Análisis Léxico.
Un Scanner para MiniLan (II)

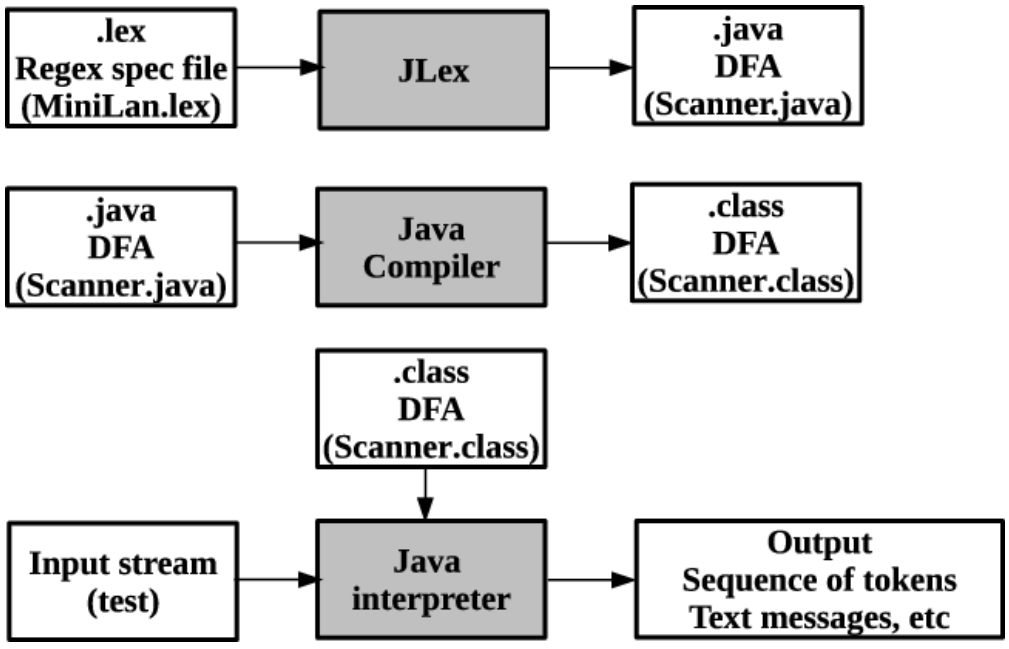
Autómatas y Matemáticas Discretas
Escuela de Ingeniería Informática
Universidad de Oviedo
Curso 2016-2017

Autómatas y Matemáticas Discretas
Escuela de Ingeniería Informática
Universidad de Oviedo

1. Resumen de sesiones previas

Las palabras clave, los símbolos de puntuación, los identificadores válidos, los números ... son definidos mediante expresiones regulares, las cuales son interpretadas por el analizador léxico o scanner, como por ejemplo, el generado a través de JLex.

El analizador léxico lee el flujo de caracteres de entrada, identifica las palabras contenidas y las compara con las expresiones regulares que denotan cada tipo de token.
El proceso de reconocer las palabras del flujo de entrada entre el conjunto de palabras válidas se realiza utilizando **máquinas de estado finito** o, lo que es lo mismo, los **autómatas finitos** vistos en teoría. Los generadores de analizadores léxicos generan un programa en un lenguaje de alto nivel -p.e., Java- que simula dicho autómata finito. Esto se lleva a cabo a partir del conjunto de expresiones regulares predefinidas.



2. MiniLan: un MINI LANguaje

MiniLan es el nombre de un lenguaje de programación muy simple definido para este curso. La especificación léxica de este lenguaje se puede encontrar en el Campus Virtual de la Asignatura. Es necesario leer y usar este documento como referencia para definir las expresiones regulares correspondientes a cada token, puesto que contiene una pequeña descripción de las cadenas de caracteres válidas en nuestro lenguaje MiniLan.

3. Ejercicios

Esta sesión completará el conjunto de macros y expresiones regulares en la especificación del lenguaje MiniLan, continuando con el trabajo de las sesiones previas. Para iniciar esta sesión, cámbiate al directorio `~/practicasAMD/LexicalAnalysis/ScannerMiniLan` y modifica tu fichero de especificación para reconocer los siguientes tokens:

1. **Nuevos operadores**: añade las reglas para `<`, `>` and `==`. Los mensajes a imprimir en cada caso deben ser:
   `SCANNER:: found Operator` seguido por `LT`, `GT` o `EQ`, respectivamente.

2. **Comentarios**

   - Escribe una macro denominada `LETTER` que case con una letra, bien sea minúscula o mayúscula.
   - Escribe una macro denominada `BLANK` que case con un espacio en blanco o con un tabulador.
   - Usando las dos macros anteriores, escribe la macro `COMMENT` que case con las líneas de comentario definidas para `MiniLan`.
   - Añade una regla que expanda la macro `COMMENT`. El mensaje de salida debe ser:
     `SCANNER:: comment line <texto del comentario>`.

3. **Números reales**

   - Añade una macro denominada `REAL`, su misión es definir el patrón para los números reales en MiniLan.
   - Añade la regla correspondiente a los números reales, de manera que expanda la macro `REAL`. El mensaje de salida debe ser (con XXXX el valor numérico detectado):
     `SCANNER:: found REAL <XXXX>`.

4. Finalmente, elimina la regla `DIGIT`, genera el Scanner y comprueba su funcionalidad con el archivo de test `testML2`, cuyo contenido se muestra a continuación.

```
print begin end
+ - *// () <==>
321 4.35 .345 43.
. // dot is not a real number
```

Dispones de una copia del mismo en: `/var/asignaturas/AMD/practicas/2016-2017/test_files/`

La salida por pantalla sería la siguiente:

```
SCANNER:: found Reserved Word PRINT
SCANNER:: found Reserved Word BEGIN
SCANNER:: found Reserved Word END
SCANNER:: found Operator ADD
SCANNER:: found Operator MINUS
SCANNER:: found Operator MULT
SCANNER:: comment line <// >
SCANNER:: found symbol RP
SCANNER:: found symbol LP
SCANNER:: found Operator LT
SCANNER:: found Operator EQ
SCANNER:: found Operator GT
SCANNER:: found NUMBER <321>
SCANNER:: found REAL <4.35>
SCANNER:: found REAL <.345>
SCANNER:: found REAL <43.>
SCANNER:: Unmatched input .
SCANNER:: comment line <// dot is not a real number>
```