

# Homework #6

**Q1.** Program the semantics of the language whose definition is given below. Prolog code for a subset of a language can be found in this paper: [Horn Logic Denotations and Their Applications, appears in \*Future Directions in Logic Programming\*, Springer Verlag, 1998.](#)

P ::= C.  
C ::= C1;C2 | if B then C1 else C2 endif | if B then C endif | while B do C endwhile | I := E  
E ::= E1+E2 | E1\*E2 | E1-E2 | (E) | I := E | I | N  
B ::= E1 = E2 | E1 > E2 | E1 < E2 | not(B)  
N ::= 0 | 1 | 2 ... | 8 | 9  
I ::= x | y | z | w | u | v

Assume that all programs written in this language take 2 inputs that are found in variables x and y before the program begins execution, and produce one output found in variable z after the program execution is over.

You can modify the DCG you developed in HW4 to get a DCG for the above grammar. Notice that both arithmetic and boolean expressions have side-effects.

Note from TA, for Q1.

- (1) Please make sure that the program is your own work (and do not share your code). I plan to check for any potential plagiarism. (You may use all the code from the paper of Dr. Gupta: “Horn Logic Denotations and Their Applications”)
- (2) Print all your programs and query/test-run and submit it on the due date as usual.
- (3) Send me ([rkm010300@utdallas.edu](mailto:rkm010300@utdallas.edu)) your prolog code only for Q1 attached (as \*.txt or \*.pro or \*.pl file) so that I can run your program with the following sample query by myself. (Send only your complete prolog program code. If I cannot load or run your program, then your program is not working! Put some comments if I need any special instruction to load and run).

If you prefer, you may hand in CD containing all your program codes and test runs results.

- (4) Test your prolog program with the following 8 cases of simple programs and attach your result for each case. You may use the initial values for x and y to be x=2, y=3, thus to output z.

- (1) z:=x.
- (2) z:=x+y.
- (3) z:=x; z:=z+y;
- (4) if x=y then z:=1 else z:=0 endif.
- (5) if x>0 then z:=x else z:=y endif.
- (6) if not(x=y) then z:=x else z:=y endif.
- (7) z:=0; while x>0 do z := z+1; x:=x-1 endwhile.
- (8) z:=1; w:=x; while w>0 do z :=z\*y; w:=w-1 endwhile.

For example, to run case (1) z:=x, you may use the following query where p should be your DCG starting symbol.

```
?- program(P,[z, :=, x, .],[], write(P), program_eval(P,2,3,Z)).
```

Or you may use something like main/3 as defined (in Figure 4 of the paper, “Horn’s Logic Denotations ...”) to run a query such as this one: ?- main(2,3,Z).

where main/3 is:

```
main(ValX, ValY, A) :- program(P, [z, :=, 1, ;, w, :=, x, ;, while, w, >, 0, do, z, :=, z, *, y, ;, w, :=, w, -, 1, endwhile, .],[], write(P), prog_eval(P,ValX,ValY,A)).
```

or you may name main with test case# as following (for example, for case #8)

```
main8(ValX, ValY, A) :- program(P, [z, :=, 1, ;, w, :=, x, ;, while, w, >, 0, do, z, :=, z, *, y, ;, w, :=, w, -, 1, endwhile, .],[], write(P), prog_eval(P,ValX,ValY,A)).
```

**Q2.** Use Mixtus to generate code for a sample program. Remember you'll have to comment out the code for update and access routines before partial evaluation.

To run Mixtus type 'mixtus' on apache, jupiter, cs1, cs2, .. (any of the Sun machines).

To load a program file code.pl in mixtus type 'pconsult('code.pl').' at the mixtus prompt.

To partial evaluate a goal under mixtus type the goal main(...), type 'pe(main(...))' at the mixtus prompt.