Implementación de un procesador de lenguajes para Autómatas de Moore

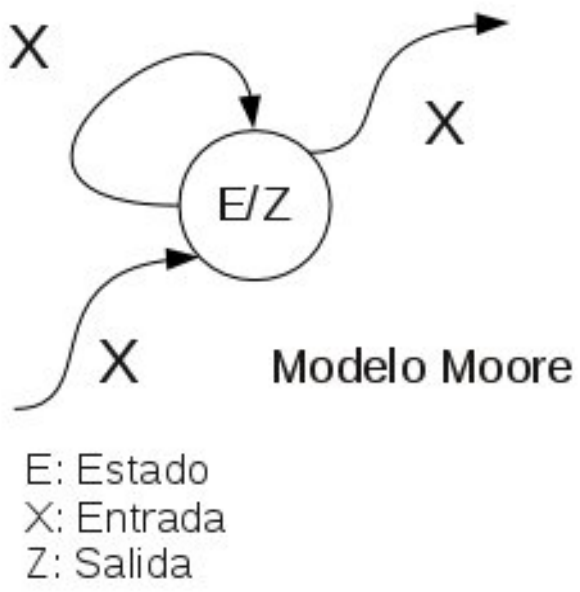
Julián García Sánchez Iván Illán Barraya
Alejandro Medina Jiménez Javier Monescillo Buitrón

5 de enero de 2019

Índice

1. ¿Qué es un autómata de Moore?

En Teoría de la computación una **Máquina de Moore** es un autómata de estados finitos para el cual la salida en un momento dado sólo depende de su estado en ese momento, mientras que la transición al siguiente estado dependerá del estado en el que se encuentre y de la entrada introducida.
El diagrama de estados para una máquina de Moore incluirá una señal de salida para cada estado. [1]



Ejemplo de Máquina de Moore simple

1.1. Definición formal

Una máquina de Moore se define como una 6-tupla:

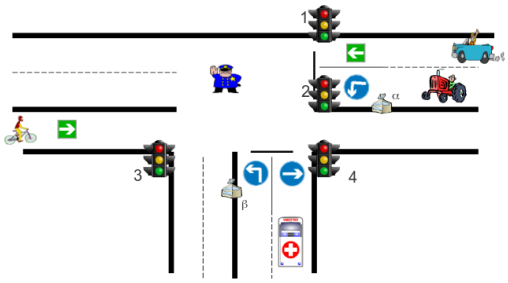
$$Mmor = (S, S_0, \Sigma, \Lambda, T, G)$$

donde definimos los siguientes elementos:

- S: es un conjunto finito de estados.
- S_0: es el estado inicial, y además es un elemento de S.
- Σ: un conjunto finito llamado alfabeto de entrada.
- Λ: un conjunto finito llamado el alfabeto de salida.
- T : una función de transición $T : S \times \Sigma \rightarrow S$ mapeando un estado y una entrada al siguiente estado.
- G función de salida $G : S \rightarrow \Lambda$ mapeando cada estado al alfabeto de salida.

1.2. Ejemplo propuesto

Se define una *Máquina de Moore* para regular el tráfico en el siguiente cruce con los cuatro semáforos:



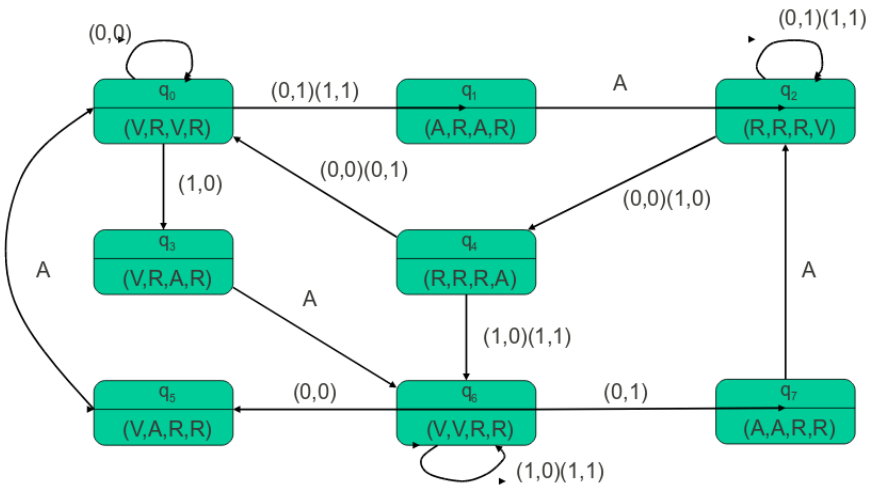
El alfabeto de entrada Σ: (0,0),(0,1),(1,0),(1,1)

- (a,b), donde a es el estado para el sensor α y b para el sensor β
- 0 indica que no hay coches en la cola
- 1 indica que hay coches en la cola

El alfabeto de salida Λ: (a_1, a_2, a_3, a_4 donde $a_i \epsilon$ (A,V,R))

- a_1 es el estado del semáforo 1
- a_2 es el estado del semáforo 2
- a_3 es el estado del semáforo 2
- a_4 es el estado del semáforo 2

El diagrama de estados sería el siguiente



2. Presentación del problema

La práctica consiste en el diseño de un procesador de lenguajes cuya entrada estará formada por una o varias máquinas de Moore y su salida será código en un lenguaje de alto nivel que lo represente, en nuestro caso Java.

Los conocimientos de la materia *Procesadores de Lenguajes* serán útiles en las distintas etapas del proceso, en primer lugar, se necesitará definir el léxico del lenguaje de entrada, la construcción del procesador de lenguajes asociado empleando los diagramas **tipo T**, el análisis léxico, el análisis sintáctico y por último el análisis semántico.

También se desarrollará una tabla de tokens, lexemas y patrones del lenguaje y el EBNF perteneciente al mismo.

2.1. Lenguaje definido

El lenguaje **Moor** que se ha definido consta de dos secciones principales:

- Sección de declaración de código
- Sección de declaración de autómatas

La *zona de declaración de código* es la parte donde se asignará un comportamiento a un código ya predefinido o que se vaya a ejecutar. Por la forma en la que se ha definido el lenguaje, esta zona será como una zona de *imports* típicamente conocida en lenguajes de alto nivel.

La estructura de esta zona es: c*N* # *código* # donde c es el comportamiento, *N* es el número de comportamiento y los **tokens** # son los delimitadores del código.
La característica principal de esta zona es que solo se podrán realizar este tipo de declaraciones, ya que en la otra zona solo podremos declarar autómatas.

Ejemplo : c1 # print('hola') #

Respecto a la *zona de declaración de autómatas* se tendrá una serie de palabras que se utilizarán para la definición del autómata. No obstante en esta zona solo se podrán declarar autómatas y las funciones asociadas a los mismos, también, como en cualquier lenguaje de alto nivel, se podrán incluir los clásicos comentarios '//' o '/* Texto comentario '*/'.

Será posible la declaración de múltiples autómatas en el fichero de prueba, además, se indicará en una tabla todos los campos que son imprescindibles para la realización del mismo. Cualquier otro carácter introducido en el fichero de prueba será considerado como error.

2.2. Tabla Token-Acción

Token	Acción
moore	Función para declarar un autómata seguido de un nombre que se escribirá entre llaves {} moore Ejemplo {}
estados	Indica la cantidad de estados totales que tendrá el autómata puede ser un único estado o varios separados por ',' estados q0,q1;
estado_in	Se selecciona un único estado inicial que tendrá que estar en los estados anteriores estado_in q0;
alf_in	La entrada o eventos del autómata que hará posible las transiciones entre estados, único o entre comas ',' alf_in a,b,c;
alf_out	La salida o comportamientos del autómata y tendrán que ser coincidentes con la zona de declaración de comportamientos, escrito un identificador único o entre comas ','
transicion	Será un tipo de función que se escribirá entre llaves {}, y permitirá que se transite mediante una entrada de un estado a otro: (<estado_origen>, entrada, <estado_destino>); se podrán indicar varias mediante el uso de comas ','
comportamiento	Será un tipo de función que se escribirá entre llaves {}, y permite asignar a un estado un comportamiento: (<estados>,<comportamiento>); se podrán indicar varios mediante el uso de comas ','

Notese que para cerrar una sentencia es necesario de indicar al final de dicha sentencia el carácter ';', esto no es necesario para las funciones *moore*, *transicion* y *comportamiento*.

2.3. Ejemplo autómata completo

```
c1      # System.out.println("Activado sensor 1"); #

c2      # System.out.println("Activado sensor 2"); #

c3      # System.out.println("Activado sensor 3"); #

/*
Comentario

*/

moore Automata_de_Ejemplo{

        estados q0,q1,q2;
        estado_in q0 ;
        alf_in e1,e2,e3;
        alf_out c1,c2,c3;

        transicion {
                (q0,e1,q1),
                (q1,e2,q1),
                (q1,e3,q2);
        }

        comportamiento {
                (q0,c1),
                (q1,c2),
                (q2,c3);

        }
}
```

3. EBNF

PROGRAMA	::=	DEC_COMP AUTOMATA {AUTOMATA}
DEC_COMP	::=	CMP CODIGO { CMP CODIGO }
CODIGO	::=	’#’ ASCII ’#’
AUTOMATA	::=	**moore** ID CUERPO_AUTOMATA
CUERPO_AUTOMATA	::=	’{’ ESTADOS ESTADO_INI ALF_IN ALF_OUT TRANSICION COMPORTAMIENTOS ’}’
ESTADOS	::=	**estados** { ID ’,’ } ID ’;’
ESTADO_INI	::=	**estado_in** ID ’;’
ALF_IN	::=	**alf_in** { EVENTOS ’,’ } EVENTOS ’;’
ALF_OUT	::=	**alf_out** { CMP ’,’ } CMP ’;’
TRANSICION	::=	**transicion** ’{’ TRANSICION_DEF { ’,’ TRANSICION_DEF } ’;’ ’}’
TRANSICION_DEF	::=	’(’ ID ’,’ ID ’,’ ID ’)’
COMPORTAMIENTOS	::=	**comportamientos** ’{’ COMP_DEF { ’,’ COMP_DEF } ’;’ ’}’
COMP_DEF	::=	’(’ ID ’,’ ID ’,’ ID ’)’
CMP	::=	’c’NUMEROS
NUMEROS	::=	0 | 1 | .. | 9
COMENTARIOS	::=	’/*’ ASCII ’ * /’

4. Tablas de Tokens

Token	Lexema	Patrón
moore	moore	m·o·o·r·e
estados	estados	e·s·t·a·d·o·s
estado_in	estado_in	e·s·t·a·d·o·_·i·n
alf_in	alf_in	a·l·f·_i·n
alf_out	alf_out	a·l·f·_o·u·t
transicion	transicion	t·r·a·n·s·i·c·i·o·n
comportamiento	comportamiento	c·o·m·p·o·r·t·a·m·i·e·n·t·o
Paréntesis abierto	(	(
Paréntesis cerrado	)	)
Llave abierta	{	{
Llave cerrada	}	}
Punto y coma	;	;
Coma	,	,
Asterisco barra	*/	*·/
Barra asterisco	/*	/·*
ID	hola	[A-Za-z][A-Zaz0-9_]*
CMP	c1	c[1-9][0-9]*
CODIGO	#codigo aquí #	#· codigo aquí ·#

5. ¿Cómo se va a construir el procesador del lenguaje diseñado?

En el siguiente diagrama tipo T se explica el funcionamiento del procesador que se pretende implementar, tomamos **Moor** como lenguaje fuente, y Java como lenguaje objeto, y además el lenguaje que implementa es Java, es decir, nuestro compilador compila a Java, y esta escrito a Java.
Para utilizar ese compilador, utilizamos un compilador auxiliar, que está escrito en código máquina para dar lugar a bytecode, el típico archivo **.class** que generamos al compilar el archivo **.java**.
Por último tenemos que compilar el bytecode, con un compilador que acepta bytecode escrito en código máquina.

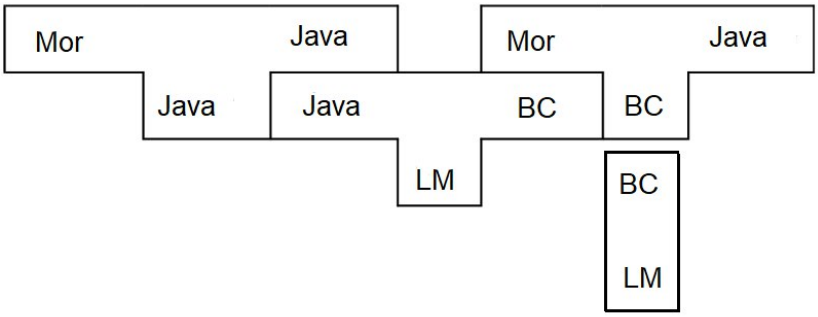
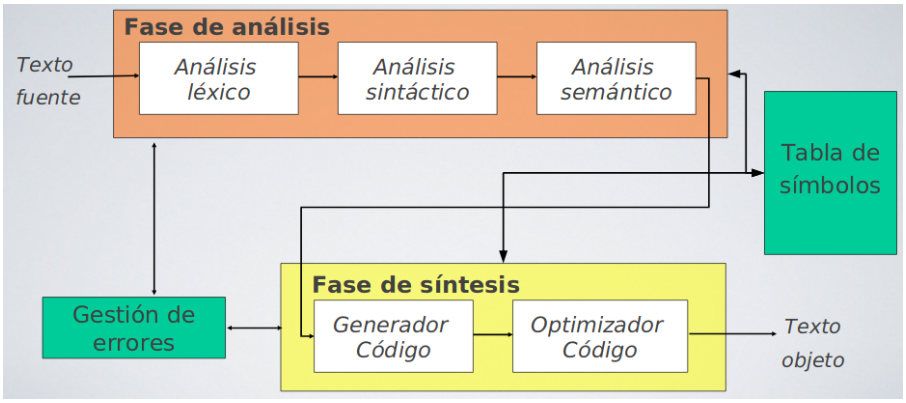


Diagrama tipo T asociado

6. Estructura del procesador de lenguajes

Siguiendo la estructura que se ha utilizado en clase, tenemos la siguiente organización:



El procesador de lenguajes que se va a construir seguirá una estructura similar salvo por el bloque de la fase de síntesis, ya que no se ha visto en la asignatura.

Se procederá a la construcción de la fase de análisis que consta de las siguientes etapas:

- Análisis léxico
- Análisis sintáctico
- Análisis semántico

7. Analizador léxico

7.1. Jflex

7.2. ANTLR

8. Analizador sintáctico

8.1. CUP

8.2. ANTLR

9. Analizador semántico

Referencias

[1] Autómatas de Moore https://es.wikipedia.org/wiki/M%C3%A1quina_de_Moore