GNU Pascal 1990 Compiler

Universidad de Alcalá
Ingeniería Técnica en Informática de Gestión
Escuela Politécnica Superior
Dpto. de Automática
{diego.lucena.pumar}@gmail.com

29 de septiembre de 2014

Índice

i. **Desarrollar el estándar ISO Pascal 7185:1990 además de la construcción de un prototipo** para su parte léxica (basada en Flex) y su parte sintáctica (basada en Bison).
ii. **Síntesis y Lenguaje Matemático propio de la Teoría de Lenguajes de Programación** así como su evolución e influencias históricas.

i. Crear un **lenguaje claro y natural orientado a la enseñanza** de los fundamentos de la programación de computadores. Por ello se estructuran los módulos como funciones y procedimientos.
ii. Definir un lenguaje que **permita realizar programas lo más eficientes posibles**. El tipado de datos es explícito.

Un Lenguaje Formal se compone de un conjunto de signos finitos y unas leyes para operar con ellos.

i. Al conjunto de símbolos de un lenguaje se les denomina *Alfabeto*, denotado como Σ.

ii. Al conjunto de leyes que describen al lenguaje se les denomina *Sintaxis*.

Se puenden definir a través de:

i. Mediante cadenas producidas por una gramática de Chomsky.

ii. Por medio de una Expresión Regular.

iii. Por cadenas aceptadas por un Autómata.

Dadas las siguientes palabras:

$$\{Javier, compró, una, casa\} \tag{1}$$

Se puede construir la frase:

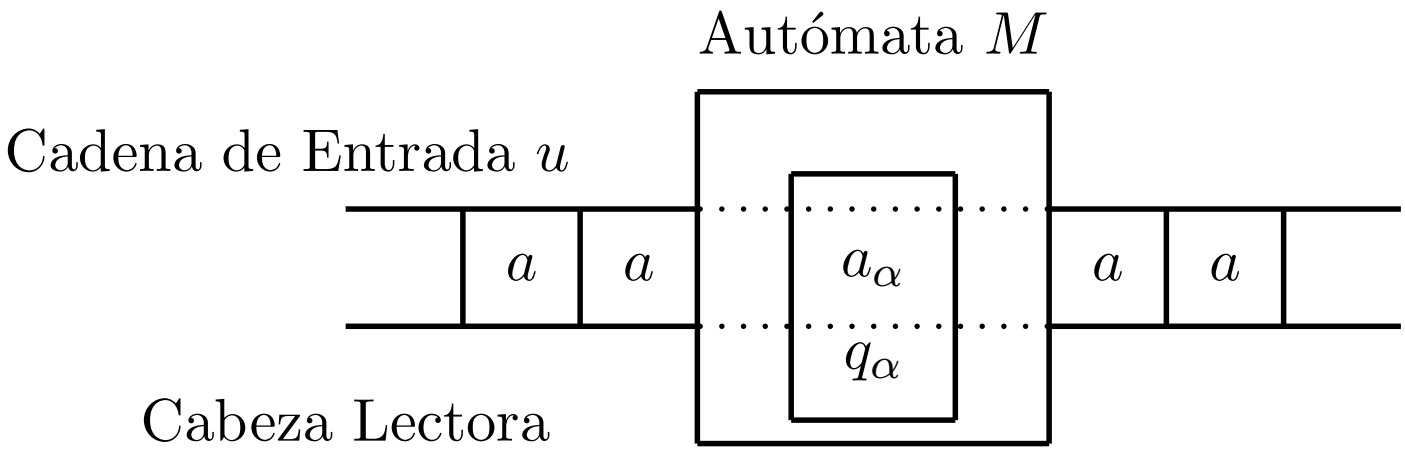
$$Javier\ compró\ una\ casa \tag{2}$$

que sintáctica y semánticamente es correcta, pero la oración:

$$Una\ casa\ compró\ Javier \tag{3}$$

es sintácticamente correcta pero no semánticamente.

Se conoce como Autómata Finito a máquinas abstractas que procesan cadenas para un determinado lenguaje.



I. Cinta semi-infinita: Dividida a su vez en celdas donde se escribe la cadena de entrada.
II. Unidad de Control (también llamada Cabeza Lectora): Que se encarga de procesar la citan.
III. El Autómata propiamente dicho que mantiene la lógica del lenguaje a través de una serie de estados (de aceptación y finales).

Dependiendo de la configuración de los estados internos del Autómata, diferenciamos tres tipos:

i. Autómatas Finitos Determinista: Transiciones del tipo: $\delta(q, a)$. Procesan la palabra λ.

ii. Autómatas Finitos No Determinista: Transiciones del tipo: $\Delta(q, a)$. No procesan la palabra λ.

LEX o Lenguaje de Especificación para Analizadores Léxicos, se trata de un lenguaje que relaciona Expresiones Regulares con acciones determinadas.

La estructura de un programa LEX es la que sigue:

I. Sección de Definiciones: En ella se definen variables, constantes y los patrones necesarios para el resto del programa.

II. Sección de Reglas: Contiene el conjunto de reglas, definidas de la siguiente manera:

$$er_{\lambda} \quad \{sentencias\} \tag{4}$$

III. Sección de Código C: Consiste en una serie de sentencias auxiliares en Lenguaje C que permiten una mayor flexibilidad al desarrollador/programador.

Yacc se trata de un popular "Front-End" para construir compiladores a nivel sintáctico diseñado originalmente por S.C. Johnson en 1970.
El análisis realizado por Yacc es del tipo LALR.

i. Apartado de rutinas en C: Delimitada por los símbolos {%(apertura)%} (cierre) contiene las directivas del preprocesador además, de variables y definiciones necesarias para el resto del programa.
ii. Apartado de Tokens: Establece los Tokens a utilizar en el programa.
iii. Sección de Reglas de Traducción: Se definen en el mismo, las acciones semánticas que se corresponde a su vez con instrucciones en Código C.
iv. Apartado de Código en C: Se trata del conjunto de rutinas en C definidas por el desarrollador/programador.

$$E \longrightarrow E + T \mid T \tag{5}$$

Donde:

i. E: Es un símbolo No Terminal.

ii. T: Es un símbolo Terminal.

```
pascal: pascal.tab.o pascal.lex.o
    (CC) -o gp90c pascal.tab.o pascal.lex.o (LDLIBS)
pascal.lex.o: pascal.lex.c pascal.tab.h
    (CC) -c pascal.lex.c
pascal.tab.o: pascal.tab.c pascal.tab.h
    (CC) -c pascal.tab.c
pascal.tab.c: pascal.y
    (BISON) -d pascal.y
pascal.lex.c: pascal.l
    (FLEX) pascal.l
    mv lex.yy.c pascal.lex.c
```

i.