```
digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
lowercase = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't'
          | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' ;
uppercase = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R'
          | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' ;
letter = lowercase | uppercase ;      spc = {' '}- ;
comma = ',' , [spc] ;      semicolon = ';' , [spc] ;      dot = '.' , [spc] ;      nat = {digit}- , [spc] ;
id = (letter | '_') , {letter | digit | '_'} , [spc] ;      id list = [nat] , id , {spc , [nat] , id}
int = ['-' | '+'] , nat ;      value = ['-' | '+'] , { digit }- , ['.' , nat] ;
equal = '=' , [spc] ;      assign = ':=' , [spc] ;
left bracket = '(' , [spc] ;      right bracket = ')' , [spc] ;
unary op = ( 'not' | 'abs' ) , [spc] ;
binary op = ('+' | '-' | '*' | '/' | '&' | '|' | '<' | '<=' | '=' | '>=' | '>' | '<>') , [spc] ;
expr = value | id | left bracket , expr , right bracket | unary op , expr | expr , binary op , expr ;
variable = ('variable:' | 'var:') , spc , id , equal , expr , {comma , id , equal , expr} ;
mutex = ('mutex:' | 'mtx:') , spc , id list ;
semsync = ('semaphore:' | 'sem:' | 'synchronization:' | 'sync:') , spc , [int , spc] , id list ;
cpvw = ('C' | 'P' | 'V' | 'W') , left bracket , id , right bracket ;
instruction = id , assign , expr | cpvw | sum | left bracket , [sum] , right bracket ;
sequence = instruction , {semicolon , instruction} ;
plus = '+' , [spc] ;      left square bracket = '[' , [spc] ;      right square bracket = ']' , [spc] ;
sum = sequence , {plus , [left square bracket , expr , right square bracket , plus] , sequence} ;
proc = ('process:' | 'proc:') , spc , id , equal , sequence , {comma , id , equal , sequence} ;
declaration = {variable | mutex | semsync | proc} ;
program = [spc] , declaration , ['init:', spc , id list] ;
```

Figure 1: The Paml grammar – following EBNF standard