mathit{exit}(1)}(\vec{\mathsf{RD}}) & = & F_2(\vec{\mathsf{RD}}) & = (\mathsf{RD}_1 \setminus \{(\mathtt{y}, \ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathtt{y}, 1)\} \\
F_{\mathit{entry}(2)}(\vec{\mathsf{RD}}) & = & F_3(\vec{\mathsf{RD}}) & = \mathsf{RD}_2 \\
F_{\mathit{exit}(2)}(\vec{\mathsf{RD}}) & = & F_4(\vec{\mathsf{RD}}) & = (\mathsf{RD}_3 \setminus \{(\mathtt{z}, \ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathtt{z}, 2)\} \\
F_{\mathit{entry}(3)}(\vec{\mathsf{RD}}) & = & F_5(\vec{\mathsf{RD}}) & = \mathsf{RD}_4 \cup \mathsf{RD}_{10} \\
F_{\mathit{exit}(3)}(\vec{\mathsf{RD}}) & = & F_6(\vec{\mathsf{RD}}) & = \mathsf{RD}_5 \\
F_{\mathit{entry}(4)}(\vec{\mathsf{RD}}) & = & F_7(\vec{\mathsf{RD}}) & = \mathsf{RD}_6 \\
F_{\mathit{exit}(4)}(\vec{\mathsf{RD}}) & = & F_8(\vec{\mathsf{RD}}) & = (\mathsf{RD}_7 \setminus \{(\mathtt{z}, \ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathtt{z}, 4)\} \\
F_{\mathit{entry}(5)}(\vec{\mathsf{RD}}) & = & F_9(\vec{\mathsf{RD}}) & = \mathsf{RD}_8 \\
F_{\mathit{exit}(5)}(\vec{\mathsf{RD}}) & = & F_{10}(\vec{\mathsf{RD}}) & = (\mathsf{RD}_9 \setminus \{(\mathtt{y}, \ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathtt{y}, 5)\} \\
F_{\mathit{entry}(6)}(\vec{\mathsf{RD}}) & = & F_{11}(\vec{\mathsf{RD}}) & = \mathsf{RD}_6 \\
F_{\mathit{exit}(6)}(\vec{\mathsf{RD}}) & = & F_{12}(\vec{\mathsf{RD}}) & = (\mathsf{RD}_1 1 \setminus \{(\mathtt{y}, \ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathtt{y}, 6)\}
\end{array}
$$

G is defined by $G(\vec{\mathsf{CS}}) = (G_1(\vec{\mathsf{CS}}), \ldots, G_{12}(\vec{\mathsf{CS}}))$, where for $\vec{\mathsf{CS}} = (\mathsf{CS}_1, \ldots, \mathsf{CS}_{12})$, the G_j are defined as in Section 1.5.

$$
\begin{array}{lcll}
G_{\mathit{entry}(1)}(\vec{\mathsf{CS}}) & = & G_1(\vec{\mathsf{CS}}) & = \{((\mathtt{x},?),(\mathtt{y},?),(\mathtt{z},?))\} \\
G_{\mathit{exit}(1)}(\vec{\mathsf{CS}}) & = & G_2(\vec{\mathsf{CS}}) & = \{tr : (\mathtt{y},1) \mid tr \in \mathsf{CS}_1\} \\
G_{\mathit{entry}(2)}(\vec{\mathsf{CS}}) & = & G_3(\vec{\mathsf{CS}}) & = \mathsf{CS}_2 \\
G_{\mathit{exit}(2)}(\vec{\mathsf{CS}}) & = & G_4(\vec{\mathsf{CS}}) & = \{tr : (\mathtt{z},2) \mid tr \in \mathsf{CS}_3\} \\
G_{\mathit{entry}(3)}(\vec{\mathsf{CS}}) & = & G_5(\vec{\mathsf{CS}}) & = \mathsf{CS}_4 \cup \mathsf{CS}_{10} \\
G_{\mathit{exit}(3)}(\vec{\mathsf{CS}}) & = & G_6(\vec{\mathsf{CS}}) & = \mathsf{CS}_5 \\
G_{\mathit{entry}(4)}(\vec{\mathsf{CS}}) & = & G_7(\vec{\mathsf{CS}}) & = \mathsf{CS}_6 \\
G_{\mathit{exit}(4)}(\vec{\mathsf{CS}}) & = & G_8(\vec{\mathsf{CS}}) & = \{tr : (\mathtt{z},4) \mid tr \in \mathsf{CS}_7\} \\
G_{\mathit{entry}(5)}(\vec{\mathsf{CS}}) & = & G_9(\vec{\mathsf{CS}}) & = \mathsf{CS}_8 \\
G_{\mathit{exit}(5)}(\vec{\mathsf{CS}}) & = & G_{10}(\vec{\mathsf{CS}}) & = \{tr : (\mathtt{y},5) \mid tr \in \mathsf{CS}_9\} \\
G_{\mathit{entry}(6)}(\vec{\mathsf{CS}}) & = & G_{11}(\vec{\mathsf{CS}}) & = \mathsf{CS}_6 \\
G_{\mathit{exit}(6)}(\vec{\mathsf{CS}}) & = & G_{12}(\vec{\mathsf{CS}}) & = \{tr : (\mathtt{y},6) \mid tr \in \mathsf{CS}_{11}\}
\end{array}
$$

Furthermore, the following definitions are given in Section 1.5.

$$
\begin{array}{lcl}
\alpha(\mathsf{CS}) & = & \{(x, \mathsf{SRD}(tr)(x)) \mid x \in \mathsf{DOM}(tr) \wedge tr \in \mathsf{CS}\} \\
\gamma(\mathsf{RD}) & = & \{tr \mid (\forall x \in \mathsf{DOM}(tr) :: (x, \mathsf{SRD}(tr)(x)) \in \mathsf{RD})\} \\
\vec{\alpha}(\mathsf{CS}_1, \ldots, \mathsf{CS}_{12}) & = & (\alpha(\mathsf{CS}_1), \ldots, \alpha(\mathsf{CS}_{12})) \\
\vec{\gamma}(\mathsf{RD}_1, \ldots, \mathsf{RD}_{12}) & = & (\gamma(\mathsf{RD}_1), \ldots, \gamma(\mathsf{RD}_{12}))
\end{array}
$$

Section 1.6: Type and Effect Systems

Read section 1.6 of our textbook: *Principles of Programming Analysis* [5].

9. (10 points) [Concepts]

   Using the type system specified in Table 1.2, what is the least type of the following program?

   ```
   [q := 0]1; if [r > q]2 then [x := r]3 else [y := r]4
   ```

10. (10 points) [Concepts]

    Using the type system specified in Table 1.3, what is the least type of the following program?

    ```
    [q := 0]1; if [r > q]2 then [x := r]3 else [y := r]4
    ```

Section 1.7: Algorithms

Read section 1.7 of our textbook: *Principles of Programming Analysis* [5].

11. (suggested practice) Do exercise 1.7.
12. (suggested practice) Do exercise 1.8.

Section 1.8: Transformations

Read section 1.8 of our textbook: *Principles of Programming Analysis* [5].

13. (suggested practice) Do exercise 1.9.
14. (15 points; extra credit) [Concepts]

    Do exercise 1.10.

References

[1] Ralph-Johan Back and Joakim von Wright. *Refinement Calculus: A Systematic Introduction*. Graduate Texts in Computer Science. Springer-Verlag, 1998.

[2] Edsger W. Dijkstra and Carel S. Scholten. *Predicate Calculus and program semantics*. Springer-Verlag, NY, 1990.

[3] David Gries. Teaching calculation and discrimination: A more effective curriculum. *Communications of the ACM*, 34(3):44–55, March 1991.

[4] David Gries and Fred B. Schneider. *A Logical Approach to Discrete Math*. Texts and Monographs in Computer Science. Springer-Verlag, New York, NY, 1994.

[5] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer-Verlag, second printing edition, 2005.