Homework 4
Due Date: November 29, 2000

Somesh Jha
Computer Science Department,
University of Wisconsin,
Madison, WI 53706.

November 15, 2000

Question 1 (30 points):
Part a: Express the relations given in Figure 1 using the relations given in class and various operators (of course!). Please give a short justification for your answer.
Part b: Let CousinsFatherSide(n), CousinsMotherSide(n), and Cousins(n) be the set of cousins that are separated by n-steps on the father's side, mother's side, and father or mother's side. Write a recursive expression for these relations, i.e., express the relations with parameter n in terms of relations with parameter $n-1$ and the the basic relations introduced in the

FatherInLaw : People $\leftrightarrow$ Males
SisterInLaw : People $\leftrightarrow$ Females
FirstCousinsFatherSide : People $\leftrightarrow$ People
FirstCousinsMotherSide : People $\leftrightarrow$ People
FirstCousins : People $\leftrightarrow$ People
SecondCousinsFatherSide : People $\leftrightarrow$ People
SecondCousinsMotherSide : People $\leftrightarrow$ People
SecondCousins : People $\leftrightarrow$ People

Figure 1: Some relations

class.

Question 2 (20 points): Establish whether the following equations between the relations are true or not. Please justify your answer. You should enter these equations in *ladybug* and play around with them. Assume that the relations have appropriate types.

$$\begin{array}{l} (P;Q)^\top \;=\; Q^\top;P^\top \\ (P \subseteq Q) \Rightarrow (P;P \subseteq Q) \\ (Q \neq \emptyset) \Rightarrow (\mathtt{Un};Q;\mathtt{Un} \;=\; \mathtt{Un}) \end{array}$$

`Note:` The symbol $\emptyset$ denotes the empty relation.

Question 3 (75 points): In this question you will write a specification for the problem described. Our aim is to model a library. There are two types `PERSONS` and `BOOKS`. There is a function `issued` : `BOOKS` $\rightarrow$ `PERSONS` and a set `Library` : *Set* `BOOKS`. If book b is issued to a person p, then `issued`$(b) = p$. `Library` represents the set of books in the library. There is also a set of books on reserve given by the set `Reserve` : *Set* `BOOKS`. The operations are:

- *Issue a book*
  This operation issues a book to a person. A book on reserve or already issued cannot be issued.
- *Return a book*
  This operation models the act of a person returning a book.
- *Adding a book to the library*
  This operations models a new book being added to the library.
- *Putting a book on reserve*
  This operations models a book being put on reserve. A book which is currently issued cannot be put on reserve.
- *Taking a book off reserve*
  This operation models a book being taken off reserve.

The claims (or assertions in Alloy) are:

- Issuing a book and then returning it results in the same state.
- If a book is on reserve, it should never be issued. This claim should be true before and after every operation.

Part a: Write a mathematical description for all the operations and claims and explain your answer.
Part b: Now write the specification in *ladybug*.
Part c: Now express your design in *Alloy*.
Part d: Based on your experience for part b and c, compare ladybug and Alloy.

Question 4 (25 points): Consider the following data flow problem:
Upward Exposed Uses: This analysis determines what uses of variables at particular points are reached by particular definitions. For each use of a variable in a basic block find all the definitions that *reach this use*, i.e., there is path from the definition to the use so that the definition is not killed. Setup this data flow problem as a model checking problem. Use the steps we outlined in class: setup the model, write the CTL expression, provide the fixpoint equations. Provide informal reasoning why your answer is correct. You do not have to give a formal proof.

Question 5 (50 points): Note: For this part form a team of two.
One person will be the *implementor* and other the *tester*. The implementor will implement a *quick sort* routine. Read the quick sort algorithm from your favorite algorithms book. Use the description of the standard C library function **qsort**. Tester should write standard tests to test the basic functionality of quick sort (these are the standard test cases you will write during unit testing). Tester should also devise test cases that meet the *statement* and *decision* coverage criteria. Notice that for deriving test cases that meet the statement and decision coverage criteria you will have to construct the program graph for the quick sort code. Comment on your results.
Note: Please turn in the code, test cases, and the program graph.