Lenguajes y Máquinas

José David Romero - 202222606

Sofia Losada – 20221008

Documentación P0 Lenguajes y Máquinas

En el P0 se desarrolló un algoritmo en Python que identifica si la sintaxis de una serie de instrucciones que recibe es correcta. De esta manera, se verifican los nombres de las funciones y sus respectivos parámetros, previamente definidos y de ser así se analiza si la estructura dada corresponde a la establecida. De no ser así el algoritmo retorna el string "defectuoso".

El algoritmo tiene la estructura MCV. Esto con el fin de construir una cadena operativa para realizar operaciones interconectadas conversando su respectiva independencia.

Model:

Primero, se establecen dos estructuras, la primera el diccionario de funciones, donde se crean las funciones move, skip, null y su respectiva cantidad de parámetros, estas funciones tienen un único parámetro de tipo entero. En el set variables globales se crean todas variables que se pueden utilizar en las diferentes funciones a implementar.

Después se implementan las funciones "paréntesis_okay" y "delimitador", en donde se verifica que los paréntesis en el archivo de texto estén completos y correctamente implementados, pues estos delimitan el comienzo y el final de una instrucción.

Luego se implementan las funciones "defvar", "put", "pick", "turn", "face", "run_dirs", "move_dir" y "move_face" en donde se recorren las funciones y se verifica que los parámetros y variables dados se encuentren en el set de variables globales creado, al igual que respeten la cantidad de paréntesis.

Después el algoritmo ejecuta la función "llamar_funcion", "recorrer_llamado_funcion" y "funcion_bien_definida". La cual valida la sintaxis de una función llamada utilizando la representación tokenizada, asegurándose de que la llamada incluya el número correcto de parámetros y que estén en el orden esperado. Teniendo esto, "recorrer_llamado_funcion" asocian el token a la función correspondiente

y “funcion_bien_definida” verifica que la función se declaró correctamente, teniendo en cuenta los parámetros y el cuerpo de la función.

Finalmente, las funciones identificador_llamado y “analizador” ejecutan un algoritmo que funciona como enrutador en donde toma el primer token de la lista y clasifica dependiendo de la función a la que está asociado y después “analizador” se encarga de validar que las funciones cumplan con los criterios de balance de delimitadores y coherencia en el uso de tokens, de no ser así los cataloga como defectuoso.

Controller:

Primero se establecen cuatro listas:

```
estructuras_de_control = ["IF","LOOP","REPEAT"]
funciones = ["=","MOVE","SKIP","TURN","FACE","PUT","PICK","MOVE-DIR","RUN-DIRS","MOVE-FACE","NULL"]
condiciones= ["FACING?","BLOCKED?","CAN-PUT?","CAN-PICK?","CAN-MOVE?","ISZERO?","NOT"]
variables=[":LEFT",":RIGHT",":AROUND",":NORTH","SOUTH",":EAST",":WEST",":FRONT",":BACK",":BALLOONS",":CHIPS","DIM","MYXPOS","MYYPOS",
           "MYCHIPS","MYBALLOONS","BALLOONSHERE","CHIPSHERE","SPACES"]
```

Después se ejecutan las funciones ”delimitador_cadenas“ y “delimitador_instruccion“, las cuales tienen en cuenta paréntesis abiertos y cerrados y espacios para determinar el fin de una cadena y de una instrucción.

Luego se ejecuta la “funcion abrir_archivo”, la cual se encarga de recibir la ruta del archivo de tipo txt donde se encuentran las instrucciones a ser analizadas.

Al terminar, se ejecuta la función “instrucciones”, la cual procesa una cadena de texto y elimina los espacios y saltos de línea innecesarios, después verifica la correcta delimitación por paréntesis y toma las instrucciones encerradas en paréntesis. Al no encontrar errores, retorna la lista cadenas, de lo contrario retorna un mensaje de error.

Después se implementa la función “tokenizador” la cual se encarga de realizar el análisis léxico de las instrucciones recibidas y las convierte en tokens que representan los diferentes componentes sitácticos.

Para esto se implementa un ciclo while que itera sobre cada carácter de cadena. De esta manera si encuentra un “(“ o “)” añade el carácter correspondiente a la lista de tokens inicializada, con el valor asociado vacío. Después continúa la iteración y utiliza delimitador_cadenas para determinar el final de la “subcadena” sobre la que está iterando y dependiendo de su contenido la clasifica en alguna de las estructuras creadas al principio, pues comparó si la cadena sobre la que iteraba se encontraba en alguna de las listas. En caso de que en la iteración encuentre un “defvar” le asigna el token respectivo y busca la variable asociada para asignarle el token “var”. Si encuentra “defun” implementa lo mismo que “defvar”, pero como esta busca definir funciones y no variables se le asigna un token diferente y la lista de valores les asigna el token “var”. Por último, si encuentra un caracter que no coincida con ningun elemento en las estructuras iniciales retorna un mensaje de error.

Finalmente se implementan las funciones ”tokenizar" la cual en principio determina si la lista dada por parámetro contiene un mensaje de error de ser así se tokeniza. De lo contrario, invoca la función tokenizador y devuelve una lista con los tokens representando lo composición sintáctica de la instrucción.

Vista:

En la vista se encuentra la interacción directa con el usuario donde se le presenta el menú y le da la posibilidad de dar el archivo de txt a analizar.

Mejoras:

Después de desarrollar el P0 consideramos que podríamos mejorar el resultado si ejecutamos implementaciones óptimas para analizar los caracteres, pues al analizar archivos con muchos elementos, la complejidad aumentaría considerablemente, lo que dificulta la ejecución del análisis. De igual manera, consideramos que la implementación mejoraría mucho si al final el usuario supiera cuál fue el error, ya que el objetivo del código es que el robot realice ciertas acciones.

Por otro lado, en la implementación vemos como oportunidad de mejora el reconocimiento de los números de más de un dígito, pues de esta manera el algoritmo puede reconocer el valor si alguno de los parámetros numéricos es de más de un dígito. Igualmente, en la implementación de los condicionales, podríamos mejorar cómo identificamos la cantidad de valores exactos dentro de la función como parámetros.