Universidad del Valle de Guatemala

Facultad de Ingeniería

Proyecto 2: Generador de analizador Léxico

Diseño de Lenguajes de Programación

José Rodrigo Martínez Zúñiga

15163

Guatemala,

2021

Diseño

La solución propuesta para este proyecto esta basada en TypeScript debido a la ventaja que se pueden utilizar funciones de ES7 y es mas fácil manejar 'tipos' para las variables. El diseño propuesto esta basado en las diferentes declaraciones que presenta CoCol. Las declaraciones que presenta CoCol son las siguientes:

```
Vocabulario de Cocol

ident = letter {letter | digit}.
number = digit {digit}.
string = '"' {anyButQuote} '"'.
char = '\'' anyButApostrophe '\''.

Sintaxis de Cocol

Cocol = "COMPILER" ident
       ScannerSpecification
       ParserSpecification
"END" ident '.'.

ScannerSpecification =
       ["CHARACTERS" {SetDecl}]
       ["KEYWORDS" {KeyworDecl}]
       ["TOKENS" {TokenDecl}]
       {WhiteSpaceDecl}.

SetDecl             = ident '=' Set.
Set                 = BasicSet { ('+'|'-') BasicSet }.
BasicSet            = string | ident | Char [".." Char].
Char                = char | "CHR" '(' number ')'.
KeywordDecl         = ident '=' string '.'
TokenDecl           = ident ['=' TokenExpr ] ["EXCEPT KEYWORDS"] '.'.
TokenExpr           = TokenTerm {'|' TokenTerm }.
TokenTerm           = TokenFactor {TokenFactor}
TokenFactor         = Symbol
                    | '(' TokenExpr ')'
                    | '[' TokenExpr ']'
                    | '{' TokenExpr '}'.
Symbol              = ident | string | char
WhiteSpaceDecl      = "IGNORE" Set
```

Figura 1. Especificaciones de CoCol.

En el código se crearon clases para cada declaración presentada en el archivo de especificación. Las clases que se utilizaron son las siguientes:

1. Para "Characters"
    a. CharactersSetDecl: Este representa el **SetDecl**
    b. CharacterSet: Este representa **Set**.
    c. BasicSet: Este representa **BasicSet**.
    d. CharToChar: Este representa CHR()..CHR()
    e. Char: Este representa un character, ya sea un String o un codigo ASCII.
2. Para "Keywords"
    a. KeywordDecl: Este representa **KeywordDecl**.
3. Para "Tokens"
    a. TokenDeclaration: Este representa **TokenDecl**.
    b. Expression: Este representa **TokenExpr**.

c. Term: Este representa **TokenTerm**.
d. Factor: Este representa **TokenFactor**.
e. Symbol: Este representa **Symbol**.

4. WhiteSpaceDeclaration: Este representa **WhiteSpaceDecl**.

Cada una de estas clases tiene dos funciones principales: set{clase}() y toRegularExpression(). La funcion set{clase}() recibe un string y genera los atributos de la clase. La funcion de toRegularExpression() genera una expresión regular dependiendo de los atributos de la clase. El siguiente diagrama es el diagrama de flujo que sigue el código:

Para generar el Scanner se utilizan dos archivos de Python, la primera parte (base1.py) contiene código de creación de variables y metodos a utilizar, y la segunda parte (base2.py) contiene codigo para probar verificar los tokens dentro de un string. Lo que se hace para crear el Scanner de un lenguaje descrito en un archivo ATG es crear un archivo python con el nombre del compilador. En ese archivo se pega el código de Python de base1,py, despues se pega codigo para crear el AFD (creación de nodos y transiciones), y por ultimo se pega el codigo que se encuentra en base2.py.

Problemas enfrentados

El mayor problema con el que se enfrento y no se pudo resolver es la utilización de parentesis como parte de los carateres del lenguaje, como se encuentra en el archivo de CoCol.ATG. La manera en la que se piensa resolver eso es poder definir caracteres de parentesis para poder generar la expresion regular y leerla al momento de generar el arbol sintactico. De esta manera yo puedo decir que el carácter de apertura puede ser un “¡” y el de cierre es “!”, de esta manera no se cruzan al momento de crear el arbol sintactico.

Conclusiones

- Poder separar todos los componentes por clases hace la lectura mas sencilla de comprender e implementar.
- Python puede no ser el mejor lenguaje para generar el scanner porque no permite crear strings en diferentes lineas de codigo, el string tiene que estar en la misma linea.
- Es importante terminar los proyectos dependientes para facilitar el desarrollo del siguiente proyecto.