# Teoría de lenguajes, autómatas y compiladores

Trabajo práctico 2

Generador de Analizador Sintáctico Descendente con Retroceso (ASDR)

ITBA

**Conrado E. F. Mader Blanco** - Legajo 51270
**Tomás Alfredo Mehdi** - Legajo 51014
**Federico Ramundo** - Legajo 51596

# Índice

# 1. Objetivo

El trabajo consiste en programar en C un Generador de Analizador Sintáctico Descendente con Retroce- so(ASDR). Dada una gramática libre de contexto, sin recursividad a izquierda, el programa deberá generar un ASDR que estará escrito en lenguaje C.(ASDR.c). El ASDR podŕa luego recibir una cadena de entrada y determinar si pertenece o no al lenguaje generado por la gramática, mostrando además la derivación que llevó a dicha cadena.

# 2. Funcionamiento del programa

El generador recibe como entrada un archivo de extensión .gr, donde est?a la especificación de una sola gramática que se asume libre de contexto y sin recursividad a izquierda. La sintaxis de ejecución del generador es:

genASDR nombrearchivogramatica

El generador obtiene como salida un archivo ASDR.c que ser?a el código fuente del Analizador Sintáctico Descendente con Retroceso implementado a través de procedimientos.

El ADSR tiene una función principal , una función por cada no terminal de la gramática y una función para procesar la cadena de entrada.
Se utilizó la heurística siguiente: mientras el método intenta diferentes derivaciones, si el conjunto de hojas del árbol es mayor a la longitud de la cadena a analizar entonces se aplica el retroceso.

# 3. Estructura de las gramáticas

Un archivo de gramática (.gr) es un archivo en el que se especifican las gramáticas de la siguiente forma:

NombreDeLaGramatica = ( SimbolosNoTerminales, SimbolosTerminales, SimboloInicial, Producciones)

En esta estructura se tienen las siguientes consideraciones:

- Los SimbolosNoTerminales son letras en mayuścula separadas por comas.
- Los SimbolosTerminales son letras en minuścula separadas por comas.
- Para el símbolo lambda se usa el caracter \(barra invertida).
- Los caracteres del archivo pueden (o no) estar separados por espacios, tabuladores o fines de linea.
- Se recibe una sola gramática por archivo.
- El nombre de la gramática empieza por una letra.(después puede seguir cualquier símbolo, sin espacios.

# 4. Funcionamiento del ASDR

El analizador recibe como entrada una palabra que podría pertenecer al lenguaje es decir, una cadena formada sólo por letras minúsculas. La sintaxis de ejecución del analizador es:

ASDR palabra

El programa ASDR devuelve '*palabra* no pertenece' si la cadena no pertenece o '*palabra* pertenece' y a continuación el conjunto de producciones que dieron lugar a dicha palabra, y en el orden que fueron utilizadas.

# 5. Consideraciones realizadas

Para ejecutar el *parsing* de la entrada de archivos se utilizó LEX tal y como hicimos con el trabajo anterior.
Para realizar el funcionamiento del trabajo se utilizó el algoritmo provisto por la cátedra en la teoría. Implementamos el algoritmo provisto y adaptamos las estructuras definidas por el trabajo anterior. Antes, al evaluar sólo gramáticas regulares guardábamos como mucho dos caracteres en la parte derecha de las producciones. Ahora al ser libres de contexto pueden venir varios caracteres.

# 6. Diseño y desarrollo

## 6.1. Análisis de la gramática

Tal y como se mencionó anteriormente, para controlar que el archivo de entrada sea correcto nos proveemos de la herramienta LEX. A su vez, la utilizamos para armar la estructura de la gramática en memoria.
Una vez obtenida la gramática se procede a generar el archivo compilabre *ASDR.c*. Para ello se imprime en un archivo lineas de código C, en base a la estructura de la gramática.

## 6.2. Procesamiento de cadenas

Para procesar las cadenas se tiene una función que retorna un booleano que procesa una cadena, basándose en el algoritmo visto en clase: Se genera una función por cada símbolo no terminal y se va consumiendo la cadena a medida que se avanza por las funciones. Si se produce un error entonces no se acepta esa cadena. Si por otro lado se llega al final de la cadena y las funciones terminan positivamente, se acepta la cadena.
Se guarda, además, las producciones utilizadas para luego poder imprimirlas e indicar al usuario cual es el camino a seguir para obtener dicha cadena.

## 7. Dificultades encontradas

La principal limitación encontrada es que nuestro programa utiliza una función de LEX que funciona en Ubuntu pero no en Mac OSX. Esta función es *gr_switch_to_buffer*. Desconocemos el funcionamiento interno de LEX así que no supimos como arreglarlo, pero no supuso un problema dado que todos poseemos Ubuntu.
En general podemos decir que este trabajo nos pareció más fácil que el anterior, dado que ya nos defendíamos mejor con LEX. Además, el método de análisis descendente lo teníamos fresco del parcial que dimos hace sólo unas pocas semanas, y nos acordábamos bastante del procedimiento a seguir.
El lenguaje C no genera problemas dado que ya estamos acostumbrados a programar con él desde hace 3 años de carrera.

## 8. Futuras extensiones

Una posible extensión podría ser desarrollar un analizador sintćtico ascendente. Para ello se tendría que cambiar el funcionaminto de varias funciones para seguir el algoritmo correspondiente con los analizadores ascendentes. Resultaría interesante que el usuario pudiese elegir a la hora de ingresar una gramática si la desea analizar ascendente o descenentemente.
Otra posible extensión podría ser que el programa mantenga un log de las producciones que fue probando y las reducciones que realizó, para darse una idea de si la heurística utilizada es buena o si al analizador le está costando mucho llegar a la conclusión.

## 9. Ejemplo de uso

Luego de hacer make, se genera el ASDR.c con una gramática.
Dada la gramática:

G1 = (A, B, C, a, b, c,A, A ->aBC|cB, B ->aA|b, C ->c|\)

Se ejecuta el comando *./bin/genASDR ./gr/ejemplo.gr* y se obtiene un archivo llamado ASDR.c.
A continuación se compila con la linea *gcc -o gram1 ASDR.c* y se ejecuta pasándole una cadena, tal y como se ve en la 1.
Si se le pasa una gramática inválida, como por ejemplo, con recursividad izquierda, el programa lo indica a la hora de intentar generar el ASDR.c.

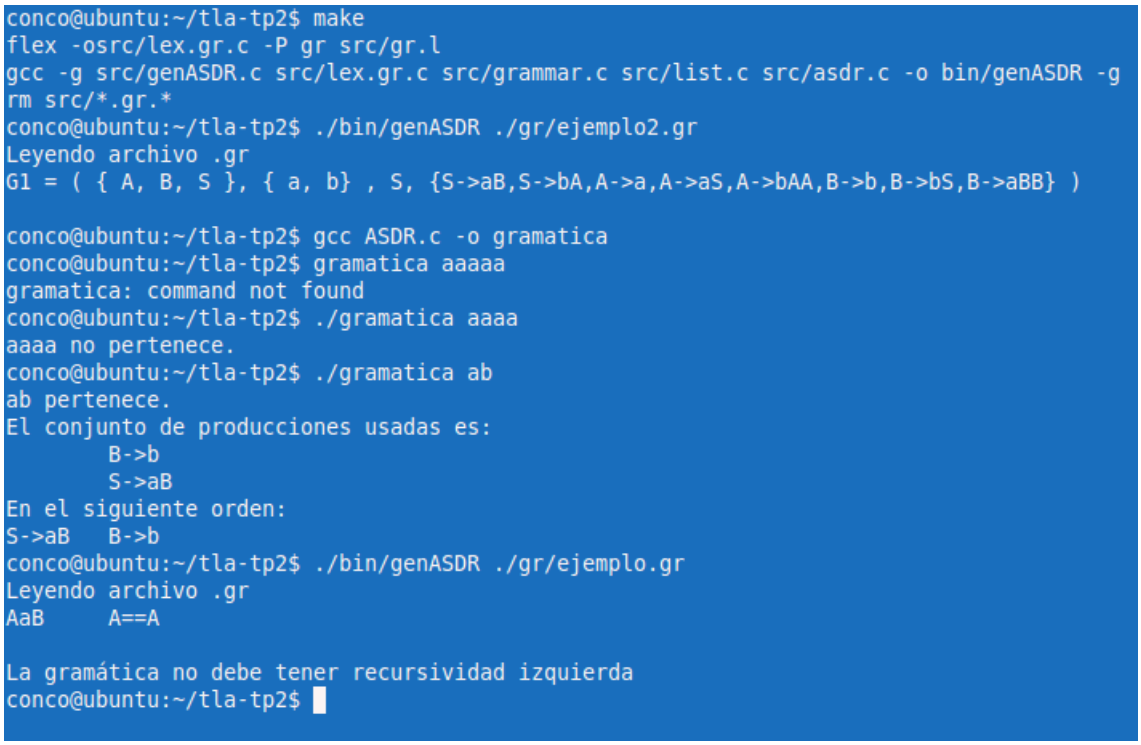


Figura 1: Ejemplo de uso