Project 2 . Building a Parser

Aleksei Sorokin
Joshua Kelly
April 27, 2021

1 Introduction

The purpose of this program is to convert context free grammar into xml- like tree structure indicating the parse tree of the input program.

2 Data Structures

Ret: a struct is designed to help run through the given program and keep the desired type and string organized
Token: an array of integers used to distinguish what type of characters are currently present and what indent should be used.
Error Token: a binary number that indicates whether an error has occurred or not.
Transition table: A two-dimensional array that uses 18 rows and 14 columns that are acted upon by two integer fields: action and type.

		Space,Tab	newline	/	*	(	)	+	-	:	=	.	digit	letter	other
start	1	17	17	2	10	6	7	8	9	11	error	13	14	16	error
	2	div	div	3	4	div	div	div	div	div	div	div	div	div	div
	3	3	18	3	3	3	3	3	3	3	3	3	3	3	3
	4	4	4	4	5	4	4	4	4	4	4	4	4	4	4
	5	4	4	18	5	4	4	4	4	4	4	4	4	4	4
	6	lparen	lparen	lparen	lparen	lparen	lparen	lparen	lparen	lparen	lparen	lparen	lparen	lparen	lparen
	7	rparen	rparen	rparen	rparen	rparen	rparen	rparen	rparen	rparen	rparen	rparen	rparen	rparen	rparen
	8	plus	plus	plus	plus	plus	plus	plus	plus	plus	plus	plus	plus	plus	plus
	9	minus	minus	minus	minus	minus	minus	minus	minus	minus	minus	minus	minus	minus	minus
	10	times	times	times	times	times	times	times	times	times	times	times	times	times	times
	11	error	error	error	error	error	error	error	error	error	12	error	error	error	error
	12	assign	assign	assign	assign	assign	assign	assign	assign	assign	assign	assign	assign	assign	assign
	13	error	error	error	error	error	error	error	error	error	error	error	15	error	error
	14	number	number	number	number	number	number	number	number	number	number	15	14	number	number
	15	number	number	number	number	number	number	number	number	number	number	number	15	number	number
	16	identifier	identifier	identifier	identifier	identifier	identifier	identifier	identifier	identifier	identifier	16	16	identifier	identifier
	17	17	17	white_sp	white_sp	white_sp	white_sp	white_sp	white_sp	white_sp	white_sp	white_sp	white_sp	white_sp	white_sp
	18	comment	comment	comment	comment	comment	comment	comment	comment	comment	comment	comment	comment	comment	comment

3 Algorithms

Global variables:
token: ret type, used to store the current token

FP: *File type, points to the file
error: int type, used for indication of errors
level: int type, determines the size of the indent
num_tokens: int type, shows the number of tokens
tokens[300]: array of ret type, used to store tokens

Algorithm 1: Main

Given data: File name
Input: FP, the file pointer
Output: None

Plan(pseudocode):

```
Error = 0
Level = 0
FP:= open file F
If (file is not empty)
  |     add "<Program>" to context
  |     add current level to type
  |     if (reached end of file)
  |     | set type to 0
  |     End
  |     Else
  |     | call scan function
  |     End
  |
  |     If ( type is 12)
  |     | return an error
  |     End
  |     If (type is id, read, write, $$)
  |     | call stmt_list function
  |     | call the match($$) function
  |     Else
  |     | return an error
  |     End
  |     add"</Program>" to context string
  |     set token type to current level
  |     increase token number by 1
  |     closed the file pointer
  |
  |     if (error token is not zero)
  |     | print out error to the screen
  |     End
  |     Else
  |     | For each number of tokens
  |     | | Print out a space before each token
  |     | | Print out each token
```

```
|      | End
|      End
|

Else
| print File opening failed
End
End
```

Algorithm 2: scan

Given Data: file pointer t
Output Data: None

Plan(pseudocode):

```
Use struct ret to make t
Int cs = 1
Set string context to null
Set char ch to FP, file pointer
Reset FP
Set int chType to the translated vision of ch
While (location in translation table is 0)
|   set cs equal to the location in the translation table
|   set ch to the file pointer
|   at the file pointer to t. context
|   set ch to the file pointer
|   unset ch from the current file pointer location
|   set chType to the translated copy of ch
End
Set t.type to the type at the corresponding location in translation table according to cs and
chType

If (t.type is 9)
|    if(t.context is set to write
|      | set t.type to 11
|    End
|     if (t.context is set to read)
|      | set t.type to 10;
|     End
End

Return t to main

End
```

Algorithm 3: add_terminal

Input: None
Output: helps to increment context string for each token type

Plan(pseudocode):

```
Set the tokens[num_tokens].context to "<" int_to_token(token.type) + ">"
Set the tokens[num_tokens].type equal to level
Increase num_tokens by 1

Increase level by 1
Set tokens[num_tokens].context equal to token.context
Set tokens[num_tokens].type equal to level
 Increase num_tokens by 1
Decrease level by 1

Set the tokens[num_tokens].context to "<" int_to_token(token.type) + ">"
Set the tokens[num_tokens].type equal to level
Increase num_tokens by 1
End
```

Algorithm4: match

Input: The current token type in integer form
Output: N.A.

Plan(pseudocode):

```
Increase level by 1
If (given number is equal to the token type
|       if (the token type is not equal to 0)
|       | call the add_terminal function
|       End
|       set get_new to 0
|       while (get_new and FP are equal to zero
|       |    if (file pointer at the end of the stream is 1)
|       |    | set token type to 0
|       |    | set get_new to 1
|       |    End
|       |    else
|       |    |  set token to call the scan function
|       |    | if (token type is less than or equal to 12 or equal to 0)
|       |    | set get_new to 1
|       |   End
|      End
End
Else
| set the error token to 1
```

| Decrease level by 1
End

End

Algorithm 5: mult_op

Given data: the current token type
Input: N.A.
Output: Adds mult_op to context string if conditions are met

Plan(pseudocode):

Increase level by 1
Set tokens[num_tokens].Context to read "<Mult_op>"
Set tokens[num_tokens].type to the same as the current level
Increase num_tokens by 1

If (the current token type is *)
| call the match(*) function
End
Instead, if (the current token type is /)
| call the match(/) function
End
Else
| set error token to 1
End
Increase level by 1
Set tokens[num_tokens].Context to read "</Mult_op>"
Set tokens[num_tokens].type to the same as the current level
Increase num_tokens by 1
Decrease the current level by 1
End

Algorithm 6: add_op

Given data: N.A.
Input: Called when the right conditions are met
Output: Adds "<add_op> to current locations within the context string

Plan(pseudocode):

Increase level by 1

```
Set tokens[num_tokens].context equal to "<add_op>
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
If (the current token type is +)
| call match(+) function
End
Instead, if (the current token type is -)
          | call the match(-) function
         End
Else
| set error token to 1
End
Set tokens[num_tokens].context equal to "</add_op>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Decrease current level by 1
End
```

Algorithm 7: Factor

Given: N.A.
Input: Only called when correct conditions are met
Output: adds factor to current location in the context string

Plan(pseudocode):

```
Increase level by 1
Set tokens[num_tokens].context equal to "<factor>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
If (the current token type is id)
| call match(id) function
End
Instead, if (the current token type is equal to number)
          | call the match(number) function
         End
Instead, if (The current token type is ()
         | call the match('(') function
         | call the expr function
         | call the match(')') function
        End
Else
| set error token to 1
End
Set tokens[num_tokens].context equal to "</factor>"
```

Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Decrease current level by 1
End

Algorithm 8: Factor_tail

Given: N.A.
Input: Only called when the correct conditions are met
Output: adds factor tail to the current location in the context string

Plan(pseudocode):

Increase level by 1
Set tokens[num_tokens].context equal to "<factor_tail>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
If (the current token type is *, /)
| call the mult_op function
| call the factor function
| call the factor_tail function
End
Instead, if (the current token type is +, -,), id, read, write, $$:)
|
End
Else
| set error token to 1
End
Set tokens[num_tokens].context equal to "</factor_tail>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Decrease current level by 1
End

Algorithm 9: Term

Given: N.A.
Input: Only called when the correct conditions are met
Output: adds term to the current location in the context string

Plan(pseudocode):

Increase level by 1
Set tokens[num_tokens].context equal to "<term>"
Set tokens[num_tokens].type equal to the current level

Increase num_tokens by 1
If (the current token type is id, number, ()
| call the factor function
| call the factor_tail function
End
Else
| set error token to 1
End
Set tokens[num_tokens].context equal to "</term>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Decrease current level by 1
End

Algorithm 10: Term_tail

Given: N.A.
Input: Only called when the correct conditions are met
Output: adds term tail to the current location in the context string

Plan(pseudocode):

Increase level by 1
Set tokens[num_tokens].context equal to "<term_tail>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
If (the current token is +, -)
| call the add_op function
| call the term function
| call the term_tail function
End
Instead, if (the current token type is), id, read, write, $$)
|
End
Else
| set error token to 1
End
Set tokens[num_tokens].context equal to "</termr_tail>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Decrease current level by 1
End

Algorithm 11: expr

Given: N.A.
Input: Only called when the correct conditions are met

Output: adds expr to the current location in the context string

Plan(pseudocode):

```
Increase level by 1
Set tokens[num_tokens].context equal to "<expr>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
If (the current token type is id, number, (
| call the term function
| call the term_tail function
End
Else
| set error token to 1
End
Set tokens[num_tokens].context equal to "</expr>
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Decrease current level by 1
End
```

Algorithm 12: stmt

Given: N.A.
Input: Only called when the correct conditions are met
Output: adds stmt to the current location in the context string

Plan(pseudocode):

```
Increase level by 1
Set tokens[num_tokens].context equal to "<stmt>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Switch (the current token type)
|    for the current case being id
|     | call the match(id) function
|     | call the match(:=) function and send 7 to it
|     | call the expr function
|     | break out of switch
|     end
|    for the current case being read
|     | call the match(read) function
|     | call the match(id) function
|     | break out of switch
|    End
|    for the current case being equal write
|     | call the match(write) function
```

```
|     | call the expr function
|     | break out of switch
|     End
|    If not condition is met yet
|      | set error token to 0
|     End
End
Set tokens[num_tokens].context equal to "</stmt>
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Decrease current level by 1
End
```

Algorithm 13: stmt_list

Given: N.A.
Input: Only called when the correct conditions are met
Output: adds stmt_list to the current location in the context string

Plan(pseudocode):

```
Increase level by 1
Set tokens[num_tokens].context equal to "<stmt_list>"
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
If (the current token type is id, read, write)
| call the stmt function
| call the stmt_list function
End
Instead, if (the current token type is $$
            |
           End
Else
| set error token to 1
End
Set tokens[num_tokens].context equal to "</stmt_list>
Set tokens[num_tokens].type equal to the current level
Increase num_tokens by 1
Decrease current level by 1
End
```

4 Test cases

For the purpose of this class, we have selected a small test page call inp.txt included within the zipped file. The contents of this page are:

Test case 1)

read A

```
C:\Alex\university\programming
<Program>
  <stmt_list>
    <stmt>
      <read>
        read
      </read>
      <id>
        A
      </id>
    </stmt>
    <stmt_list>
    </stmt_list>
  </stmt_list>
</Program>

Terminated with return code 0
```

```
Test case 2)
A:= 56 ba + 3
```

```
C:\Alex\university\programming
error.

Terminated with return code 0
Press any key to continue ...
```

```
Test case 3)

read A
read B
sum := A + B
write sum
write sum / 2
```

C:\Alex\university\programming

```
<Program>
  <stmt_list>
    <stmt>
      <read>
        read
      </read>
      <id>
        A
      </id>
    </stmt>
    <stmt_list>
      <stmt>
        <read>
          read
        </read>
        <id>
          B
        </id>
      </stmt>
      <stmt_list>
        <stmt>
          <id>
            sum
          </id>
          <assign>
            :=
          </assign>
          <expr>
            <term>
              <factor>
                <id>
                  A
                </id>
              </factor>
              <factor_tail>
              </factor_tail>
            </term>
            <term_tail>
              <add_op>
                <plus>
                  +
```

C:\Alex\university\programming

```
                <plus>
                  +
                </plus>
              </add_op>
              <term>
                <factor>
                  <id>
                    B
                  </id>
                </factor>
                <factor_tail>
                </factor_tail>
              </term>
              <term_tail>
              </term_tail>
            </term_tail>
          </expr>
        </stmt>
        <stmt_list>
          <stmt>
            <write>
              write
            </write>
            <expr>
              <term>
                <factor>
                  <id>
                    sum
                  </id>
                </factor>
                <factor_tail>
                </factor_tail>
              </term>
              <term_tail>
              </term_tail>
            </expr>
          </stmt>
          <stmt_list>
            <stmt>
              <write>
                write
```

C:\Alex\university\programming

```
        </stmt>
        <stmt_list>
          <stmt>
            <write>
              write
            </write>
            <expr>
              <term>
                <factor>
                  <id>
                    sum
                  </id>
                </factor>
                <factor_tail>
                  <mult_op>
                    <division>
                      /
                    </division>
                  </mult_op>
                  <factor>
                    <number>
                      2
                    </number>
                  </factor>
                  <factor_tail>
                  </factor_tail>
                </factor_tail>
              </term>
              <term_tail>
              </term_tail>
            </expr>
          </stmt>
          <stmt_list>
          </stmt_list>
        </stmt_list>
      </stmt_list>
    </stmt_list>
  </stmt_list>
</Program>
```

Type here to search ENG 7:43 PM 4/17/2021