Joon Kyung Kim

Sung Hye "Lauren" Jeon

Taekyu Kim

Project Phase 1

**Grammar Changes**

The grammar LL(1) is made by not allowing the rules to be ambiguous, to be left recursive, to have any joint first sets, and to have any joint first set & follow set if the former can be nullified.

Rules modified :

funct\_declaration\_list

funct\_declaration

main\_function

type

id\_list

stat\_seq

stat

expr

value\_tail

index\_expr

To remove ambiguity, the group was broken with lower precedence closer to the root.

This was done for the rule expr and index\_expr. To remove left recursion, our team did left factoring and created tail with new rules and made the tail to loop. In this step, the rule expr and index\_expr were modified again. To not have any joint first sets, we bound the first token that caused the error and wrote a tail and a modifed productions of the rule. These rules were modified in this process: funct\_declaration\_list, funct\_declaration, main\_function (‘void’ caused the problem), type (‘array[INTLIT]’ caused the error ), stat(‘ID’ caused the error ), id\_list (‘ID’ caused the error ), stat\_seq (stat caused the error ), and value\_tail ( [index\_expr caused the error ] ).

ANTLR was used to describe the Tiger’s lexical specifications, and the scanner and the parser were generated from the specifications. When the specifications had errors, ANTLR threw the errors and reported details: the location of the error and the tokens that caused the error.

Also, out team added the line ‘k=1; backtrack = no;’ inside the option in the .g file to check if the grammar is LL(1) and used ANTLR’s error handling to fix the problems.

**Expects of Error Generated**

The given Tiger file contains an error that is missing a semicolon in the for loop (sum := sum + X[i] \* Y[i]). Therefore, when we run our ANTLR code and give input as the given Tiger file, the result should say something about there is a missing semicolon, or it could give more error messages due to the missing semicolon. When we run the given Tiger code with our ANTLR code, it gives errors as shown in Figure 1.

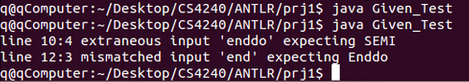


Figure 1. Error generated by given Tiger file.

**Sample Tiger Program**

For our test Tiger file, we have tried to cover as many grammar cases as possible. We started the program with declaring a global scope type declaration in double array. We have wrote a function named 'Smaller' that takes two integer values and return a integer value. In the 'Smaller' function, it has a variable declaration and a ‘if else’ block. Since there are two ‘if’ blocks related each other, at the end of the ‘if’ blocks, it should have two 'endif'. After the ‘if’ blocks, it returns the result of ‘if’ blocks.

The second function 'Void\_Check' is for checking another type of return value (void) and NULL for parameter lists. Inside of the function, it has a variable declaration, and a if block. Since the return type is void, it returns nothing.

For The third function 'Concat' gets one integer and one fixedpt type variables as parameter and has return type fixedpt. The function starts with a variable declaration and a ‘if’ block after the declaration. Inside of the ‘if’ block, it has a while loop that calculates fixedpt variables. After the while loop and if block, it will return the result 'toReturn' which is fixedpt.

The last function is the main function, and it should be the last function of the program. The main function is consisted with two blocks. The first block starts with declaring integer base type array, and the next declaration uses the type ArrayInt that we have declared. In the first block, it has a for loop that does integer calculation and uses a function 'printi()'. In the second block, it uses type 'DoubleArray' that we have declared at the start of the program as global scope. In the for loop, it has a integer calculation, and it uses the function 'printi()' after the for loop.