CS 140 Logic Programming
Fall 2006
Machine Assignment #2
Due September 25

Implement the ***solve*** algorithm in Figure 1 of the paper
Describing Prolog by its interpretation and Compilation, *Comm ACM Dec 1985*

In Assignment #2 you will only implement the propositional version of Prolog that *does not* include unification. It simply interprets a Prolog program without parameters or functions, given as data (constants) in the body of your interpreter.

This assignment *per se* has no relation with Machine Assignment #1. Nonetheless keep in mind that in the forthcoming Machine Assignment #3 you will have to combine Assignments #1 and #2 to produce a mini Prolog interpreter.

Start debugging with very simple examples and present as one of you input programs the example in my paper, namely:

```
1.  a :- b, c, d.
2.  a :- e, f.
3.  b :- f.
4.  e.
5.  f.
6.  a :- f.
```

Try to emulate as much as possible the behavior of a real Prolog interpreter when given the above input program or a similar one. Your implementation should handle multiple queries including of course multiple solutions to queries when they exist, as well as the final failure indicator. Report what happens when dealing with programs like:
p :- a. ***a.*** ***p :- a, p.*** or ***p :- p, a .*** ***a.*** with query ***?-p***
Finally, explain the **logical** and **procedural** semantics of the above programs