

Lexical Analyser Overview

Name: Janga Tushita Sharva

Roll Number: CS21BTECH11022

August 27, 2023

1 End to End Compilation steps of lex.yy.c

Please find the main.l file in the folder submitted and execute the following commands. (These commands match with the compilation steps provided in the lecture slides.)

```
$ lex lex_source_program.l
```

For compiling lex.yy.c file,

```
$ gcc -o ex.o lex.yy.c
```

Now, for using the lexical analyser on input file please make sure you load the files to **T1 Directory**. Now you have to give three arguments: **The input file name, The C output file name, The token output file name.** For example,

```
$ ./ex.o 1.txt C_1.txt seq seq_token_1.txt
```

The output files C_1.txt and seq_token_1.txt will be opened from the folders **TC** and **TK** folders respectively.

2 Regular Definitions

```
letter [a-zA-Z]
digit [0-9]
integerset {digit}+
character (\'(.)\')
```

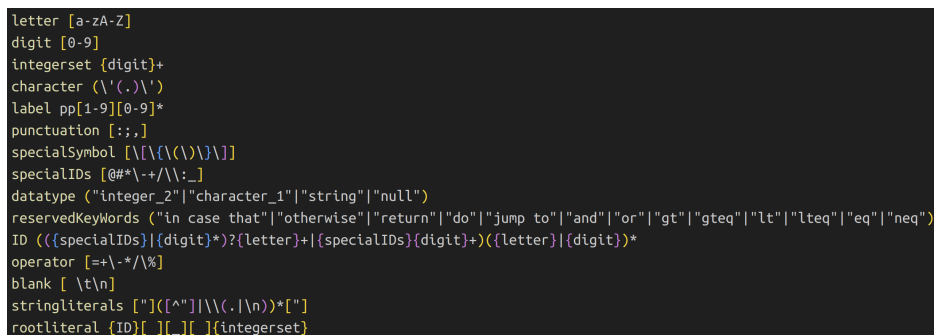


Figure 1: Regular Definitions

letter, digit, character and integerset: These are simple regular expressions used so as to provide a skeleton to the structure of other regular expressions.

label: This expression is used to capture all the strings which represent the program counter. This had been kept separate from the punctuation :.

character, stringliterals: These expressions are used to capture the characters and string literals respectively. Regarding the former one, it is just a single character enclosed within single quotes. The single dot there represents any character except the newline character. And coming to the later one, pattern matches a string that starts with a double quote mark, followed by either a sequence of characters that are not double quotation marks, or escaped characters or any characters/newlines following an escape character, and finally ends with another double quote mark.

specialSymbol: This expression matches the brackets in the source language. This is different from the regular expression `specialIDs` which represents

specialIDs: The valid characters at the beginning of an identifier. They could've been concatenated easily to the actual regular expression for ID, but still :)

ID: I tried to capture all the valid identifiers here. My thought process was as follows: Only special characters or Only integers are invalid. So when they are independently appearing, we are making sure that they are accompanied by atleast one letter. And the second definitions, when they do appear together, they are valid. In both cases, they can then be followed by any combination of letters and digits.

blank: All the strings with spaces are captured by this definition. I chose to keep this because I wanted to carry forward the exact spacing from source language code to the translated C code.

rootliteral: This is used especially to capture the strings of the form `identifier _ n`, which is by definition the n^{th} root of the identifier. Here comes a limitation which is discussed in the next section.

punctuation: As the name suggests, it is a set of punctuation, but I had not mentioned the double quote and single quotes here. I had used those quotes as beginning of strings, so that the constant value inside the strings is recognised as a constant instead of a potential identifier or instead of generating an error message.

Rest all are self explanatory.

3 Known Limitations

3.1 The root literal

My code cannot capture the expressions of the form `integer _ integer`. In ideal case, it must be able to recognise the first and third strings as integers and the underscore as operator. This is the implementation I used:

```
char *str;
str = malloc(yyvaleng);
strcpy(str, yytext);
char id[50]; char num[50]; char opt[2];
sscanf(str, "%s %s %s", id, opt, num);

fprintf(fp, "pow(%s, 1./%s)", id, num);
fprintf(yyout, "ID: %s\n", id);
fprintf(yyout, "operator: _\n");
fprintf(yyout, "constant: %s\n", num);
break;
```

In this, if I must be able to decide if it is an integer or an identifier, I must be able to check for the regular definition of the first string individually (whether it is an identifier or a number), for which I was not able to figure out a solution. Thus I left it as an exception to my code.

3.2 Again, the root literal

The length of the names of identifiers and the number of digits cannot exceed 50.