Grupo Jueves 12:00-14:00 semanas B

- Práctica 2-

Autor: Sergio Ros Alcázar
NIP: 874792
Autor: Irene Pascual Albericio
NIP: 871627

EJERCICIO 1:

1. Resumen

   Para indicar el principio de la sección declaración, detectará “%{“ hasta que detecte “%%”, que es el comienzo de la siguiente sección, la sección de reglas.
   La última sección comenzará cuando termine la anterior, es decir, cuando vuelva a detectar “%%”.
   Para los comentarios, estos empezarán en cualquier sección, siempre que empiecen por “//”.
   Las reglas son aquellas que empiezan por “{“ y terminan en “}”.
   Las instrucciones se marcan cada vez que aparece el “;” y no está dentro de printf o comentarios.
   Problemas resueltos:

   - No contabilizar el “;” en printf(“”) y comentarios:
     - Printf: si hay unas comillas de apertura, luego cualquier carácter una o más veces, el “;”, cualquier carácter una o más veces y las comillas de apertura→ no se detecta como instrucción.
     - Comentarios→ hacemos que el comentario lea cualquier cosa que haya después del “//” para no tener que preocuparnos.
   - No contabilizar la regla dentro de otra regla.
     - Creamos una nueva sección: LINEAREGLA.
     - Cada vez que se detecte un “{“ en la sección de código se irá a esta nueva sección, para que así, si vuelve a detectar “{“, seguido de cualquier posible carácter, seguido de “}”, no lo contabilice como una nueva regla.
     - También indicamos si dentro de una regla hay “{“ … “}”, tenga en cuenta si hay algún “;” para que lo contabilice como instrucción.

2. Pruebas

```
hendrix02:~/2ºCARRERA/tcomp/ flex ej1_2.l
hendrix02:~/2ºCARRERA/tcomp/ gcc lex.yy.c -lfl -o ejercicio1
hendrix02:~/2ºCARRERA/tcomp/ ./ejercicio1 < texte2_1.txt
int main() {    yylex ();a==0) a++aqui
 L1: 6

 L2: 5

 L3: 3

 C: 4

 R: 3

 I: 5
hendrix02:~/2ºCARRERA/tcomp/
```

EJERCICIO 2:

1. Primero hemos averiguado los lenguajes de ambos autómatas, y luego hemos realizado la intersección de los dos.

   Para el lenguaje del primer autómata M1:

   - He hecho uso de Jflap para dibujar 3 autómatas, los cuales solo diferían en su estado inicial. Por tanto he creado un nuevo autómata y he llevado con épsilon transiciones al q0 del primero, al q1 del segundo y al q2 del tercero (esto lo he hecho para que el lenguaje acepte las cadenas que acaben en q0 sea donde empiece).
   - Como M1 era la intersección de esos 3 autómatas, he negado el estado final de cada uno, por tanto, ahora los estados finales son q1 y q2 de todos ellos. Una vez ya estaban combinados sus negaciones, he negado esa unión, quedando por tanto un único estado final.

   Para el lenguaje del segundo autómata M2:

   - He llevado a cabo los mismos pasos que para el autómata anterior. O sea, he creado 4 autómatas, unidos por un estado inicial que enlazaba mediante épsilon transiciones con q0 del primero, q1 del segundo, q2 del tercero y q3 del cuarto.
   - He vuelto a negar sus estados finales de cada uno, siendo estos: q1, q2, q3 y q4.
   - Por último los he combinado y lo he negado, quedando por tanto un único estado final.

   Para la intersección de M1 y M2:

   - De nuevo, he hecho como en los anteriores: he combinado los dos autómatas anteriores y los he unido mediante las épsilon transiciones a los estados iniciales que tenían cada uno.
   - Luego, cada autómata lo he negado, quedando por tanto como estados finales todos excepto un solo estado.
   - Después he unido todos esos estados finales en uno solo mediante muchas épsilon transiciones.
   - Para finalizar, el nuevo autómata creado lo paso a un autómata finito determinista, y lo he negado (que en consecuencia, solo quedaba un solo estado final).

2. Pruebas:

```
hendrix02:~/2ºCARRERA/tcomp/ flex ej2_2.l
hendrix02:~/2ºCARRERA/tcomp/ gcc lex.yy.c -lfl -o ejercicio2
hendrix02:~/2ºCARRERA/tcomp/ ./ejercicio2
abbabb
**abbabb


abbabbh
abbabbh


abbabb ab
abbabb ab


ab babb
ab babb


abbabb
abbabb


bbab
--bbab


abbabbbbab
**--abbabbbbab


hendrix02:~/2ºCARRERA/tcomp/
```