TETRISJET

TETRIS GAME PROJECT PROPOSAL

Tanmay Devale : 2019A7PS0066G

Ramanathan Rajaraman : 2019A7PS0115G

Vishal Sharma : 2019A7PS0036G

Param Biyani : 2019A7PS0059G

Sarthak Chaudhary : 2019A7PS0125G

Vishal Vivek Bharambe : 2019A7PS0160G

This Report is made as a partial fulfillment of the requirements for the Midsem assignment submission for COMPILER CONSTRUCTION CS F363, under Prof. Ramprasad S. Joshi .

TABLE OF CONTENTS

SYNTAX DIRECTED TRANSLATION SCHEME

PARSER GRAMMAR

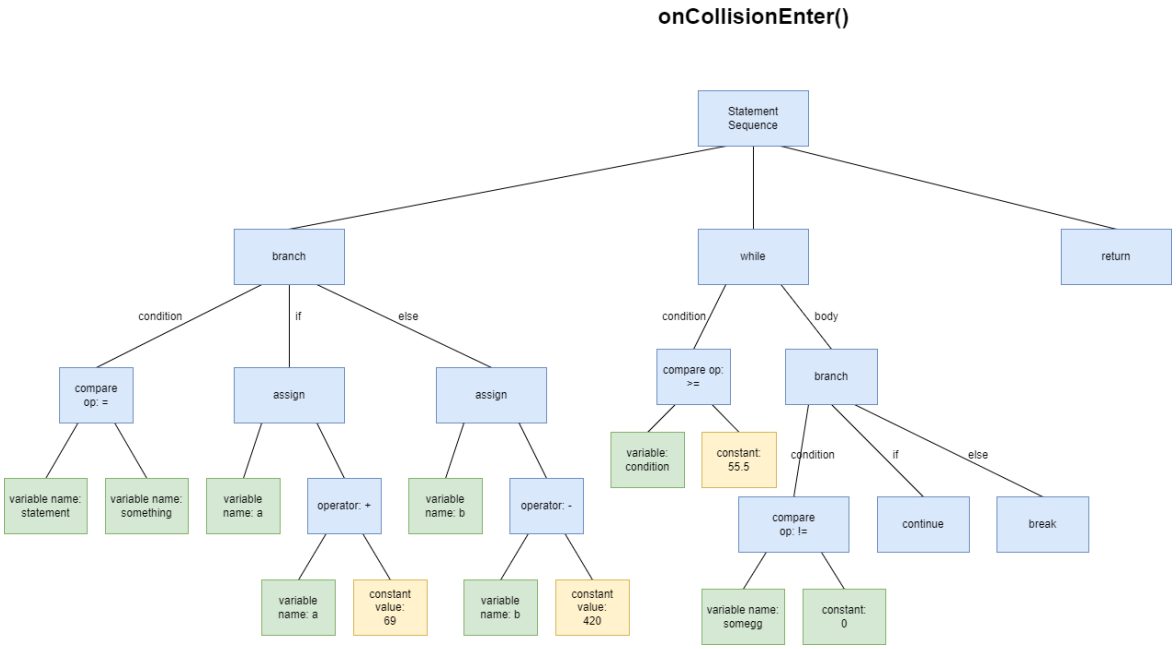
The following code block contains the CFG grammar as we have currently implemented it in bison. We follow the hierarchy from the complete file -> event functions -> statement list. Statements follow a syntax identical to C++. This allows us to skip the generation of an AST for translation. We can use the Parser to do a syntax check ensuring that it is in line with the environment in our game engine written in C++. This also ensures that error detection and highlight the lines number in question is also streamlined.

```
FILE      : FUNC FILE
          | comment FILE
          |
          ;
FUNC      : KEYWORDFUNC '(' ')' '{' STM_LIST '}' {printf("%s parsed\n",
$1);}
          ;

STM_LIST  : STM STM_LIST {printf("%s parsed\n", $1);}
          |
          ;

STM       : IF '(' EXPR ')' STM ELSE STM {printf("%s-%s parsed\n", $1, $6);}
          | IF '(' EXPR ')' STM {printf("%s parsed\n", $1);}
          | WHILE '(' EXPR ')' STM {printf("%s parsed\n", $1);}
          | DEC {printf("%s parsed\n", $1);}
          | ASN {printf("%s parsed\n", $1);}
          | CONTINUE ';' {printf("%s parsed\n", $1);}
          | BREAK ';' {printf("%s parsed\n", $1);}
          | RETURN ';' {printf("%s parsed\n", $1);}
          | MOVEMENT '(' id ')' ';' {printf("%s parsed\n", $1);}
          | id '.' id '(' string ')' ';' {printf("%s parsed\n", $1);}
          | comment
          | '{' STM_LIST '}'
          | ';'
          ;

DEC       : INTTYPE id '=' EXPR ';' {printf("%s parsed\n", $1);}
          | FLOATTYPE id '=' EXPR ';' {printf("%s parsed\n", $1);}
          | BOOLTYPE id '=' EXPR ';' {printf("%s parsed\n", $1);}
          | CHARTYPE id '=' char_const ';' {printf("%s parsed\n", $1);}
```

```
          | INTTYPE id';' {printf("%s parsed\n", $1);}
          | FLOATTYPE id';' {printf("%s parsed\n", $1);}
          | BOOLTYPE id';' {printf("%s parsed\n", $1);}
          | CHARTYPE id';' {printf("%s parsed\n", $1);}
          ;

ASN       : id '=' EXPR ';' {printf("%s parsed\n", $1);}
          | id '=' char_const ';' {printf("%s parsed\n", $1);}
          ;

EXPR      : EXPR or_const EXPR
          | EXPR and_const EXPR
          | EXPR eq_const EXPR
          | EXPR rel_const EXPR
          | EXPR op_single EXPR
          | TERM
          ;

TERM      : float_const
          | int_const
          | char_const
          | id
          | id '.' id '(' ')' ';'
          | '(' EXPR ')'
          ;
```

TOP DOWN AST EXAMPLE

This AST starts at the FUNC level of the grammar. The Levels above this are trivial. The tree corresponds to the test case file called `testcase.txt`.



TOKEN LIST

```
%token IF ELSE WHILE CONTINUE BREAK RETURN HEADER
%token KEYWORDFUNC MOVEMENT BLOCKTYPE GRIDTYPE INTTYPE FLOATTYPE BOOLTYPE
CHARTYPE
%token float_const int_const char_const id op_single rel_const or_const
and_const eq_const
%token string comment space ERROR
```

CHALLENGES

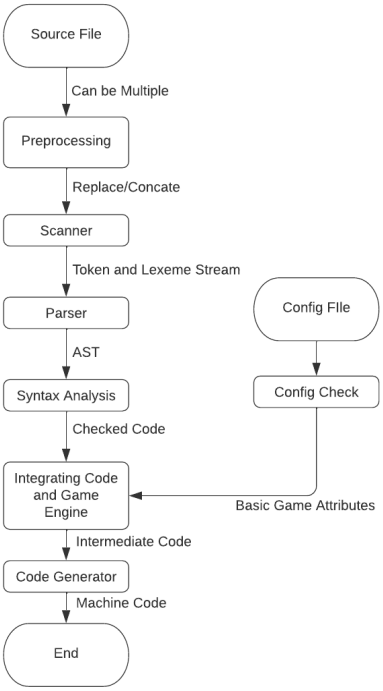
As the syntax and semantics we are following are identical to C++ our parser only needs to confirm if the programs conforms to our specifications. The only major challenge we see as possible is the non trivial information leading to errors after our syntax check in the form of scoping and typing.

Our Parser is able to detect the error location but giving suggestions on possible fixes is still a challenge.

TEST CASES

We have included a set of 4 test files to confirm syntax checking capabilities of our parser. This is a fairly exhaustive list for our specifications.

A COMPLETE END-TO-END TETRIS GAME ENGINE PROGRAMMING TOOLCHAIN



Our Pipeline remains the same except the AST is not passed to the Intermediate Code. The file after syntax check can be directly integrated with the game engine.

ACKNOWLEDGEMENTS

The proposal made above is to the best of our knowledge, consistent with the structure we plan to pursue for this project.

We would like to thank our TA Amit Chauhan and Rohit Garg for their guidance and Prof. Ramprasad S. Joshi and Kunal Kishore Korgaonkar.