

Actividad 3.3 - Context Free Grammar

Emiliano Cabrera - A01025453

Do Hyun Nam - A01025276

29 de abril de 2022

Prof. Gilberto Echeverría

Actividad 3.3

Escribe la notación BNF y EBNF para la gramática necesaria para definir módulos y funciones en Elixir.

BNF

Functions

```
<function> ::= def <variable>(<variable-expression>), do:
<single-function-expression> end | def
<variable>(<variable-expression>) do <function-expression> end |
def <variable>, do: <single-function-expression> end | def
<variable> do <function-expression> end
```

```
<variable> ::= <letter><variable> | _<variable> |
<letter> | <letter><number> | <mac><number>
```

```
<letter> ::= 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
```

```
<number> ::= <digit> | <digit><number>
```

```
<digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

```
<variable-expression> ::= <variable> |
<variable>,<variable-expression>
```

```
<single-function-expression> ::= <number> | <variable> |
<number><operation> | <real><operation> | true | false
```

```
<operation> ::= <operator><number> | <operator><real>
```

```
<real> ::= <number>.<number>
```

```
<operator> ::= + | - | / | *
```

```
<function-expression> ::=
<single-function-expression><function-expression> |
<single-function-expression>
```

```

Modules
    <module> ::= defmodule <masc-variable> do <mult-function>
end

    <masc-variable> ::= <masc><variable>

    <masc> ::= 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

    <mult-function> ::= <function> | <function> end
<mult-function>

```

EBNF

```

    Functions

    FUNCTION ::= def VARIABLE VAR-EXPRESSION, do:
FUNC-EXPRESSION end | def VARIABLE VAR-EXPRESSION do
{FUNC-EXPRESSION} end

    VARIABLE ::= [{'_'}]LETTER[VARIABLE][{DIGIT}]

    LETTER ::= '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'

    DIGIT ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' |
'9' | '0'

    VAR-EXPRESSION ::= VARIABLE['',{VARIABLE}]

    FUNC-EXPRESSION ::= {{DIGIT}} | VARIABLE |
{{DIGIT}OPERATION} | {REAL{OPERATION}} | true | false |
{FUNC_EXPRESSION}

    REAL ::= {DIGIT}'.'{DIGIT}

    OPERATION ::= OPERATOR{DIGITS} | OPERATOR REAL

    OPERATOR ::= '+' | '-' | '/' | '*'

```

```

Modules

    MODULE ::= defmodule MASC-VARIABLE do {FUNCTION} end

    MASC-VARIABLE ::= MASC[{VARIABLE}]

    MASC ::= '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'

```