CS2353 Discrete Math

Project: Programming Language

Abstract:

Discrete Mathematic is a study of structure. Have you wonder how it would be use in the computer science world? Well, computer scientist used discrete math every day. Even the tools are made by discrete math, such programing language, cryptography, network, etc. The goal for this paper is to analyze and appreciate how programming language being made.

Introduction:

This paper will describe how programming language was created, and what data structure was used to create a programming language. The language going to be use will be like “Lisp”, what is “Lisp”? Lisp is one of the oldest programming languages in computer science beside Fortran. It is being used by many scientists for AI research because of its heavy use of mathematical theory. The mathematical grammar structure for our lisp going to be ***polish notation*** for the sake simplicity. And the programming language was used to write the language is going to be C due for its fast compile time, and awesomeness.

What is a programming language?

To create a programming language, we must understand what is programming language? A programming language is a formal language comprising a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms. In this case, the programing language we are going to use to implement algorithm to

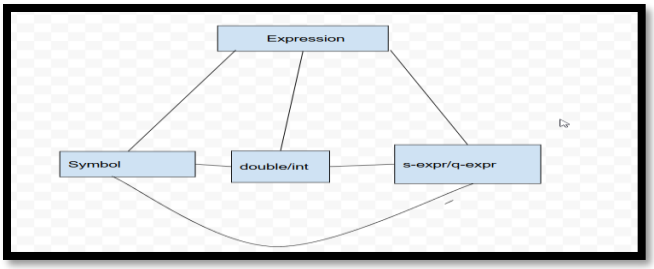
create another language is going to C. Thing to keep in mind while reading this, q-expr is like a list of an array, s-expr here mean the parathesis mathematic used to create order of operations.

What is a parser?

A parser in programming language is a process of analyzing string of symbols, in computer languages to form a grammar structure rule for a programming language. Parser is the most important step to create a programming language. It allows programmers to establish their own grammar rules. In this program, the parser library already written for us. Which help to create the programming rule tremendously easier. In this picture, there are two common data types in every language, double and int. The rule for it was [regex](), regex is a matching language. It a tool was used by many security researchers to find malware pattern efficiently. In this case, the software used it to establish a rule to look for the input from 0-9, - sign is an optional value.

```
mpca_lang(MPCA_LANG_DEFAULT,
    "                                                                    \
   double   :  /-?[0-9]+\\.[0-9]+/;                                      \
   int      : /-?[0-9]+/;                                                \
   sexpr    : '(' <expr>* ')';                                           \
   qexpr    : '{' <expr>* '}';                                           \
   symbol   : \"list\" | \"head\" | \"tail\" | \"join\" | \"eval\"       \
            | '+' | '-'  | '*' | '/' | '%'                               \
            | \"add\" | \"sub\" | \"mul\" | \"mod\";                      \
   expr     : <double> | <int> | <symbol> | <sexpr> | <qexpr>;           \
   magicK   : /^/ <expr>* /$/ ;                                          \
 ",
 Int, Double, Expr, Sexpr, Qexpr, Symbol, MagicK);
 //
```

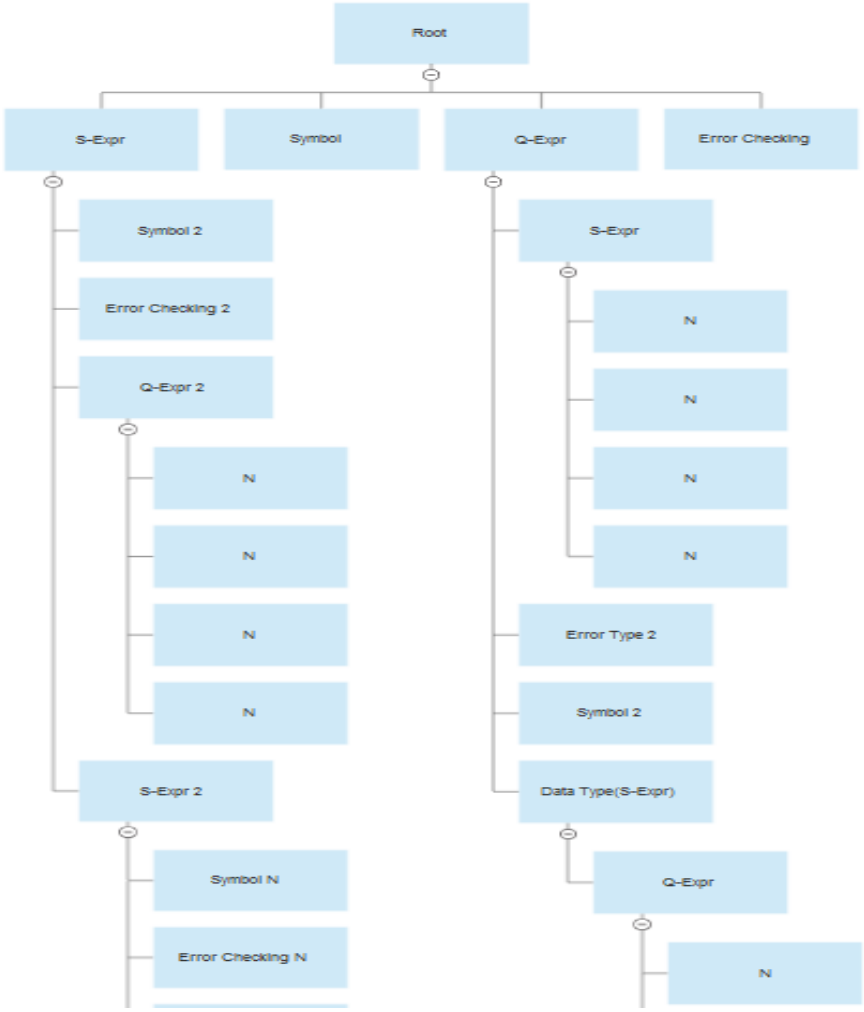
So how was all this related to discrete math? If we visualized this in our head. It would be a graph. Yes, graph theory was used here to create a grammatical structure. How exactly this work? So, an expression will start with a symbol, or double or int or s-expr or q-expr, and the pattern could be repeat, which means this is a walk graph.




```
MagicK> + 1 2 3
6
MagicK> + 1 2 3 (* 2 4)
14
```

Now, let looks at it in action. Remember, the format used here is polish notation. Analyze this code we see that it starts with a symbol (+), with int, the parentheses brace is our s-expr. Time to look at our code compare it to the graph. Isn't it the same structure in the graph? And it is a walk graph. Alright that sum up how graph theory was implemented. It is time for even more fun stuff.

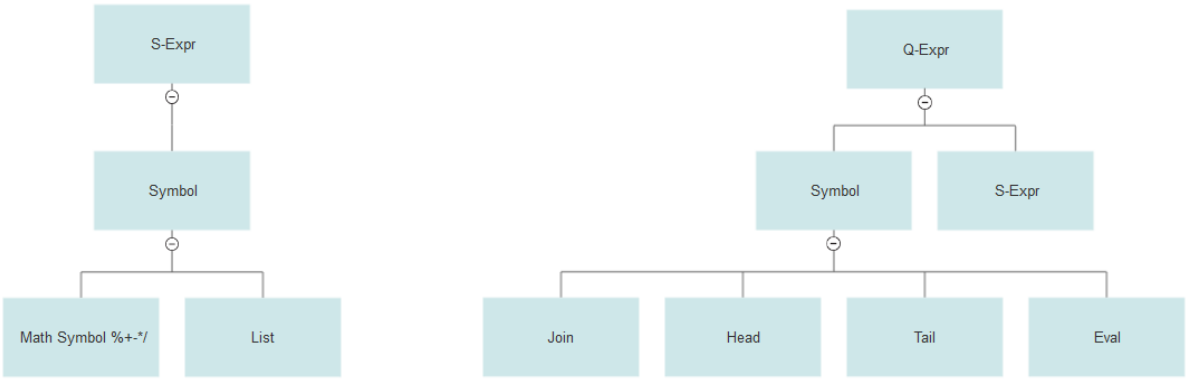
Abstract Syntax Tree and Error Handling:



Data structure, one of the most important topics in computer science. Yet, it also the hardest one to implement. In order able to create rules to do math and run in reasonable time. One must be able to find a right abstract data structure for the project. For this project, the discussion will be on Abstract Syntax Tree, which being commonly used to create programming language. Due to its complexity and the use of recursion. The AST was one of the hardest things to implement in this project. Based on the table, the tree can grow infinite large. In order to traverse around the tree, computer scientist create a technique called “Recursion”. “Recursion” is method solving complex problem that involve repeating pattern. The first step to implement recursion was to

observe the pattern of this tree. The tree started from the root, the root had four classes, but only two classes able to grow. From this information, the base cases here are symbol and error checking. If everything went through smoothly, return the result, if there is an error stop the recursive program immediately. Check to see if there is S-expr and Q-expr, if there is one went through its child object. Keep going until it computes all the value without receiving any error. Quiet, useful right? Not only it allows to traverse through the data. It was able to help handling the errors while traversing through the tree.

You, as a reader might have a question, “Why did the writer wrote S-Expr and Q-Expr when they look almost identical in the tree?” The answer for it is, they are not the same internally. Q-Expr had certain symbols S-Expr cannot use, and they behave differently. Only S-Expr allows to perform math computation. Treat Q-Expr as a vector, it has a list of value. To compute the value, programmer needs to take the data out from the list then perform the calculation. Same thing happening here.



Conclusion:

Computer Scientists used discrete mathematic every day. This is a part of their life, whether they are developing a software or invent a new algorithm. Discrete Math must be there to help

scientists produce an efficient solution. Even thou, this is only a small part in a developing programming language. It already a complex task to do and required tremendous of time to find and implement the right algorithm.