Homework 3: Data Flow Analysis

Due: problems 1–9, March 6, 2006; problems 10-14, March 27, 2006.
In this homework you will learn more about data flow analysis.
If you wish, you can work in groups.

Section 1.7: Algorithms

Read section 1.7 of our textbook: *Principles of Programming Analysis* [4].

1. (20 points) Do exercise 1.6. In your proof use a calculational style (as in the handouts we provided in class [1, Section 4.2] [2, Chapter 4], see also Gries's article in *CACM* [3]), in which you justify each step.
2. (20 points) Do exercise 1.7. Again use the calculational style for any calculations or proofs involved.
3. (25 points) Do exercise 1.8. Again use the calculational style for any calculations or proofs involved.

Section 2.1: Intraprocedural Analysis

Read section 2.1 of our textbook: *Principles of Programming Analysis* [4].

4. This problem is about finding examples to illustrate what starting values is needed to compute a solution by iteration, for the data flow analyses in this section.
   (a) (10 points) According to the book, when using Chaotic iteration to solve for a solution of the Available Expressions analysis (in Section 2.1.1), one should start with a tuple in which each element is $\mathbf{AExp}_\star$. To see why this is necessary, create an example program in the WHILE language for which the Chaotic iteration of the function that represents that program's Available Expressions analysis does not get the right result, if the iteration starts with $\vec{\emptyset}$. (Hint: the next part of this problem will be easiest if your example is very small.)
   (b) (10 points) Show how, the Chaotic iteration for your example gives the right result for the Available Expressions analysis, if you start the iteration with a tuple in which each element is $\mathbf{AExp}_\star$.
   (c) (10 points) Similarly, give an example program in the WHILE language in which the chaotic iteration for the Very Busy Expressions analysis (of section 2.1.3) does not give the right result, if the iteration starts with $\vec{\emptyset}$.
   (d) (10 points) Show how, the Chaotic iteration for your example gives the right result for the Very Busy Expressions analysis, if you start the iteration with a tuple in which each element is $\mathbf{AExp}_\star$.
   (e) (10 points) What property of an analysis, in general, determines what starting value is appropriate for iterations used to find a solution?
   (f) (10 points; extra credit) Characterize the class of programs (perhaps syntactically) for which the starting value of an iteration makes no difference.
5. (suggested practice) Do exercise 2.1.
6. (15 points) Do exercise 2.2.

7. (20 points) Do exercise 2.3.

8. (25 points) Do exercise 2.4.

9. (20 points) Do the first part of Mini Project 2.1. Use the calculational style for your proof.

   Note added after this was assigned: this should definitely have had more points. It may be best to skip this or give a non-calculational proof to avoid taking too much time on it.

Section 2.2: Theoretical Properties

Read section 2.2 of our textbook: *Principles of Programming Analysis* [4].

10. (10 points) Do a proof of part (iv) of lemma 2.14 using the calculational style in your proof.

11. Write additions to table 2.6 to handle the following new pieces of syntax. (Hint, all of these can be handled without changing the configurations, and in particular you do not need to introduce "errors" or $\perp$ to the domain **State**.

    (a) (5 points) A `break` statement. This breaks out of the closest surrounding `while` loop. You can assume that `break` statements only occur inside `while` loops.

    (b) (5 points) A `return` statement, which takes no arguments (as in `void` methods in Java) and simply halts execution without changing the state.

    (c) (10 points) A `throw` statement, and a `try-catch` statement of the form `try` S_1 `catch` S_2, where S_1 and S_2 are statements. where you can assume that all `throw` statements occur inside a `try-catch` statement, and in which a `throw` statement jumps (without changing the state) to the `catch` block of closest surrounding `try-catch` statement.

    (d) (5 points) Nondeterministic (demonic) choice statements of the form $S_1 \,[]\, S_2$, whose meaning is to execute either S_1 or S_2. (Hint: use 2 rules.)

    (e) (5 points) Parallel execution statements of the form $S_1 \,||\, S_2$, whose meaning is to execute both S_1 and S_2 in an interleaved fashion.

    (f) (10 points) Assume statements of the form `assume` b, whose meaning is to do nothing if b is true, but to refuse to execute if b is not true.

12. (30 points; extra credit) Which of the additions to the WHILE language in the previous problem, if any, invalidate Lemma 2.14? Give a counterexample for any parts invalidated, and use the calculational style to show how the counterexample works out. Conversely, prove the parts not invalidated by various additions using the calculational style.

13. (30 points) Do the second part of Mini Project 2.1. Use the calculational style for your proofs.

14. (60 points) Do Mini Project 2.2. Use the calculational style for your proofs.

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