Lexer Assignment Documentation

Mia de Leon, Anurag Kashyap, Austin Kao

February 9, 2021

Description

This is a lexer for the Tiger programming language using ML-Lex.

Regular Expressions for Identifiers and Integer literals

Strings beginning with A-Z, a-z

When these strings are parsed, the lexer generates the identifier tokens that take on the value of the characters inside.

Strings only containing characters 0-9

When these strings are parsed, the lexer generates the integer literal tokens that take on the value of the numbers inside.

Reserved Words

Tiger Expression	Token Name	Regular Expression
while	WHILE	“while”
for	FOR	“for”
to	TO	“to”
break	BREAK	“break”
let	LET	“let”
in	IN	“in”
end	END	“end”
function	FUNCTION	“function”
var	VAR	“var”
type	TYPE	“type”
array	ARRAY	“array”
if	IF	“if”
then	THEN	“then”
else	ELSE	“else”
do	DO	“do”
of	OF	“of”
nil	NIL	“nil”

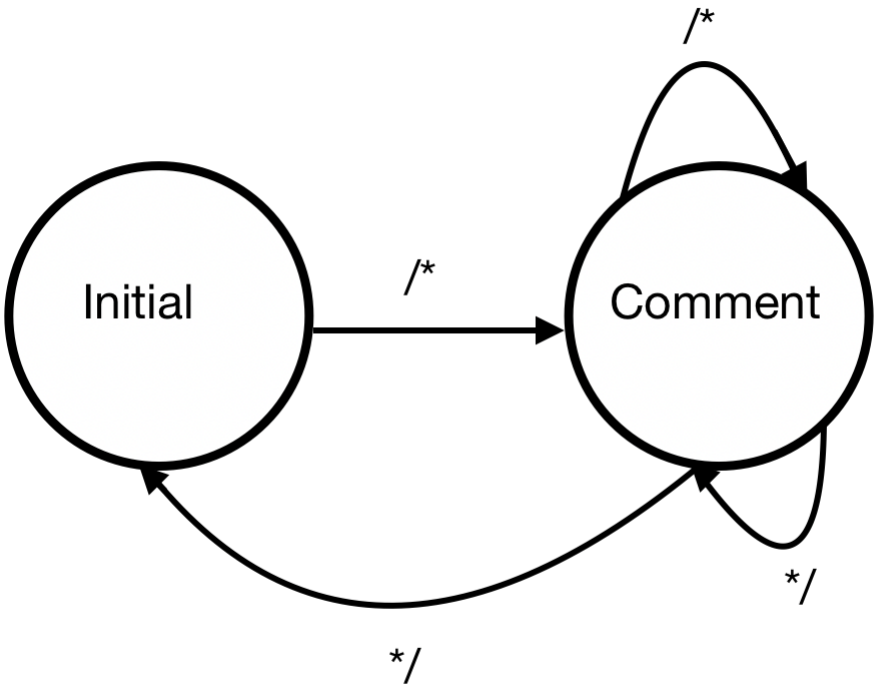
Punctuation Symbols

Tiger Expression	Token Name	Regular Expression
,	COMMA	,
:	COLON	:
;	SEMICOLON	;
(	LPAREN	(
)	RPAREN	)
[	LBRACK	[
]	RBRACK	]
{	LBRACE	{
}	RBRACE	}
.	DOT	.
+	PLUS	+
-	MINUS	-
*	TIMES	*
\	DIVIDE	\
=	EQ	=
<>	NEQ	<>
<	LT	<
<=	LE	<=
>	GT	>
>=	GE	>=
&	AND	&
|	OR	|
:=	ASSIGN	:=

Comment Handling

In order to implement nested comment handling, a variable called commentCounter is used to keep track of how many open comment (/*) and close comment (*/) expressions there are. When /* is parsed, 1 is added to the commentCounter and when */ is parsed, 1 is subtracted from the commentCounter. When the first /* is seen, the parser moves from state 'INITIAL' to state 'COMMENT'. The parser stays in state 'COMMENT' until all of the comments are closed (when the commentCounter is 0). The code also enables new line and tabs to be inserted into comments so that the parser can continue when these are encountered.

NFA for Comment Handling:

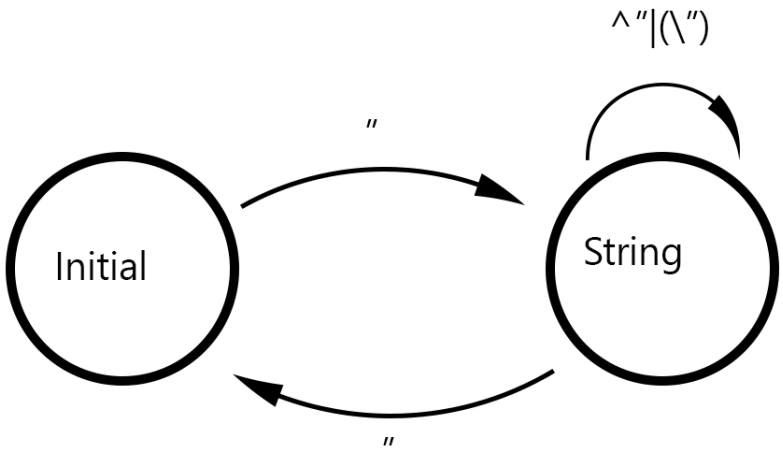


Regular Expressions for comments:
Based on the NFA for comments, we derived the regular expression "/*"(^"*/")*"*/" for a comment with no nested comments. (Note: The text inside "" is part of the expression, not the quotes themselves)

String Handling

Three variables- insideString, currString, and startString- are used to handle strings. The parser moves from the 'INITIAL' state to the 'STRING' state when a /" is encountered. When the 'STRING' state is entered, the insideString statement is set to 1, indicating that a string is currently open. currString is assigned the value of an empty string which is added to as characters within the string are parsed. The startString variable keeps track of the starting yypos of the string. The 'STRING' state is able to parse tab and new line escape sequences into their strings. It it also able to parse lowercase and uppercase alphabetical characters and numerical digits.

NFA for String Handling:



Regular Expressions for Strings:
Based on the NFA for strings, we derived the regular expression "(ˆ" |(\"))*"

Error Handling

When we parse illegal characters, we throw an error message saying "illegal character " and then the illegal character.

End of File Handling

When we parse the end of file (EOF) character, we perform checks to see if we are still in the middle of a comment or a string, which is illegal for a file. We throw error messages in such a scenario. If we are outside of any comments and strings when we parse the EOF character, the file is valid, and the parsing concludes.