Manual de usuario

Por Hernán José Cervera Manzanilla

1. Funcionamiento de los analizadores

1.1. Analizador léxico

Dado un archivo de tipo MIO, hallado en el mismo directorio que el JAR *analex*, si el programa se ejecuta desde la línea de comandos y el nombre del archivo es colocado como único parámetro, el programa entonces lee el archivo y almacena cada línea en una lista (*java.util.List*) . Luego, cada elemento de la lista, que representa una línea del archivo MIO, es partido en lexemas reconocibles por el analizador léxico. Esto se almacena en una lista (*java.util.ArrayList*) de vectores de cadenas. El programa entonces itera sobre todos los elementos de todos los vectores de la lista, reconociendo qué tipo de lexema es y produciendo los archivos temporales.

Si el programa se ejecuta sin errores, la consola no reporta nada y se generan los archivos. Si el programa se ejecuta con errores, se reporta en la consola en qué línea el archivo tiene un token no válido y no se generan archivos temporales.

1.2. Analizador sintáctico

Dado un archivo de tipo LEX, hallado en el mismo directorio que el JAR *anasin*, producido por el analizador léxico, si el programa se ejecuta desde la línea de comandos y el nombre del archivo es colocado como único parámetro, entonces el programa puede ejecutarse. Comienza cargando cada línea del archivo LEX en una lista (*java.util.ArrayList*). Esta lista es pasada a un analizador sintáctico predictivo implementado recursivamente. Para poder crear un analizador de este tipo, fue necesario convertir la gramática original en una de tipo LL(1). De esta manera, el analizador sintáctico evita el no determinismo, sorteando así las complicaciones de implementación del backtracking que hubiera existido de otra forma.

Si el programa se ejecuta sin errores, la consola reporta que se aceptó la cadena extraída del archivo LEX. Si el programa se ejecuta con errores, se reporta en la consola que hubo un error, pero no reporta cuál.

2. Casos de uso

2.1. Analizador léxico

2.1.1. Caso exitoso # 1

2.1.1.1. Archivo de entrada

factorial.mio

```
# Programa que calcula el factorial de un número
PROGRAMA factorial
# VarX acumula los productos por iteración
VarX = 0x1
# VarY contiene el iterador del factor
VarY = 0x0
LEE Num
# Aplica Num! = 1 * 2 * 3 * ... * Num
REPITE Num VECES
VarY = VarY + 0x1
VarX = VarX * VarY
FINREP
IMPRIME "Factorial de "
IMPRIME Num
IMPRIME " es "
IMPRIME VarX
FINPROG

```

2.1.1.2. Archivos de salida

factorial.lex

```
PROGRAMA
[id]
[id]
=
[val]
[id]
=
[val]
LEE
[id]
REPITE
[id]
VECES
[id]
=
[id]
[op_ar]
[val]
[id]
=
[id]
[op_ar]
[id]
FINREP
IMPRIME
[txt]
IMPRIME
[id]
IMPRIME
[txt]
IMPRIME
[id]
FINPROG

```

factorial.sim

```
IDS
factorial,ID01
VarX,ID02
VarY,ID03
Num,ID04

TXT
"Factorial de ",TX01
" es ",TX02

VAL
0x1,1
0x0,0

```

2.1.2. Caso exitoso # 2

2.1.2.1 Archivo de entrada

si.mio

```
PROGRAMA pruebaSi
SI varx < 0x0A ENTONCES
IMPRIME 0x01
SINO
IMPRIME 0xAD3
LEE vary
FINSI
FINPROG
```

2.1.2.2. Archivos de salida

si.lex

```
PROGRAMA
[id]
SI
[id]
[op_rel]
[val]
ENTONCES
IMPRIME
[val]
SINO
IMPRIME
[val]
LEE
[id]
FINSI
FINPROG
```

si.sim

```
IDS
pruebaSi,ID01
varx,ID02
vary,ID03

TXT

VAL
0x0A,10
0x01,1
0xAD3,2771
```

2.1.3 Caso de fallo # 1

2.1.3.1. Archivo de entrada

fallo.mio

```
PROGRAMA fallo_Lex
var = 0x2A
REPITE var VECES
IMPRIME
FINREP
FINPROG
```

2.1.3.2. Salida

```
C:\Users\hjcer\Desktop\Analizadores>java -jar analex.jar falloLex.mio
Invalid token at line: 1
```

2.2. Analizador sintáctico

2.2.1. Caso exitoso # 1

2.2.1.1. Archivo de entrada

factorial.lex

```
PROGRAMA
[id]
[id]
=
[val]
[id]
=
[val]
LEE
[id]
REPITE
[id]
VECES
[id]
=
[id]
[op_ar]
[val]
[id]
=
[id]
[op_ar]
[id]
FINREP
IMPRIME
[txt]
IMPRIME
[id]
IMPRIME
[txt]
IMPRIME
[id]
FINPROG

```

2.2.1.2. Salida

```
C:\Users\hjcer\Desktop\Analizadores>java -jar anasin.jar factorial.lex
The word is accepted
```

2.2.2. Caso de exitoso # 2

2.2.2.1. Archivo de entrada

si.lex

```
PROGRAMA
[id]
SI
[id]
[op_rel]
[val]
ENTONCES
IMPRIME
[val]
SINO
IMPRIME
[val]
LEE
[id]
FINSI
FINPROG

```

2.2.2.2. Salida

```
C:\Users\hjcer\Desktop\Analizadores>java -jar anasin.jar si.lex
The word is accepted
```

2.2.3. Caso de fallo # 1

2.2.3.1. Archivo de entrada

falloSin.lex

```
PROGRAMA falloSin
var = 0x2A
REPITE var VECES
IMPRIME
FINREP
FINPROG

```

2.2.3.2. Salida

```
C:\Users\hjcer\Desktop\Analizadores>java -jar anasin.jar falloSin.lex
ERROR: No match
```