DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico II

Teoría de las Comunicaciones
Primer Cuatrimestre de 2016

Integrante	LU	Correo electrónico
Federico De Rocco	408/13	fede.183@hotmail.com
José Massigoge	954/12	jmmassigoge@gmail.com
completa tus datos leandro		

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

En el presente Trabajo Práctico diseñamos una gramática para el lenguaje *Dibu* e implementamos sus analizador léxico y su analizador sintáctico y semántico (parser). Utilizando estas herramientas creamos un programa que permite traducir de *Dibu* a *SVG* y compilar dicha traducción, en caso de que sea válida. En los siguientes items se describirán la gramática para el lenguaje, el lexer y el parser.

2. Gramática

En la siguiente sección vamos a mostrar y describir la gramática construida para el lenguaje *Dibu*, llamo a esta gramática G:

P $\rightarrow$ S newline P | λ
S $\rightarrow$ id PARAMS
PARAMS $\rightarrow$ id = V PARAMS P | id = V
V $\rightarrow$ num | string | (num, num) | [ARRAY]
ARRAY $\rightarrow$ [num, num], ARRAY | [num, num]

G = {{P, S, PARAMS, V, ARRAY}, {num, string, (, , ,), [,], =, id}, Descripto por la gramática, P}

Como se describe en el enunciado, el lenguaje *Dibu* es una serie de instrucciones de la forma:

IDENTIFICADOR PARAM1=V1, PARAM2=V2, ..., PARAMN=VN

Nuestra gramática consiste en cinco producciones. Comenzando con ARRAY la cual describe la secuencia de valores numericos separados por una coma, la cual se utilizara en V para describir arreglos. La producción V que contiene las combinaciones de terminales que se pueden esperar como valores de los parámetros (num, string, point, array). PARAMS posee la serie de compuesta por: parámetro = valor. S describe una instrucción, con su nombre y parámetros. Y finalmente P, describe la serie compuesta de las instrucciones descriptas en el no-terminal S. Con estas producciones se puede ver con estos que nuestra gramática describe precisamente la serie de instrucciones de *Dibu*. Se debe aclarar que esta gramática no describe todas las restricciones de *Dibu*, como por ejemplo que solo aparezca la instrucción size una vez. Estas cuestriones se tratarán en el lexer y el parser.

3. Lexer

En esta sección vamos a explicar el analizador léxico implementado para la gramática descripta anteriormente. Al aplicar un lexer sobre una cadena de caracteres este devolverá verdadero su la misma no contiene errores de carácter léxico como por ejemplo caracteres que el lenguaje no reconoce. Para llevar a cabo este analizador contamos con una serie de tokens de los cuales se da una descripción de sus posibles contenidos. Estos son:

NUMBER: Son todos los números compuestos por los caracteres numéricos del 0 al 9 y el punto. El lexer identifica el tipo del valor numérico siendo estos enteros y de punto flotante de acuerdo a si el numero posee o no el punto. Esto se hace para el posterior control de errores.

STRING: Cadenas de caracteres entre comillas compuestas por las letras de la a a la z, incluidas mayúsculas, los números del 0 al 9 y los símbolos +,- y *.

ID: Igual que STRING pero sin las comillas.

LPAREN: El carácter [.

RPAREN: El carácter].

LBRACKET: El carácter (.

RBRACKET: El carácter).

COMMA: El carácter ,.

EQUALS: El carácter =.

NEWLINE: El salto de linea (). Este tiene la particularidad de que aumenta en una linea el largo del texto, esto se utiliza para detectar la posición de los errores.

IGNORE: El carácter vacío.

ERROR: Este aparece cuando se encuentra un token desconocido. En consecuencia se guarda la linea, valor, posición y tipo de valor para el error. Con este último cumplimos que si la cadena es inválida se detectara el error y se informará de él.

Como se puede ver con esta serie de tokens y sus respectivas implementaciones podemos describir todos los posibles no-terminales necesarios para nuestra gramática. Quedan todavía restricciones como por ejemplo que ID solamente puede tener como valores a los identificadores de las instrucciones.

4. Parser