Laboratorio 2: Gramática Libre de Contexto

Teoría de Lenguajes

28 de mayo de 2024

1 Introducción

Este laboratorio consiste en usar la biblioteca NLTK [1] para Python [2] con el objetivo de que el estudiante diseñe e implemente Gramáticas Libre de Contexto (GLC) como analizadoras de lenguajes.

La propuesta consiste de 3 programas a implementar. Cada programa recibe una entrada que es procesada con una GLC y despliega una salida. En cada programa se deberá construir una gramática para reconocer la entrada, basándose en la descripción dada en esta letra y un conjunto de ejemplos de entrada y salida.

2 Modo de trabajo

Cada programa a implementar es un módulo **Python 3** ejecutable que recibe como parámetros los nombres de los archivos de entrada y salida.

Los programas pedidos, excepto que se indique lo contrario, analizan sintácticamente la entrada con una gramática libre de contexto. Las posibles salidas de los programas son:

- **NO PERTENECE**, en caso de que la entrada no pertenezca al lenguaje generado por la gramática.
- **NO PERTENECE - FUERA DE ALFABETO**, en caso de que

algún símbolo de la entrada no esté incluido en los símbolos del alfabeto del lenguaje.

- **PERTENECE**, en caso de que la entrada pertenezca al lenguaje generado por la gramática.

Se permite importar variables y funciones entre los módulos de los programas implementados pero no crear módulos auxiliares.

Para realizar este laboratorio se imparte el archivo **lab2.zip** que contiene ejemplos de entrada y salida, el módulo *programa0.py* y el script (*test.py*) que ejecuta cada programa con las entradas disponibles y compara las salidas obtenidas con las esperadas.

3 Programas a implementar

Utilizando el módulo ***NLTK***, que dispone de gramáticas libre de contexto, implemente en Python 3 los siguientes programas:

3.1 programa0.py

El programa 0 se encuentra implementado como guía para la implementación de los restantes programas.

El programa verifica que la tira de entrada pertenezca al lenguaje $\{0^n1^n : n > 0\}$.

La gramática que se entrega implementada (con la sintáxis de NLTK) es:

```
S -> '0' S '1'| '0' '1'
```

Ejemplo 1:

```
0 0 0 1 1 1
```

Salida:

```
PERTENECE
```

Ejemplo 2:

```
0 0 1
```

Salida:

```
NO PERTENECE
```

Ejemplo 3:

```
0 0 b b
```

Salida:

```
NO PERTENECE - FUERA DE ALFABETO
```

3.2 programa1.py

Este programa reconoce al lenguaje $\{0^n1^m2^k : (n = m \vee m = 2k \vee n = 3k) \wedge min(n, m, k) > 0\}$.

Ejemplo 1:

```
0 0 1 1 2 2 2
```

Salida:

```
PERTENECE
```

Ejemplo 2:

```
0 0 1 2
```

Salida:

```
NO PERTENECE
```

Ejemplo 3:

```
0 0 1 1 2 2 3
```

Salida:

```
NO PERTENECE - FUERA DE ALFABETO
```

3.3 programa2.py

Este programa reconoce operaciones matemáticas en notación infija (e.g. 3 + 4) y las escribe en notación prefija (e.g. +(3, 4)).

Las operaciones están formadas por los siguientes elementos:
- números naturales del 0 al 100
- operadores: +, -, * y /

Ejemplo 1:

```
(3 + 4) * 2
```

Salida:

```
*(+(3,4),2)
```

Ejemplo 2:

```
3
```

Salida:

```
3
```

Ejemplo 3:

```
+ 3
```

Salida:

```
NO PERTENECE
```

Ejemplo 4:

```
6 ^ 3
```

Salida:

```
NO PERTENECE - FUERA DE ALFABETO
```

3.4 programa3.py

En este programa se reconocen líneas de código válidas según una sintaxis inspirada en la de Python. Las líneas de código serán analizadas una a una,

de manera independiente, y no en conjunto [1].

El reconocedor a implementar debe tener en cuenta los siguientes elementos:

- Variables: Las variables serán definidas únicamente por una letra minúscula de la *a* a la *z* (e.g. "a" es una variable, "ab" **no** es una variable, la tira vacía tampoco).
- Números: Se utilizarán los números naturales del *0* al *100*.
- Operadores booleanos: menor (<), menor o igual (<=), mayor (>), mayor o igual (>=), igual (==) y distinto (!=).
- Operadores de asignación: =, += y -=.
- Expresiones booleanas: estas estarán compuestas por números (n), variables (v) y operadores booleanos (o) de la siguiente manera: n o n (e.g. 3 == 10), n o v (e.g. 3 > a), v o n (e.g. a != 5), v o v (e.g. a <= b). Las expresiones booleanas no pueden ser vacías.

Las líneas de código podrán contener:

- Estructuras de control: if, elif, while. Estas estructuras estarán seguidas siempre por paréntesis "()" y terminarán con ":".
  Dentro de los paréntesis habrá una expresión booleana como fue definida anteriormente (e.g. "if (a > 5):").
- Estructuras de control: for. Esta estructura estará seguida siempre por paréntesis "()" y terminarán con ":"
  Dentro de los paréntesis habrá algo de la forma (<variable> in range(<número>):) (e.g. "for (i in range(10):"). Las variables y los números son los definidos anteriormente.
- Asignaciones: estas serán compuestas por variables (v), operadores de asignación (o) y números (n) de la siguiente manera: v o n (e.g. a=3), v o v (e.g. b+=c).

Ejemplo 1:

[1] Es decir, en ningún caso se evaluará un conjunto de líneas relacionadas, como por ejemplo un bloque "for".

```
    elif (b != c):
```

Salida:

```
    PERTENECE
```

Ejemplo 2:

```
    3 += 4
```

Salida:

```
    NO PERTENECE
```

Ejemplo 3:

```
    a *= 4
```

Salida:

```
    NO PERTENECE - FUERA DE ALFABETO
```

Ejemplo 4:

```
    for (a in range(b)):
```

Salida:

```
    NO PERTENECE
```

Ejemplo 5:

```
    while (4 > 1):
```

Salida:

```
    PERTENECE
```

Ejemplo 6:

```
if (b # 1):
```

Salida:

```
NO PERTENECE - FUERA DE ALFABETO
```

4 Entrega

La fecha límite para la entrega es el **28 de junio a las 23:59**.

Se habilitará un formulario en EVA para realizar la entrega.

Los grupos no tienen por que ser los mismos que en el laboratorio 1, pero deberán tener ente 2 y 4 integrantes. No se aceptarán entregas individuales.

En caso de precisar rearmar un grupo, el foro de "Busco grupo" sigue estando disponible para la conformación de grupos.

Se debe entregar:

- Los archivos **programa1,2,3.py** con los programas implementados
- Un archivo **integrantes.txt** con las cédulas (sin puntos ni dígito de verificación) y nombre de los integrantes del grupo, uno por línea, separado por comas como se muestra en el ejemplo. La primera línea debe contener la cantidad de integrantes del grupo. Opcionalmente, se puede utilizar el final de este archivo para comentarios que el grupo considere pertinentes.

Ejemplo de archivo integrantes.txt:

```
3
7123456, ApellidoA1 ApellidoA2, NombreA1 NombreA2
8654321, ApellidoB1 ApellidoB2, NombreA1 NombreB2
9876543, ApellidoC1 ApellidoC2, NombreC1 NombreC2
Hicimos dos soluciones para el programa3, entregamos la que consideramos mejor y dejamos comentada la otra.
```

5 Referencias

1. NLTK
2. Python