PROGRAMMING LANGUAGES AND COMPILER

PROJECT-PHASE 1

OMAR MOHAMED EMAM 44

OMAR YOUSSEF 45

ISLAM YOUSRY 14

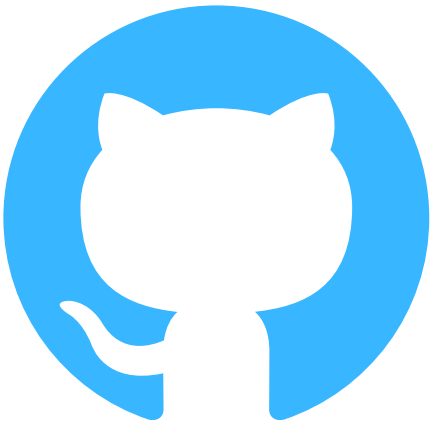
MOHAMED SAAD ALI 53

Requirements :

YOUR TASK IN THIS PHASE OF THE ASSIGNMENT IS TO DESIGN AND IMPLEMENT A LEXICAL ANALYZER
GENERATOR TOOL.
1- THE LEXICAL ANALYZER GENERATOR IS REQUIRED TO AUTOMATICALLY CONSTRUCT A LEXICAL
ANALYZER FROM A REGULAR EXPRESSION DESCRIPTION OF A SET OF TOKENS. THE TOOL IS REQUIRED
TO CONSTRUCT A NONDETERMINISTIC FINITE AUTOMATA (NFA) FOR THE GIVEN REGULAR EXPRESSIONS, COMBINE THESE NFAS TOGETHER WITH A NEW STARTING STATE, CONVERT THE
RESULTING NFA TO A DFA, MINIMIZE IT AND EMIT THE TRANSITION TABLE FOR THE REDUCED DFA
TOGETHER WITH A LEXICAL ANALYZER PROGRAM THAT SIMULATES THE RESULTING DFA MACHINE.
2- THE GENERATED LEXICAL ANALYZER HAS TO READ ITS INPUT ONE CHARACTER AT A TIME, UNTIL IT
FINDS THE LONGEST PREFIX OF THE INPUT, WHICH MATCHES ONE OF THE GIVEN REGULAR EXPRESSIONS. IT SHOULD CREATE A SYMBOL TABLE AND INSERT EACH IDENTIFIER IN THE TABLE. IF
MORE THAN ONE REGULAR EXPRESSION MATCHES SOME LONGEST PREFIX OF THE INPUT, THE LEXICAL
ANALYZER SHOULD BREAK THE TIE IN FAVOR OF THE REGULAR EXPRESSION LISTED FIRST IN THE
REGULAR SPECIFICATIONS. IF A MATCH EXISTS, THE LEXICAL ANALYZER SHOULD PRODUCE THE TOKEN
CLASS AND THE ATTRIBUTE VALUE. IF NONE OF THE REGULAR EXPRESSIONS MATCHES ANY INPUT
PREFIX, AN ERROR RECOVERY ROUTINE IS TO BE CALLED TO PRINT AN ERROR MESSAGE AND TO
CONTINUE LOOKING FOR TOKENS.
3- THE LEXICAL ANALYZER GENERATOR IS REQUIRED TO BE TESTED USING THE GIVEN LEXICAL RULES OF
TOKENS OF A SMALL SUBSET OF JAVA. USE THE GIVEN SIMPLE PROGRAM TO TEST THE GENERATED
LEXICAL ANALYZER.
4- KEEP IN MIND THAT THE GENERATED LEXICAL ANALYZER WILL INTEGRATE WITH A GENERATED
PARSER WHICH YOU SHOULD IMPLEMENT IN PHASE 2 OF THE ASSIGN

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	<	=	>
0	1	2	2	2		2	7	7	7	7	7	7	7	7	7	7	4	4	4
1																		2	
2																			
3																			
4																		2	
5							9	9	9	9	9	9	9	9	9	9			
6																			
7					5		7	7	7	7	7	7	7	7	7	7			
8							8	8	8	8	8	8	8	8	8	8			
9							9	9	9	9	9	9	9	9	9	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
0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1																										
2																										
3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4																										
5																										
6	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
7																										
8					5																					
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
0	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
1																										
2																										
3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
4																										
5																										
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7																										
8																										
9																										

Download output file by clicking this icon