CSE505 – Spring 2018

Assignment 4– Prolog (continued)

Due Date: Weds, April 19 and 26 (11:59 pm)

You may work in pairs for this assignment.

Problem 3 (due April 26): Develop a definite-clause grammar (DCG) for TINYPL by completing the definition given in the file `grammar.pl`. You need to provide the DCG for a function definition and also for function calls – their syntax charts are shown on the next page. Lecture 19, slide 15 gives you a good start on constructing the required DCG.

For each function definition that is parsed, a Prolog fact of the form `fun(Name,Stmts)` should be asserted using the `assertz` builtin predicate. For this assignment, a function's parameters and declarations should be parsed but no semantic term need be constructed for them. For the test program in `defs.txt`, the `fun` facts to be asserted are illustrated in the file `samplefacts.txt`.

In the file `analyzer.pl`, comment out the `calls/2` facts that you used in Problem 2, but keep the remaining predicates. The `calls/2` facts will be generated by the `load` predicate in file `tinypl.pl` after parsing the TINYPL program.

The file `tiny.pl` is the top-level file and it includes `grammar.pl` and `analyzer.pl`. Compile `tiny.pl` in Prolog and proceed as follows:

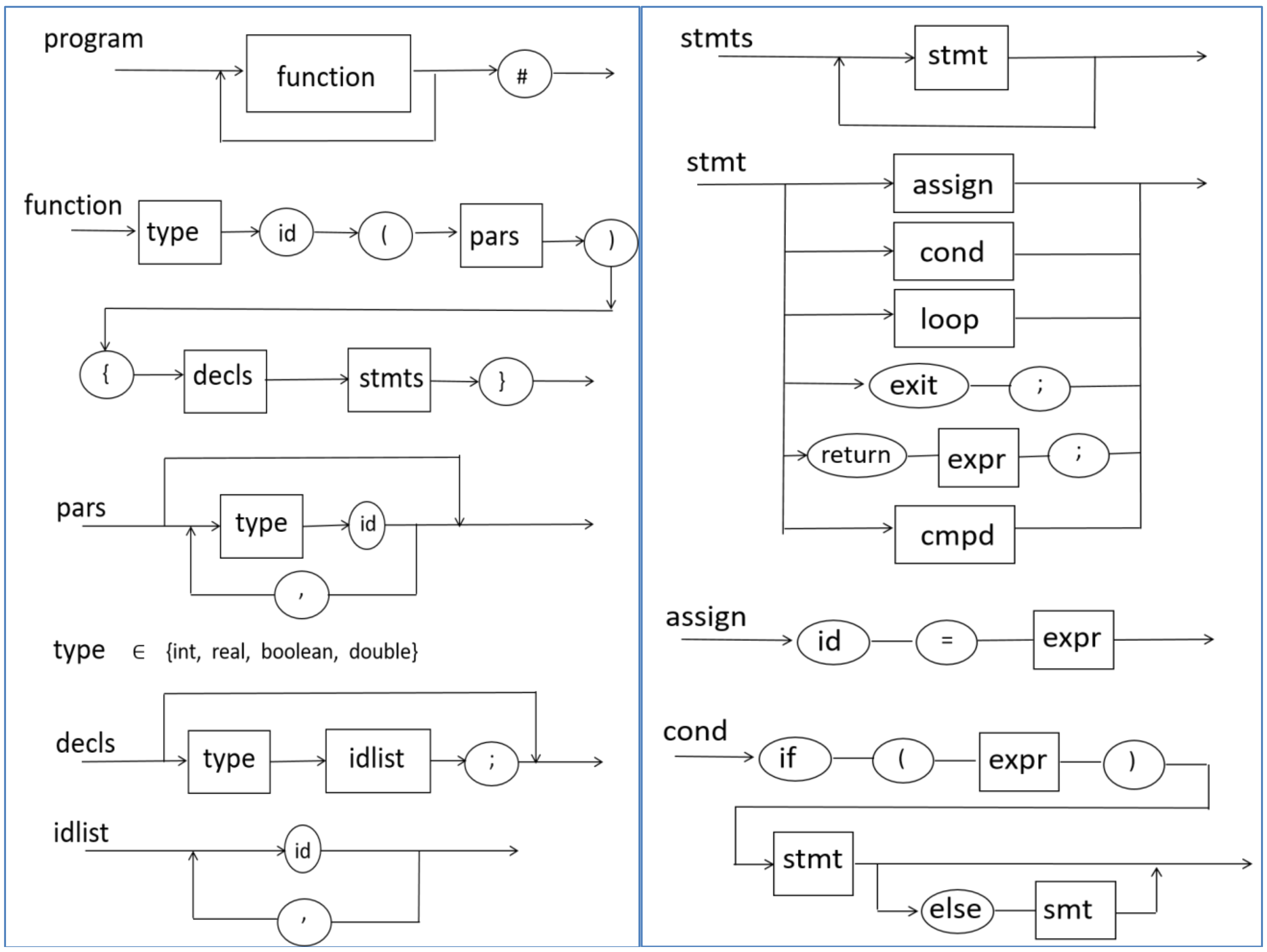
```
?- load('defs.txt').    % invoke the TinyPL parser on defs.txt
?- go.
     Tiny PL Call Graph Analyzer. Commands are:
        callers(f, L).
        undefined(L).
        is_recursive(f).
        all_calls(f, L).

?- callers(f, L).
   …
?- is_recursive(f).
   …
```

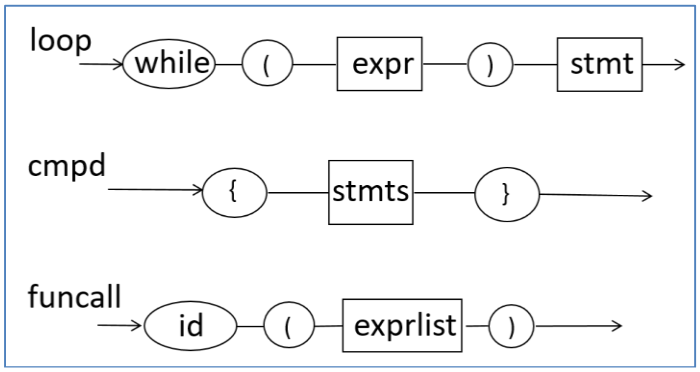
WHAT TO SUBMIT:

Problems 2-3 (submit by April 26): Make a directory called `A4_Prob23_UBITId` if working solo or make a directory called `A4_Prob23_UBITId1_UBITId2` if working as a pair (give UBITId's in alphabetic order). Put `defs.txt`, `analyzer.pl`, and `tinypl.pl` in this directory, compress the directory, and submit it using the `submit_cse505` command.

End of Assignment 4

Syntax Charts on next page →


program
function
#
function
type
id
(
pars
)
{
decls
stmts
}
pars
type
id
,
type ∈ {int, real, boolean, double}
decls
type
idlist
;
idlist
id
,
stmts
stmt
stmt
assign
cond
loop
exit
;
return
expr
;
cmpd
assign
id
=
expr
cond
if
(
expr
)
stmt
else
smt


loop
while
(
expr
)
stmt
cmpd
{
stmts
}
funcall
id
(
exprlist
)