HW3 README

name: BoMoon Jung

student ID: 21600635

date: 2021.10.28

1. How to input the command line(in case VScode)

1) Language: python3

2) command for parser: % python3 Main.py -p “{concrete code}”

   ex) % python3 Main.py -p "{with {x 3} {with {f {fun {y} {+ x y}}} {with {x 5} {f 4}}}}"

3) command for interpreter: % python3 Main.py -p “{concrete code}”

   ex) % python3 Main.py "{with {x 3} {with {f {fun {y} {+ x y}}} {with {x 5} {f 4}} }}"

2. Module description

1) Main.py: Main function is in here.

2) Parser.py: Parser

3) Interpreter.py: Interpreter

4) LFAEDS.py: LFAEDS type definition

5) LFAEDSValue.py: LFAEDS-Value type definition

6) DefrdSub.py: DefrdSub type definition

3. BNF for LFAEDS

LFAEDS ::= <num>
| {+ <LFAEDS> <LFAEDS>}
| {- <LFAEDS> <LFAEDS>}
| <id>
| {fun <id> <LFAEDS>}
| {dsfun <id> <LFAEDS>}
| {<LFAEDS> <LFAEDS>}

4. Design

1) First, the concrete code is received by CLI. If the CLI receives the '-p' option, only the parser is executed and the result is printed. If there is no '-p' option, the overall design is to convert the abstract code through the parser and output the actual result through the interpreter.

2) Various 'define-types' are each declared as a class. And it inherits the declared class and declares each sub-type as a child class. In this way, an object is created and it becomes easier to process by sub-type when the interpreter is used.

3) By using recursion in the parser and interp functions, it is correctly implemented even if a value other than <num> is returned, such as (+ <LFAEDS> <LFAEDS>).

4) To implement Laziness, the exprV class, which is a child class of LFAEDSValue class, is used. In this class, there is a variable called box. If there is a calculated number, the number is stored, and if there is no calculated number, 'False' is stored. So, after checking the box in the strict function, you can take the appropriate action.

5) After declaring the DefrdSub class to implement defering Subsitution, declare mtSub with nothing and aSub that serves as a cache as child classes. And whenever the id value appears, the cache is checked through the lookup function. At this time, if the id cannot be found in the cache, a ‘free identifier’ error is output.

6) To implement Dynamic Scope, add the Dsfun class as a child class of LFAEDS. And pass the cache at the corresponding depth and the most recent cache as parameters to the lookup function. Therefore, while checking the cache of the corresponding depth, it receives the 'value' of the corresponding cache. At this time, the type of 'value' just received is the type of 'exprV'. And 'expr' of 'exprV' receives LFAEDS type. In this case, if 'expr' is 'dsfun', then the cache of 'exprV' is updated with the latest cache. Then, dynamic scope becomes possible. If it is a static scope, you can leave the cache of exprV in its original state.