Lenguajes, Gramáticas y Autómatas

Ing. en Informática – 1ª Convocatoria (11 de Junio 2007)

Tiempo de Realización: 3 horas.

Escribid las soluciones en hojas separadas para cada ejercicio

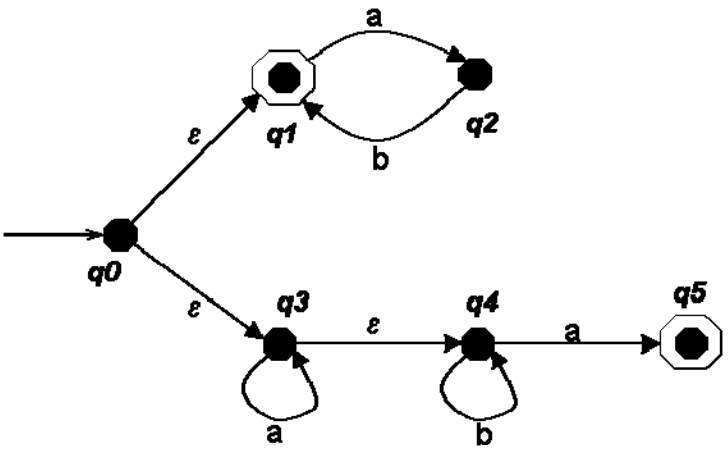
Nombre Alumno/a:.....

Parte teórica **(8 puntos)**

1. Construid una Gramática Independiente del Contexto (GIC) en Forma Normal de Greibach que genere el siguiente lenguaje, y explicad brevemente las ideas aplicadas en el proceso de construcción **(1,25 puntos)**:

$$L = \{a^i b^j c^k \mid i, j, k \in Z^+ \ \wedge (i \neq j + k)\}$$

2. Dado el Autómata Finito No Determinista (AFN) con ε–transiciones de la figura,



Importante: La solución de los distintos apartados de este ejercicio deberá contener el proceso de transformación completo aplicado a cada autómata en particular.

Se pide:

2.1. Construid, aplicando la transformación formal vista en clase, un AFN sin ε–transiciones equivalente **(1 punto)**.

2.2. Partiendo del AFN sin ε–transiciones del apartado anterior, construid, aplicando la transformación formal vista en clase, un Autómata Finito Determinista equivalente **(1 punto)**.

3. Sea el lenguaje $L \subseteq \{0, 1, \#\}^*$, construid un Autómata Finito Determinista (AFD) que reconozca el lenguaje, y explicad brevemente su funcionamiento **(1,25 punto)**:

$$L = \{ w\#x \mid w, x \in \{0, 1\}^* \wedge \text{ x termina con la subcadena 01} \wedge |x|_1 \text{ es impar} \wedge |wx|_0 \text{ es múltiplo de 2 } \}$$

4. Demostrad formalmente, **aplicando técnicas vistas en clase**, para cada uno de los siguientes lenguajes, si se trata de un lenguaje regular, si es un lenguaje independiente del contexto, o si es un lenguaje que no es ni regular ni independiente del contexto **(2 puntos)**:

   4.1. $L = \{ 0^i 1^i 0^j \mid i, j \geq 0 \wedge i + j \text{ es par}\}$
   4.2. $L = \{ 0^i 1^j 0^j \mid i, j \geq 1 \wedge j > i\}$

5. Construid un Autómata de Pila No Determinista (ADPND) para el siguiente lenguaje. Además de la tabla de transiciones del autómata, debéis dar una explicación razonada sobre su funcionamiento **(1,5 puntos)**.

   $$L = \{a^n w \mid n = |w|_a \wedge w \in (a+ba)^*\}$$

Parte práctica **(2 puntos)**

Se tiene una red de sensores meteorológicos que proporcionan valores de temperatura y velocidad del viento. Un receptor recoge las mediciones realizadas por cada sensor durante varios días y las envía para su procesado a la central meteorológica, añadiendo información sobre las fechas de las mediciones. El formato con el que esta información se envía al computador de la central es el siguiente **(2 puntos)**:

<**SecuenciaDeMediciones**> *fecha*
<Temperatura> *número_real [unidades_temperatura] [margen_de_error]*
<VelViento> *número_real [unidades_velocidad_viento] [margen_de_error]*
<Temperatura> *número_real [unidades_temperatura] [margen_de_error]*
<VelViento> *número_real [unidades_velocidad_viento] [margen_de_error]*
...
<**SecuenciaDeMediciones**> *fecha*
<Temperatura> *número_real [unidades_temperatura] [margen_de_error]*
<VelViento> *número_real [unidades_velocidad_viento] [margen_de_error]*
...

Notas:

- *fecha* viene siempre en el siguiente formato **dd-mm-aaaa**. No hace falta comprobar si es una fecha posible, sólo hay que verificar que tiene ese formato (por ejemplo, se aceptará como fecha válida el 99-99-9999 pero no se aceptará el 1-1-2007).
- Hay un número desconocido de secuencias de mediciones, y hay un número desconocido de mediciones para cada secuencia, pero al menos habrá una medición (de temperatura y velocidad del viento) por secuencia.
- Cada sensor envía siempre una medición, formada por un valor de temperatura y otro de velocidad del viento, aunque las unidades de temperatura y velocidad del viento varían entre los distintos sensores.
- *unidades_temperatura* es un parámetro opcional que puede tomar dos valores: **C** (para grados centígrados) y **F** (para grados farenheit).
- *unidades_velocidad_viento* es un parámetro opcional que puede tomar dos valores: **m/s** (para metros por segundo) y **f/s** (para pies por segundo).

- *margen_de_error* es un parámetro opcional que es un número real que indica el margen de confianza del sensor para cada medición. Puede ir precedido de un signo +, un signo - o bien los signos +/- y si aparece irá siempre entre paréntesis.
- Supondremos, por simplificar, que los números reales que aparecen siempre tienen punto y parte decimal y que no pueden venir dados en notación exponencial. Lo que si pueden tener es un signo – o + delante (evidentemente, los que aparecen entre paréntesis ya tienen sus propios signos y no podrán tener más que un +, un – o un +/-, siendo por ejemplo inválido encontrar +/-+0.5).

A continuación se muestra un ejemplo de una posible salida del receptor:

```
<SecuenciaDeMediciones> 12-12-2003
<Temperatura> 25.0 C
<VelViento> 9.35 (-0.2)
<Temperatura> +89.1 F
<VelViento> 10.50 f/s (+/-0.5)
<Temperatura> -15.2 C (+1.33)
<VelViento> 3.0 m/s
<SecuenciaDeMediciones> 14-12-2003
<Temperatura> -26.0 (+/-0.2)
<VelViento> 9.65 f/s
```

Se pide:

1. Escribid un analizador léxico en FLEX para la sintaxis dada: este analizador deberá reconocer los siguientes tokens en un fichero de texto y pasarlos a un programa Bison:
   - Un token para <SecuenciaDeMediciones>, otro para <Temperatura> y otro para <VelViento>.
   - Un token para fechas con el formato explicado.
   - Un token para números reales con las simplificaciones expuestas en las notas.
   - Un token para números reales precedidos de un +, o un – o un +/- y encerrados entre paréntesis.
   - Un token para cada unidad de temperatura (2) y un token para cada unidad de velocidad del viento (otros 2).

2. Escribid un analizador sintáctico en BISON que acepte un fichero de texto con una lista de secuencias de mediciones, que escriba por pantalla “parse error” si no se corresponde con la sintaxis descrita y que no escriba nada si este fichero es conforme a esta sintaxis. Este analizador usará el programa Flex del ejercicio anterior.

   Nota: Lo que se pide hacer en FLEX hay que hacerlo en FLEX, y lo solicitado en BISON en BISON. No se valorarán otras soluciones distintas.