PROGRAMMING LANGUAGES AND COMPILER

PROJECT-PHASE 2

OMAR MOHAMED EMAM 44

OMAR YOUSSEF 45

ISLAM YOUSRY 14

MOHAMED SAAD ALI 53

ANDREW ADEL 17

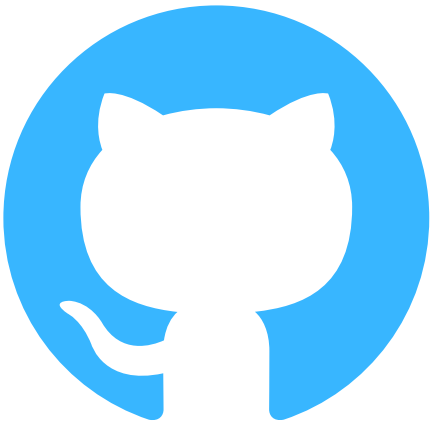
Requirements :

IT IS REQUIRED TO DEVELOP A SUITABLE SYNTAX DIRECTED TRANSLATION SCHEME TO CONVERT JAVA CODE TO JAVA BYTECODE, PERFORMING NECESSARY LEXICAL, SYNTAX AND STATIC SEMANTIC ANALYSIS (SUCH AS TYPE CHECKING AND EXPRESSIONS EVALUATION).

1- YOUR TASK IN THIS PHASE OF THE ASSIGNMENT IS TO DESIGN AND IMPLEMENT AN LL (1) PARSER GENERATOR TOOL.
2- THE PARSER GENERATOR EXPECTS AN LL (1) GRAMMAR AS INPUT. IT SHOULD COMPUTE FIRST AND FOLLOW SETS AND USES THEM TO CONSTRUCT A PREDICTIVE PARSING TABLE FOR THE GRAMMAR.
3- THE TABLE IS TO BE USED TO DRIVE A PREDICTIVE TOP-DOWN PARSER. IF THE INPUT GRAMMAR ISNOT LL (1), AN APPROPRIATE ERROR MESSAGE SHOULD BE PRODUCED.
4- THE GENERATED PARSER IS REQUIRED TO PRODUCE SOME REPRESENTATION OF THE LEFTMOST DERIVATION FORA CORRECT INPUT.
5- IF AN ERROR IS ENCOUNTERED, A PANIC-MODE ERROR RECOVERY ROUTINE IS TO BE CALLED TO PRINT AN ERROR MESSAGE AND TO RESUME PARSING.
6- THE PARSER GENERATOR IS REQUIRED TO BE TESTED USING THE GIVEN CONTEXT FREE GRAMMAR OF A SMALL SUBSET OF JAVA. OF COURSE, YOU HAVE TO MODIFY THE GRAMMAR TO ALLOW PREDICTIVE PARSING.
7- COMBINE THE LEXICAL ANALYZER GENERATED IN PHASE1 AND PARSER SUCH THAT THE LEXICAL ANALYZER IS TO BE CALLED BY THE PARSER TO FIND THE NEXT TOKEN. USE THE SIMPLE PROGRAM GIVEN IN PHASE 1 TO TEST THE COMBINED LEXICAL ANALYZER AND PARSER..

Source Code

View GitHub repository by clicking the below icon



USED DATA STRUCTURES:

- NODE WHICH CONTIANS
  - BOOLEAN TO INDCATE IF THIS NODE IS ACCEPTED OR NOT
  - STRING TO STORE THE VALUE OF THE TOKEN
  - VECTOR OF PAIRS (CONTIANING POINTER & STRING) TO POINT TO THE NEXT NODE
- GRAPH OF NODES WHICH CONTIANS
  - POINTER TO THE HEAD NODE
  - POINTER TO THE END NODE
- ENTITY WHICH IS
  - A SET OF NODES
- IN ADDITION TO USUAL DS SUCH AS
  - VECTORS
  - MAPS
  - SETS (STRINGS & CHARS)

Explanation of all algorithms

- Function **"replace_all "** :replace all matched string with replace string
- Function "**is_accepted** " :return true if the entity is accepted
- Function **"dis"** :calculate the result of disjunction of two graphs
- Function "**closure**" : calculate the result of closure of a graph
- Function "**add**" : add two graphs
- Function "**next_closure**" : get next states which is reachable from node by EPSON move
- Function "**next_move**" :get the next state of the node for all possible inputs
- Function "**calculate**" :calculate an operation to a graph
- Function "**constructNDFA**" :construct Nondeterministic automata from regulars expressions
- Function "**divide_on**" :divide the given string s into parts on the divider char
- Function "**addspace**" :add some space to some certain keywords
- Function "**add_keywords**" :add a new keyword
- Function "**add_puncuations**" :add a punctuation.
- Function "**expand**" : this function expands ranged values.
- * example 0-4 --> 0,1,2,3,4
- Function "**constructDFA**" : construct Deterministic Finite Automate from Nondeterministic finite Automate
- Function "**same_group**" :return true if the two entities belong to the same group
- Function "**minimize**" :preform minimize operation to the DFA
- Function "**constructMinimizedDFA**" :construct table state for Minimizer DFA
-

The resultant transition table for the minimal DFA

```
   ! * + - . / 0 1 2 3 4 5 6 7 8 9 < = > A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m
0 not | (!)7 (*)4 (+)1 (-)1   (/)4 (0)5 (1)5 (2)5 (3)5 (4)5 (5)5 (6)5 (7)5 (8)5 (9)5 (<)6 (=)2 (>)6 (A)3 (B)3 (C)3 (D)3 (E)3 (F)3 (G)3 (H)3 (I)3 (J)3 (K)3 (L)3 (M)3 (N)3 (O)3
1 acc |
2 acc |                              (=)9
3 acc |           (0)3 (1)3 (2)3 (3)3 (4)3 (5)3 (6)3 (7)3 (8)3 (9)3      (A)3 (B)3 (C)3 (D)3 (E)3 (F)3 (G)3 (H)3 (I)3 (J)3 (K)3 (L)3 (M)3 (N)3 (O)3 (P)3 (Q)3 (R)3 (S)3 (T)
4 acc |
5 acc |        (.)10   (0)5 (1)5 (2)5 (3)5 (4)5 (5)5 (6)5 (7)5 (8)5 (9)5
6 acc |                              (=)9
7 not |                              (=)9
8 acc |
9 acc |
10 not |           (0)12 (1)12 (2)12 (3)12 (4)12 (5)12 (6)12 (7)12 (8)12 (9)12
11 acc |           (0)11 (1)11 (2)11 (3)11 (4)11 (5)11 (6)11 (7)11 (8)11 (9)11
12 not |           (0)12 (1)12 (2)12 (3)12 (4)12 (5)12 (6)12 (7)12 (8)12 (9)12             (E)13           (L)8
13 not |           (0)11 (1)11 (2)11 (3)11 (4)11 (5)11 (6)11 (7)11 (8)11 (9)11
```

The resultant stream of tokens for the example test program.

```
METHOD_BODY
STATEMENT_LIST
STATEMENT STATEMENT_LIST`
DECLARATION STATEMENT_LIST`
PRIMITIVE_TYPE id ; STATEMENT_LIST`
int id ; STATEMENT_LIST`
id ; STATEMENT_LIST`
; STATEMENT_LIST`
STATEMENT_LIST`
STATEMENT STATEMENT_LIST`
ASSIGNMENT STATEMENT_LIST`
id = EXPRESSION ; STATEMENT_LIST`
= EXPRESSION ; STATEMENT_LIST`
EXPRESSION ; STATEMENT_LIST`
SIMPLE_EXPRESSION relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
TERM SIMPLE_EXPRESSION` relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
FACTOR TERM` SIMPLE_EXPRESSION` relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
num TERM` SIMPLE_EXPRESSION` relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
TERM` SIMPLE_EXPRESSION` relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
SIMPLE_EXPRESSION` relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
error: panic mode recovery is active
; relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
error: panic mode recovery is active
if relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
error: panic mode recovery is active
( relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
error: panic mode recovery is active
id relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
relop SIMPLE_EXPRESSION ; STATEMENT_LIST`
SIMPLE_EXPRESSION ; STATEMENT_LIST`
TERM SIMPLE_EXPRESSION` ; STATEMENT_LIST`
FACTOR TERM` SIMPLE_EXPRESSION` ; STATEMENT_LIST`
num TERM` SIMPLE_EXPRESSION` ; STATEMENT_LIST`
TERM` SIMPLE_EXPRESSION` ; STATEMENT_LIST`
SIMPLE_EXPRESSION` ; STATEMENT_LIST`
; STATEMENT_LIST`
error: panic mode recovery is active
) ; STATEMENT_LIST`
; STATEMENT_LIST`
error: panic mode recovery is active
{ ; STATEMENT_LIST`
; STATEMENT_LIST`
error: panic mode recovery is active
id ; STATEMENT_LIST`
; STATEMENT_LIST`
error: panic mode recovery is active
assign ; STATEMENT_LIST`
; STATEMENT_LIST`
error: panic mode recovery is active
num ; STATEMENT_LIST`
; STATEMENT_LIST`
STATEMENT_LIST`
error: panic mode recovery is active

error: panic mode recovery is active
}
```