CS 477: Formal Methods, Spring 2008
Problem Set 2 (due Tuesday, April 1st, 4pm)

Turn in your homework at Elaine Wilson's office (3229 Siebel).

1. Designing explicit information flow analyzer (15+(2+3+5)+15)

   Explicit information flow:
   Explicit information flow is meant to capture whether there is information leakage in a program, where sensitive information held in "high" variables leaks to "low" variables. For example, if `passwd` is a high variable, and `str` is a low variable, then an assignment of the kind `str := passwd[1]` leaks information. Information can leak also through a series of assignments: for example {`x := passwd[1]; str:= x`} also leaks information. The goal of this exercise is to build types such that the MOP definition will capture whether there is a leakage of information.

   Note: Explicit information flow captures straightforward flow of information through assignments only. It does not capture information flow through conditional constructs: for example, the program may have a statement

   ```
   if (passwd[1]='a') then x='a' else x='b'
   ```

   Though there is information leakage here, for the purposes of this exercise, we will not be concerned with it. Generic information flow is best defined as "for any pair of valuations of the high variables, the low variables computed are the same".... explicit information flow is an approximation.

   (a) Take a while programming language with integers only, as we have considered in class. Develop a MOP-based framework for deciding whether there is information flow from a set of high variables H to a set of low variables L in a given program. Your scheme should work for any given program P, H, and L. Set up a finite number of facts (depending on the variables in P) and give the transformation rules for these facts for each statement. Argue why your analysis is correct, informally.

   (b) **(i)** Is your flow analysis distributive?

   (ii) Is the MFP solution of your framework equivalent to the MOP solution? If yes, argue why; if no, give a counterexample program.

   (iii) Informally, describe how you would change the MOP framework to handle programs with procedure calls (call-by-value calls).

(c) Take a small program P of at least 10 lines without procedure calls. Using your framework, model the MOP solution as another program P' that keeps track of the set of dataflow facts for P. P' should be a direct translation of P, line-to-line, but works over a set variable D that keeps track of facts only.

Since P' is a program whose variables range over a finite-domain, you can model-check it. Using NuSMV, write a model for P', and construct LTL queries that ask whether there is a leakage in the program P'. Your LTL formula will be a simple reachability formula that asks whether a particular set of dataflow facts can be reached on some run. Report the program P, the model P', the NuSMV model, and the model-checking results. (Make sure your program P is "interesting" with respect to this problem; there should be at least some indirect flow.)