DOCUMENTACION PROYECTO PROCESADORES DE LENGUAJE

Bekeri, Beka

Cordoba, Javier

Corroto, Juan Jose

Vallejo, Fernando

Hito 1 : 07/10/18

ÍNDICE

1º Presentacion del problema

En este proyecto nos enfrentamos a un problema en el cual se nos pide crear un procesador de lenguaje para cierto Lenguaje de definición de Maquinas Moore. Para ello vamos a necesitar la implementación de analizadores léxicos, sintácticos y semánticos

Es decir, para este proyecto primero necesitamos crear un lenguaje para la definición de máquinas de moore, y respecto a ese lenguaje va a trabajar nuestro “traductor”, el cual será capaz de originar la misma máquina de moore pero en un lenguaje distinto, lenguaje objeto (JAVA).

Debido a que este proyecto abarca todo el proceso de la creación de un procesador de lenguaje podemos centrarnos en aprender, de forma práctica y divertida, el uso de los procesadores de lenguaje, ya que el dominio del problema (Máquinas de Moore) nos es conocido a todos.

2º Lenguaje Mooma

Nuestro lenguaje para el diseño de máquinas de moore se llama *Mooma* y viene definido de la siguiente forma:

```
PROGRAM ::= DEFINE,{BLOCK};
BLOCK ::= DEFINE|AUTOMATON|OUTPUT;
DEFINE ::= "define" IDENT "{" ENTRADA ";" SALIDA ";" "}";
AUTOMATON ::= "automaton" IDENT "(" IDENT ")" "{" STATES ";" INITIAL ";" TRANSITIONS "}";
ENTRADA ::= "in" ":=" {INPUT};
SALIDA ::= "out" ":=" ALFABETO;
STATES ::= "states" ":=" [{IDENT "|" INPUT,}|epsilon] IDENT "|" INPUT;
INITIAL ::= "initial" ":=" IDENT;
TRANSITIONS ::= "transitions" "{" {IDENT "->" [{IDENT "|" INPUT,}|e] IDENT "|" INPUT ";" } "}";
IDENT ::= LETRA {LETRA|DIGITO};
LETRA ::= "A" | "B" | "C" | "D" | "E" | "F" | "G"
       | "H" | "I" | "J" | "K" | "L" | "M" | "N"
       | "O" | "P" | "Q" | "R" | "S" | "T" | "U"
       | "V" | "W" | "X" | "Y" | "Z" | "a" | "b"
       | "c" | "d" | "e" | "f" | "g" | "h" | "i"
       | "j" | "k" | "l" | "m" | "n" | "o" | "p"
       | "q" | "r" | "s" | "t" | "u" | "v" | "w"
       | "x" | "y" | "z" ;
DIGITO ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ;
INPUT ::= {EVENT,|epsilon};
EVENTO ::= (1|2|3|4|5|6|7|8|9){DIGITO};
OUTPUT ::= "output" ":=" CODIGO ";";
CODIGO ::= "{:" {.} ":}";
```

Un dato importante para tener en cuenta es que nuestro lenguaje trabaja con eventos, es decir la creación del diccionario de correspondencia entre eventos y las entradas del problema ha de ser diseñado por el usuario para la comprensión del resultado.

Los programas *Mooma* se encuentran divididos en tres partes:

1. Declaración de entradas y salidas (eventos y código). Aquí se incluirá una construcción que empieza con la palabra *"define"*, seguido de un identificador, y se definirán un alfabeto de entrada (Recordamos que este alfabeto siempre consistirá en números que denotan eventos), y un alfabeto de salida. Este alfabeto de salida será una serie de identificadores que estarán definidos en otro lugar, y denotan fragmentos de código en java que serán ejecutados como salida de un estado.
2. Declaración de un autómata concreto. Ésta será otra construcción que empezará con la palabra *"automaton"* y, entre paréntesis, el identificador de una construcción *"define"* (Lo cual denotará los alfabetos de entrada y salida del autómata). Dentro de la construcción se incluirán el conjunto de estados del autómata junto con el identificador de su salida asociada, el estado inicial y una lista con las transiciones entre estados del autómata.

3. Código asociado a las salidas de los estados, que tendrá que tener un identificador asociado para poder asociarlo a un alfabeto de salida en una construcción *"define"* y a un estado en una construcción *"automaton"*.

Aquí dejamos la tabla de tokens y la tabla de palabras reservadas correspondiente a nuestro lenguaje *Mooma*:

Token	Patrón	Lexema de ejemplo
define	define	define
automaton	automaton	automaton
in	in	in
out	out	out
states	states	states
initial	initial	initial
transitions	transitions	transitions
output	output	output
Llave_derecha	}	}
Llave_izquierda	{	{
Punto_y_coma	;	;
Coma	,	,
Asignación	:=	:=
Par_derecho	)	)
Par_izquierdo	(	(
Flecha	->	->
Código_inicio	{:	{:
Código_final	:}	:}
Separator	|	|
Identificador	[A-Za-z][A-Za-z0-9]*	Foo5
Event	[1-9][0-9]*	37
Código	.*	System.out.println("HelloWorld!");

Palabra reservada	Token
define	define
automaton	automaton
in	in
out	out
states	states
initial	initial
transitions	transitions
output	output

3º Ejemplo Mooma

A continuación veamos un ejemplo de definición de una maquina moore en Mooma:

```
define alfabetos1{
    in := 1, 2;
    out := code1, code2;
}

automaton cosa1 (alfabetos1){
    states := q0|code1, q1|code2;
    initial := q0;
    transitions{
        q0 -> q1|1, q0|2;
        q1 -> q1|1, q0|2;
    }
}

code1 := {:
        System.out.println("Hello");
        :};
code2 := {:
        System.out.println("Goodbye");
        :}
```

Como se puede apreciar solo tenemos un autómata con dos eventos, en este caso (1 y 2), dos estados (q0 y q1) y sus dos salidas junto con el código correspondiente el cual va declarado al final del programa. A parte se puede observar como se declaran las transiciones en nuestro lenguaje, es decir, para cada estado con transiciones se debe declarar de forma que la estructura sea la siguiente

EstadoActual → (EstadoX al que llega)|(si ocurre el evento E1), (EstadoY al que llega)|(si ocurre el evento E2), …;

4º Diseño

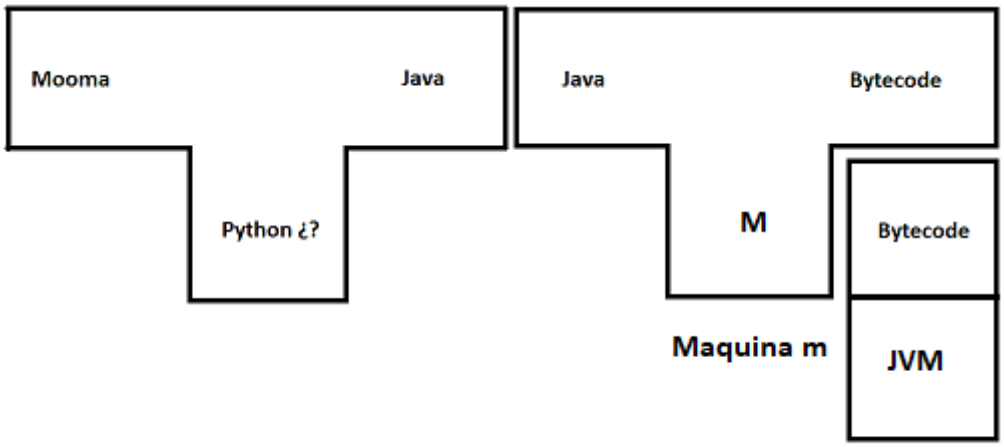
Para nuestro proyecto necesitábamos saber 3 cosas:

- Lenguaje fuente (Diseñado por nosotros). Mooma.
- Lenguaje de implementación (con el que vamos a trabajar nosotros)
- Lenguaje objeto (al cual vamos a "traducir" nuestro lenguaje)

En cuanto al lenguaje de implementación, de momento creemos que vamos a utilizar Python, debido a que todos lo manejamos y que nos parece un lenguaje útil y versátil a la hora de hacer grandes proyectos, aunque de momento necesitamos concretar los generadores de código que existen para Python.

Y respecto al lenguaje de salida vamos a utilizar Java de forma definitiva.

De modo que nuestro T-Diagrama quedaría de la siguiente forma:



ANEXO I. REUNIONES

Reunión 1

Fecha: 25/09/2018

Hora de inicio: 16:15

Hora de fin: 17: 30

Objetivos de la reunión: Al ser la primera reunión, intentaremos determinar las características de nuestro proyecto (lenguajes a utilizar, máquinas de moore, etc). Por otra parte buscamos la creación de nuestro lenguaje fuente.

Decisiones tomadas: Se terminó el diseño del lenguaje fuente del proyecto. Hemos empezado a buscar lenguajes objeto para el proyecto y hemos barajado algunas posibilidades para el lenguaje de implementación.

Miembros reunidos: Javier Córdoba, Juan Jose Corroto, Fernando Vallejo, Beka Bekeri

Reunion 2

Fecha: 02/10/2018

Hora de inicio: 16:15

Hora de fin: 18:25

Objetivos de la reunión: Una vez que sabemos cuál va a ser nuestro lenguaje fuente y que tenemos su diseño completo, podemos pasar a crear el EBNF de nuestro lenguaje. Por otro lado, Juanjo ha traído un ejemplo de máquina de moore en java, para ir viendo la portabilidad de Mooma a Java.

Decisiones tomadas: Java será definitivamente nuestro lenguaje objeto y Python tiene muchas posibilidades de ser nuestro lenguaje de implementación. El EBNF de Mooma está terminado.

Miembros reunidos: Javier Córdoba, Juan Jose Corroto, Fernando Vallejo, Beka Bekeri