MiniC Language Manual

Name: Swetanjal Dutta
Roll Number: 20171077

31/08/2020

1 Macro Syntax Specification using Context Free Grammars

1.1 Meta Notation:

- <foo> means foo is a non terminal.
- **foo**(in bold font) means foo is a terminal i.e a token.
- [x] means zero or one occurrence of x i.e x is optional. Note that ***brackets in quotes*** i.e '[' and ']' are terminals.
- x^* means zero or more occurrences of x.
- x^+ means one or more occurrences of x.
- x^+, means a comma separated list of one or more xs. Comma is a terminal.
- {} i.e large braces are used for grouping. Note that ***braces in quotes*** i.e '{' and '}' are terminals.
- | separates alternatives.
- Punctuation like round brackets, braces, semicolons and commas are terminals. Please note that they have not been written in bold in the CFG.

1.2 Production Rules:

1. <program> → <decl>$^+$
2. <decl> → <var_decl> | <method_decl>
3. <var_decl> → <type> <identifier>$^+$, ;
4. <method_decl> → {<type> | **VOID**} **ID** ([{<type> <identifier>}$^+$,]) <block>
5. <block> → '{' <var_decl>* <statement>* '}'
6. <type> → **INT** | **UINT** | **BOOL** | **CHAR**
7. <statement> → <assignment>$^+$, ;
   | <method_call>;
   | **IF** (<expr>) <block> [**ELSE** <block>]
   | **FOR** ([<assignment>$^+$,]; [<expr>$^+$,]; [<assignment>$^+$,]) <block>
   | **WHILE** (<expr>) <block>
   | **BREAK**;
   | **CONTINUE**;
   | <block>
   | **RETURN** [<expr>];
   | **PRINT** (<expr>);

8. <assignment> → <identifier> **ASSIGN** <expr>
9. <method_call> → **ID** ([<expr>$^{+}$,])
10. <expr> → <identifier>
    | <expr> <arithmetic_op> <expr>
    | <expr> <relational_op> <expr>
    | <expr> <conditional_op> <expr>
    | <expr> <equality_op> <expr>
    | <expr> **THEN** <expr> **OTHERWISE** <expr>
    | <literal>
    | <method_call>
    | **NOT** <expr>
    | **NEGATE** <expr>
    | (<expr>)
    | **READ_INT**()
    | **READ_CHAR**()
    | **READ_BOOL**()
11. <identifier> → **ID** | **ID**{'[' <expr> ']'}*
12. <literal> → **INT_LIT** | **FLOAT_LIT** | **CHAR_LIT** | **STRING_LIT** | <bool_lit>
13. <bool_lit> → **TRUE** | **FALSE**
14. <arithmetic_op> → **ADD** | **SUB** | **MUL** | **DIV** | **MOD**
15. <relational_op> → **LT** | **GT** | **LE** | **GE**
16. <conditional_op> → **AND** | **OR**
17. <equality_op> → **EQ** | **NE**

1.3 Start Symbol:

- program

2 Micro Syntax Specification using Regular Expressions

2.1 Meta Notation:

- Token Type → Lexeme

2.2 Rules:

1. FALSE → false
2. TRUE → true
3. NOT → !
4. NEGATE → ˜
5. VOID → void
6. INT → int
7. UNINT → uint
8. CHAR → char
9. BOOL → bool
10. THEN → ?

11. OTHERWISE → :
12. FOR → for
13. WHILE → while
14. IF → if
15. ELSE → else
16. BREAK → break
17. CONTINUE → continue
18. RETURN → return
19. ADD → +
20. SUB → -
21. MUL → *
22. DIV → /
23. MOD → %
24. LT → <
25. GT → >
26. LE → <=
27. GE → >=
28. AND → &&
29. OR → ||
30. EQ → ==
31. NE → !=
32. ASSIGN → =
33. PRINT → print
34. READ_INT → read_int
35. READ_CHAR → read_char
36. READ_BOOL → read_bool
37. , → ,
38. ; → ;
39. (→ (
40.) →)
41. { → {
42. } → }
43. INT_LIT → $\left[0\text{-}9\right]\left[0\text{-}9\right]^*$
44. FLOAT_LIT → $\left[0\text{-}9\right]\left[0\text{-}9\right]^*(.\left[0\text{-}9\right]\left[0\text{-}9\right]^*)?$
45. CHAR_LIT → ‘[a-zA-Z0-9 _.,;]’
46. ID → [a-zA-Z_][a-zA-Z0-9_]*
47. STRING_LIT → “[a-zA-Z0-9 _.,;]*”

3 Lexical Considerations

4 Semantic Checks