Vocab/Grammar

- A *name* or a *variable* is a sequence of characters, not including a space or one of the following: `" , ' ` ( ) [ ] { } | ; #`:
- A *primitive* is something that is assigned meaning, like math functions
- A *variable* is a name without pre-assigned meaning
- A *value* is one of : number, symbol
- A *number* is a real number.
- A *symbol* is `'` followed by a sequence of characters

The rest of the core grammar is as follows

```
expr(cont…) = (reachable expr)
            | (reachable? expr variable)
            | (path-to expr variable)
```

```
 program = def-expr ...

def-expr = def
         | expr

     def = (define (variable variable variable ...) expr)

    expr = variable
         | value
         | (primitive expr expr ...)
         | (variable expr expr ...)
         | (cond [expr expr] ... [expr expr])
         | (cond [expr expr] ... [else expr])
```

Scoping

- (define (variable v2 ...) expr) binds v2 in expr and does not bind variable in expression and only binds variable for all top level forms below the define
- (define variable expr) variable is bound in all top-level forms below the define and variable is not bound in expr

Semantics

- (define (variable v2 ...) expr) defines a function of parameters v2 ... with the body expr
- (define variable expr) defines a constant with the value
- (reachable func) takes a func and print all the reachable names from the given function application
- (reachable? func name) returns 1 if you can reach the name from this function, else 0.
- (path-to expr variable) prints the constraints to the given name.