Lenguajes Formales

Practica 1

Ismael El Fellah-Idrissi Seliga

Lluís Colom

Gabriel Garcia Rodriguez

Diseño: definición de Regex principales

Para poder realizar el análisis léxico de un fichero xml, primero, tal como indica el enunciado, se ha procedido a definir las expresiones regulares de los elementos individuales para posteriormente concatenarlas y así poder identificar elementos completos con mayor flexibilidad.
En total, se han definido cinco tipos de expresión regular siguiendo los requisitos indicados en el enunciado:

```
APERTURA: \<[a-zA-Z_]+>
CIERRE: \<\/[a-zA-Z_]+>
ATRIBUTO: [ ]*[a-zA-Z_]+[ ]*=[ ]*["'][a-zA-Z_]+["']
TEXTO: ([^><&"']+)
APERTURAS CON ATRIBUTOS: \<[a-zA-Z_]+
```

Cosas a tener en cuenta en cuanto al diseño general del analizador léxico:

- Cabe decir que cuando contabilizamos el número de caracteres de cada texto correspondiente a un elemento simple se usa la variable global *yyleng* propia de flex, la cual puede llegar a generar inexactitudes en el recuento ya que se admiten caracteres en codificación UTF-8, hecho que implica un mayor número de bytes por carácter y en consecuencia una contabilización algo superior de dicho número de caracteres.
- La expresión regular aperturas con atributos hace referencia a aquellas aperturas que contengan uno o más atributos justo antes del carácter de cierre de tag ‘>’. Básicamente es la misma expresión que la de las aperturas simples pero sin el carácter de cierre de tag.

NOTA: Flex no trabaja con expresiones regulares ‘modernas’, es por eso que se puede apreciar el uso del carácter ‘\’ propio del programa asi como el operando /, que solo matcheará la Regex en caso de ir precedida de lo que indiquemos (símbolo ‘<’ en caso de las aperturas).

Contabilización usando dichos Regex

Usando las Regex anteriores se han definido una serie de reglas que identifican los siguientes elementos:

{APERTURA}/{CIERRE}

Esta expresión resultará en match cuando encontremos un cierre precedido immediatamente de una apertura. En este caso, los caracteres que se correspondan al cierre no seran procesados por la expresión regular, quedando libres para que otras expresiones regulares puedan hacerles match. Se incrementará el contador de aperturas y el contador de elementos vacíos.

{APERTURATRIB}/[ ]+[^>]+>{CIERRE}

Esta expresión detectará elementos vacíos con atributos que puedan existir en el fichero XML. Básicamente, se detectarà un tag de apertura, seguido de un conjunto de caracteres (hasta encontrarnos con un “>”: la frontera del tag de apertura) y finalmente con un tag de cierre. Con cada match se incrementará el contador de aperturas, el contador de elementos vacíos y el contador de tags con uno o más atributos.

Como en el caso anterior, los carácteres que hagan match con el patrón a partir de APERTURATRIB no seran consumidos, quedando disponibles para otras expresiones regulares.

{APERTURA}/{TEXTO}{CIERRE} { apertura++; elementosimple++;}

Esta expresión detectará elementos simples sin atributos, formados por un tag de apertura, un texto y finalizando con un tag de cierre.

Cada match incrementará los contadores de apertura y elemento simple.

Como en los casos anteriores, los carácteres que hagan match a partir del tag de apertura no seran consumidos, quedando disponibles para otras expresiones regulares.

{APERTURATRIB}/[ ]+[^>]+>{TEXTO}{CIERRE}

Esta expresión detectará elementos simples con uno o más atributos.

Con cada match, se incrementará el contador de elementos simples, aperturas y tags con uno o más atributos.

Al igual que los casos anteriores, los elementos a continuación del tag de apertura no seran consumidos.

{CIERRE} { cierre++; }

La expresión detectará los tags de cierre que puedan existir en el documento XML procesado.

Con cada match se incrementará el contador de tags de cierre.

{ATRIBUTO} { atributo++; }

Esta expresión detectará los atributos que los tags de apertura puedan contener. Con cada match se incrementará el contador de atributos.

{TEXTO}/<\/

Esta expresión detectará en el documento XML los textos que puedan aparecer sin contener los elementos reservados del lenguaje. Con cada coincidencia se incrementará el contador de textos del documento. Además, se incrementará la variable que controla el número de caracteres totales de los textos con el numero de caracteres que contenga el texto.

Pruebas realizadas

Para probar el programa, se ha creado un fichero xml basado en el ejemplo del enunciado. Este fichero está diseñado para contener todos los tipos de elementos admitidos en el lenguaje xml de la práctica.

Algunos campos de texto y atributos contienen tabuladores/espacios/saltos de línea/acentos para demostrar la robustez de nuestras expresiones regulares.

```
<Genero gen = 'mujer'></Genero>
<nombre compuesto="cieRto" >María José</nombre>
<apellido>García Lluch      </apellido>
<telefono     prefijo ="977" movil= "falso"></telefono>
<direccion>Calle San Francisco nº 28, 3-5, 43010 Tarragona</direccion>
<nacimiento>1970
</nacimiento>
```

Cuando ejecutamos el programa, este nos devuelve por pantalla un número correcto para cada tipo de elemento presente en el fichero de entrada. Cabe notar que el número de caracteres total es superior al que se pueda ver a simple vista en el fichero ya que este contiene en los campos de texto caracteres UTF-8 que pueden ocupar 2 o más bytes (principalmente acentos) mientras que FLEX procesa el texto en unidades de 1 byte. De todas formas, esto no afecta al resto del conteo.

```
Num aperturas = 6
Num cierres = 6
Num textos = 4
Num atributos = 4
Num caracteres total = 86
Num elementos con uno o mas atributos = 3
Num elementos vacios = 2
Num elementos simples = 4
```