Phakhin Deesee 5931049121

Compiler Report

1. **Scanner**

   Convert each line of input into lists - 10 A = 1 to be [‘10’, ‘A’, ‘=’, ‘1’] -, then pass it into Parser part.

2. **Parser**

   Convert gramma to LL1

   pgm := line pgm | EOF
   line := line_num stmt
   stmt := asgmnt | if | print | goto | stop
   asgmnt := id = exp
   exp := term exp0
   exp0 := + term | - term | EOF
   term := id | const
   if := IF cond line_num
   cond := term cond0
   cond0 := < term | = term
   print := PRINT id
   goto := GOTO line_num
   stop := STOP

   First set

   pgm = {line_num, EOF}
   line = {line_num}
   stmt = {id, IF, PRINT, GOTO, STOP}
   asgmnt = {id}
   exp = {id, const}
   exp0 = {+, -, EMPTY}
   term = {id, const}
   if = {IF}
   cond = {id, const}
   cond0 = {<, =}
   print = {PRINT}
   goto = {GOTO}
   stop = {STOP}

Phakhin Deesee 5931049121

Follow set

pgm = {EOF}
line = {line_num, EOF}
stmt = {line_num, EOF}
asgmnt = {line_num, EOF}
exp = {line_num, EOF}
exp0 = {line_num, EOF}
term = {+, -, line_num, EOF}
term = {+, -, line_num, EOF}
if = {line_num, EOF}
cond = {line_num}
exp0 = {line_num}
print = {line_num, EOF}
goto = {line_num, EOF}
stop = {line_num, EOF}

Rules

1. pgm := line pgm
2. pgm := EOF
3. line := line_num stmt
4. stmt := asgmnt
5. stmt := if
6. stmt := print
7. stmt := goto
8. stmt := stop
9. asgmnt := id = exp
10. exp := term exp0
11. exp0 := + term
12. exp0 := - term
13. exp0 := EMPTY
14. term := id
15. term := const
16. if := IF cond line_num
17. cond := term cond0

18. cond0 := < term
19. cond0 := = term
20. print := PRINT id
21. goto := GOTO line_num
22. stop := STOP

Parsing table

	line_ num	EOF	id	+	-	const	IF	<	=	PRI NT	GOT O	STO P
pgm	1	2										
line	3											
asgm nt			9									
exp0	13	13		11	12							
term			14			15						
if							16					
cond 0								18	19			
print										20		
goto											21	
stop												22
stmt			4				5			6	7	8
exp			10			10						
cond			17			17						

Prepare initial list StackLL1 = ['EOF','pgm'], a empty ParsedList and these sets - for convenience of coding -

- BCodeType (the types which appeared in Grammar)
- AlphaSet (alphabet set A-Z)
- Terminal set
- NextSet (list which derived from BCodeType)
- ParsingTable

Each token of list from Scaner part,

1. Check if token and top of StackLL1 (Last-in) not the same of the token
2. Pop 1 the last-in StackLL1 and push the derived term from the last pop term into StackLL1
3. Goto step 1 if it is the same term, else append tuple (top of StackLL1, token) into ParsedList, Pop 1 the last-in StackLL1 and goto next token
4. Repeat all steps until reach all tokens
5. If found that some term is not in BCodeType or cannot derived all tokens before StackLL1 empty throw error

Submit code on Github with instruction in README.md:
https://github.com/elm-kok/Compiler.git