Homework 4: A Project, or More Data Flow Analysis

Due: April 14, 2008.

In this homework you will learn more about data flow analysis or apply what you know to a research project.

Turn in on paper your answers, which have been typed on a computer (no handwritten answers, please).

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.

Do either the project in the first section below, or the more usual homework problems from the second section.

1 Project

Note that if you do a project, you do not need to do the problems in section 2 below.

In this problem you will define a static analysis question and a static analysis to solve it, and demonstrate that on three example programs. Note that as you work this problem, all three elements may change. That is, you may find you want to have the analysis ask a different question, or you may decide that you have to change the formulation of the analysis, or that you have to change the examples. All are flexible. (That's research!)

I would prefer that you did not just use a known analysis from someone else's work. However, it is okay if your analysis has been defined or implemented by others. But remember that you are required to be scholarly in citing related work that you consult. In particular you must carefully cite and quote any ideas or words you take from other works.

1. (30 points) [Concepts]

   Describe a static analysis question related to your research or research you are interested in pursuing. This question should be of the form that will enable you to prove something about programs (e.g., that they (may) have certain errors or (may) be free of certain errors) or a form that will allow you to transform (optimize) such programs. If you are not sure what to do, I recommend trying to verify the correctness of some small programs that are important to your field.

   You should give a statement of the problem like those seen in the textbook. Such a statement should be independent of the analysis formalization and should specify the correctness of the analysis. It doesn't have to be especially formal, but it must be clear. Teach the reader what it means to answer your analysis question and use examples.

   Finally you must explain the importance and significance of your question for your research field. Why is it interesting? How would it apply in practice?

   (Note that the problem and it's use don't have to be extremely interesting, as you have to be able to solve the problem. So it may pay to start small and work up to something more interesting if you have time. It's okay to simplify things.)

2. (30 points) [Concepts]

   Give three (3) example programs that someone in your field would be able to understand. These should demonstrate a range of the different kinds of statements you wish to handle (e.g., at least loops and conditionals).

   Explain what each of the programs does for some sample inputs.

   Also give the desired output of your analysis for these programs in tabular form (as in Table 1.1), and explain any interesting parts of the table. This will further help the reader understand what you are intending to do with your analysis.

Give the operational semantics of any primitives or statements that are not in the book's WHILE language that you use in your examples. To make this easier, it's best if you stick as close to the WHILE language as you can.

(Note remember to keep things simple, you are advised to choose examples that don't involve functions or objects, to avoid complications in the semantics.)

3. (20 points) [Concepts]

   Formalize your analysis as an instance of a monotone framework (see section 2.3) or as an abstract interpretation (see chapter 4). Say which kind of formalization you are giving. Explain all your notation that is not standard.

   It is fine to use techniques that we have not yet covered in the book. You may want to consult other sections of the textbook or research papers. (Ask us if you need help finding pointers to relevant literature.) Be sure to cite all sources you use in formulating your analysis.

4. (40 points) [Concepts] [Calculate]

   Either (a) implement your analysis using TVLA [5], JastAdd [3], or some other computer system and test it on your 3 examples, or (b) write out the improvement function (F) for each of your 3 example programs and perform Chaotic iteration (as in homework 3 problems 2 and 3) on each of your example programs. This will test whether your formalization works as intended.

   You may find in doing this step that you wish to adjust earlier parts. Be sure to change all the parts of your solution to this homework together to keep them consistent.

5. (60 points; extra credit) [Concepts] [Calculate]

   Prove that your analysis is correct. (Hint: this may be easy if you derive it from a collecting semantics using abstract interpretation.)

2 Textbook Problems

Note: you should only do these problems if you do not want to do a project as described in section 1 above.

Section 2.3: Monotone Frameworks

Read section 2.3 of our textbook: *Principles of Program Analysis* [7].

1. (40 points) [Concepts] [Calculate]

   Do **just one** of the following problems. You pick which one.

   (a) Do exercise 2.9 (bit vector frameworks and distributive frameworks).

   (b) Do exercise 2.14 (detection of signs analysis).

2. (50 points; extra credit) [Concepts]

   Do exercise 2.11.

Section 2.5: Interprocedural Analysis

Read section 2.5 of our textbook: *Principles of Program Analysis* [7].

3. (40 points) [Concepts]

   Do **just one** of the following two problems. You pick one.

   (a) Read one paper on interprocedural analysis. You must pick either the paper by Gotsman, Berdine, and Cook [4] or the paper by Rinetzky, Sagiv, and Yahav [8] (see the references at the end of this homework). After reading this paper write up a report on it. Your report must address the following points.

i. What are the goals of the paper? What are the key technical ideas that are used to achieve those goals. (Explain the ideas in your own words.)

ii. In what ways is it similar to the analysis in sections 2.5 to 2.6 of our textbook [7]. In what ways is it different?

iii. Give an example program which is different than a program given in the paper, but still in the language that the paper treats. Explain how the analysis given in the paper works on the example, and show the results of the analysis for that example.

In your report be sure to properly attribute all direct quotations from the paper (or other sources). Use quotation marks (“ and ”) around all direct quotations from the paper of any length and give page numbers. Your report should *not* just consist of quotations, however, as I want you to digest the material.

(b) Do exercise 2.18 (formulate interprocedural analysis). Since we did the Reaching Definitions analysis in class, you are not allowed to choose that, but must pick one of the other three classical analyses from section 2.1.

Besides presenting your formulation of the analysis, you must also work an example to demonstrate that your analysis makes sense.

4. (20 points; extra credit) Do exercise 2.19.

5. (20 points; extra credit) Do exercise 2.20.

Section 2.6: Shape Analysis

Read section 2.6 of our textbook: *Principles of Program Analysis* [7].

6. (40 points) [Concepts] Read one paper on shape analysis. You must pick either the paper by Balaban, Pnueli, and Zuck [1], Distefano, O’Hearn, and Yang [2], or Manevich *et al.* [6] (see the references at the end of this homework). After reading this paper write up a report on it. Your report must address the following points.

   (a) What are the goals of the paper? What are the key technical ideas that are used to achieve those goals. (Explain the ideas in your own words.)

   (b) In what ways is it similar to the analysis in sections 2.5 to 2.6 of our textbook [7]. In what ways is it different?

   (c) Give an example program which is different than a program given in the paper, but still in the language that the paper treats. Explain how the analysis given in the paper works on the example, and show the results of the analysis for that example.

   In your report be sure to properly attribute all direct quotations from the paper (or other sources). Use quotation marks (“ and ”) around all direct quotations from the paper of any length and give page numbers. Your report should *not* just consist of quotations, however, as I want you to digest the material.

7. (60 points; extra credit) Do exercise 2.24 (operational semantics with garbage collection).

8. (60 points; extra credit) Do exercise 2.25 (summaries based on allocation sites).

References

[1] Ittai Balaban, Amir Pnueli, and Lenore D. Zuck. Shape analysis by predicate abstraction. In *Verification, Model Checking, and Abstract Interpretation*, volume 3385 of *Lecture Notes in Computer Science*, pages 164–180, Berlin, 2005. Springer-Verlag.

[2] Dino Distefano, Peter W. O'Hearn, and Hongseok Yang. A local shape analysis based on separation logic. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *Lecture Notes in Computer Science*, pages 287–302, Berlin, 2006. Springer-Verlag.

[3] Torbjörn Ekman and Görel Hedin. The jastadd system — modular extensible compiler construction. *Science of Computer Programming*, 69(1-3):14–26, 2007.

[4] Alexey Gotsman, Josh Berdine, and Byron Cook. Interprocedural shape analysis with separated heap abstractions. In *Static Analysis*, volume 4134 of *Lecture Notes in Computer Science*, pages 240–260, Berlin, 2006. Springer-Verlag.

[5] Tal Lev-Ami and Shmuel Sagiv. TVLA: A system for implementing static analyses. In Jens Palsberg, editor, *Static Analysis, 7th International Symposium, SAS 2000, Proceedings*, volume 1824 of *Lecture Notes in Computer Science*, pages 280–301. Springer, 2000.

[6] Roman Manevich, E. Yahav, G. Ramalingam, and Mooly Sagiv. Predicate abstraction and canonical abstraction for singly-linked lists. In *Verification, Model Checking, and Abstract Interpretation*, volume 3385 of *Lecture Notes in Computer Science*, pages 181–198, Berlin, 2005. Springer-Verlag.

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

[8] Noam Rinetzky, Mooly Sagiv, and Eran Yahav. Interprocedural shape analysis for cutpoint-free programs. In *Static Analysis*, volume 3672 of *Lecture Notes in Computer Science*, pages 284–302. Springer-Verlag, 2005.