Lisp interpreter report

MATTHEW KACHENSKY, New Mexico Institute of Mining and Technology, USA

ACM Reference Format:
Matthew Kachensky. 2022. Lisp interpreter report. 1, 1 (April 2022), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Progress report can be found below

1 IMPLEMENTATION

1.1 Variable reference

Variable references were implemented by looking through the list of current variables, and if the input matched the variable in question, it would print the value of the variable.

1.2 Constant literal

Constant literals were implemented by taking the users input, checking if the input was a number, T or NIL then just returning that value

1.3 Quotation

Quotation was implemented by taking the input with the quotation and removing the quotation and returning the value after the quotation

1.4 Conditional

This was done by separating the conditional part of the statement as well as the if true and if false part. Then we check the condition and based on the result of the condition we execute the if true or if false part

1.5 Variable definition

This was done by taking what the user wants to name the variable, then finding out what they want to set the variable to. The variable is then stored in a global dictionary

1.6 Function call

This was done by finding if the function exists in the current set of functions. If it does then take the arguments that the user entered. Make a copy of the function then replace all of the variables in the functions body and replace them with the arguments that the user provided. Then run the body of the program and print the output

Author's address: Matthew Kachensky, New Mexico Institute of Mining and Technology, Socorro, New Mexico, USA.

Made by Matthew Kachensky

Manuscript submitted to ACM

1.7 Assignment

This was done by taking what variable the user wants to change, then finding out if the variable even exists. If it does then, change the variable to the specified value

1.8 Function definition

This was done by taking the function the user wants to make, the list of arguments, and the body of the function. A list is create to specify the parameters and body which is then added in a dictionary of function names.

1.9 Arithmetic operators

This was done by taking the operator they want to perform then separating the next two statements and attempting to run them if they are expressions. Then returns the result of the operator

1.10 car and cdr

This was done by creating a copy of the variable if the input is a variable, then for car taking the first value of the list and returning that. For cdr, we return everything but the first element

1.11 cons

This was done by taking what the user wants to enter into the list. The list is then changed to add the value to the front of the list

1.12 sqrt and pow

This was done very similar to how the arithmetic operators were done in that the input was processed and after the operation takes place. The result of the operation is then returned

1.13 Conditional symbols

This was done by finding what symbol the user inputted then running the operation the user wants to perform. After which the result is determined to be True or False, then is converted into T or NIL

2 CLASS TOPICS RELATING TO IMPLEMENTATION

Since this interpreter was done in python, it makes great use of dynamic type checking. As seen in the program, a input can have multiple different types without specifying the type. Inside of this project, there is a lot of recursion which utilizes dynamic scoping as well. One of the biggest examples is in the function call. The conditional invokes a recursive call to change all variables of a certain name to the number given by the user. This makes use of dynamic scoping, as when referring to the variables in the method, it checks the value of the function which is taken from where the code was called.

3 OTHER IMPLEMENTATIONS

While a lot of the problems that I looked to solve in this project were achieved, it definitely couldn't have been done without the use of two global variables. One that looks at the currently defined variables and one that looks at the

Lisp construct	Status
Variable reference	done
Constant literal	done
Quotation	done
Conditional	done
Variable definition	done
Function call	done
Assignment	done
Function definition	done
Arithmetic operators	done
car and cdr	done
cons	done
sqrt and pow	done
Conditional symbols	done

Table 1. Progress report

currently defined functions. Because variables and functions are a huge part of lisp, it is important that they can be accessed anywhere at any time. Therefore through the use of global variables I was able to simplify and make my code easier to work with.

4 EXTRA MENTIONS

For every function implemented there are some weird gimmicks for example, if you are to enter more than what is wanted for a function. i. e. (cons '4 '(5 6 7) (4)) The program will ignore the (4) as it only cares about the 'cons', '4' and "(5 6 7)' While the program does function exactly what was required of this project, it is by no means perfect and has a lot of room for improvement

5 CONCLUDING THOUGHTS

This project was great, I had never took sometime to learn another programming language and this project allowed me to explore another programming language. Even though the project is not perfect, it was a great way to expand my knowledge and practice working with other languages which I had not yet touched

6 REFERENCES

As this is my first time using python, a lot of my references can be found: https://docs.python.org/3/ https://stackoverflow.com/