Codificación y prueba de un autómata

Lenguajes y Autómatas I

El alumno deberá codificar el autómata que calculó en la práctica anterior (el sistema **R**) y probar su funcionamiento con algunas cadenas.

Operación del programa

Su programa deberá leer una cadena desde el teclado. Como respuesta, su programa deberá decir si la cadena está o no en el lenguaje **R**.

Codificación del autómata

Se sugiere utilizar la siguiente estrategia para codificar su autómata. Cualquier estrategia alternativa a esta deberá ser señalada y justificada.

- La codificación de los estados deberá realizarse mediante el uso de enteros no negativos (`unsigned`: 0,1, 2, ...).
- Cada uno de los caracteres de su cadena será un evento para el autómata. Para almacenar su cadena podrá usar un `char *` (una cadena tipo C) o un std::string[1] (una cadena estilo C++).
- El conjunto de estados finales podrá almacenarse en un objeto de tipo *set* (std::set)[2].
- La función de transición podrá codificarse mediante la implementación de un doble *mapeo* (std::map)[3], según se explica enseguida.

Para codificar la función de transición δ: $Q \times \Sigma \rightarrow Q$ partimos de su visualización como una tabla (*tableau*), en la cual reservamos una fila para cada estado del autómata y una columna para cada evento. Vamos a usar los mapeos para implementar una *tabla virtual*, en la cual se lee la intersección entre una fila (un estado) y una columna (un evento) para conocer cual es el estado sucesor.

```
typedef std::map<char,unsigned> event;
typedef std::map<unsigned,event> transition;

transition delta;
```

De esta manera, la transición $\delta(q, \sigma)$ se puede codificar como `delta[q][sigma]`.

Tome en cuenta que en un mapeo, el uso del operador `[]` relaciona al argumento con el valor mapeado, pero si este no se especifica, entonces se genera automáticamente con el constructor por defecto. En el caso de los tipos de datos numéricos, la construcción por defecto genera un 0. Es decir, si su programa recibe una cadena con símbolos diferentes a los contemplados en su función de transición, el mapeo generará automáticamente una relación con 0 y devolverá ese dato como resultado. Por ello *se sugiere* hacer una **validación de eventos** antes de resolver una transición: siempre que sea requerida una transición no definida su autómata debe parar el procesamiento. Esta validación puede realizarse con el uso de iteradores y el método find[4].

La clase automata

Deberá diseñar la siguiente clase para implementar su autómata:

automata
-init: el estado inicial -delta: la función de transición -final: conjunto de estados finales ...
+automata() +test(string): bool

- En el constructor deberá inicializar su autómata: el estado inicial, la función de transición y el conjunto de estados finales.
- La función *test* deberá recibir una cadena de eventos y devolver como resultado si la cadena es aceptada o rechazada por el autómata.

```
if (a.test(w)) std::cout << "accepted\n";
else std::cout << "rejected\n";
```

Observe que no es necesario codificar el conjunto de estados y el alfabeto, ya que están definidos implícitamente en la definición de la función de transición, pero puede hacerlo si lo considera conveniente. Agregue a su clase las funciones adicionales que necesite para completar su operación.