

# Programa 2.1. Convertidor de un AP a gramática

**Alumno: Raziel López Escamilla.**

## Descripción general.

PAToG.py es un script que genera la gramática de un automata de pila, dicho script necesita recibir 2 parámetros de entrada, el nombre del archivo donde se encuentran las descripciones del autómata de pila y el nombre del archivo en donde se desea escribir las reglas gramaticales correspondientes al autómata de pila de entrada, ambos archivos se leeran o generaran en el mismo directorio donde se encuentra localizado el script principal (PAToG.py).

## Contenido básico para la ejecución de script.

- PAToG.py
  - Script principal encargado de obtener gramática a partir de un autómata de pila.
- pushdownAutomaton.py
  - Clase que define objeto de tipo PushdownAutomaton.
- Grammar.py
  - Clase que define objeto de tipo Grammar.
- PA1.json
  - Ejemplo de automata de pila1.
- PA2.json
  - Ejemplo de automata de pila2.
- PA3.json
  - Ejemplo de automata de pila3.

## Descripción de archivo de autómata de pila (entrada).

El archivo que describe al autómata de pila cumple con un formato JSON (Java Script Object Notation) y debe de seguir las siguiente sintaxis.

```
{
  "S" : ["f", "g", "h"],
  "F" : ["h"],
  "I" : "f",
  "E" : [
    ["f", "c", "l", "g", "c"],
    ["g", "b", "l", "g", "l"],
    ["g", "c", "c", "h", "l"]
  ]
}
```

1. La descripción del autómata de pila debe de estar contenida en 2 llaves {}

2. Indicar conjunto de estados, ejemplo:  
`"S" : ["estado1","estado2", ... , "estadoN"]`,
  - a. El nombre de cada estado debe de estar entre comillas.
3. Indicar conjunto de estados de aceptación, ejemplo:  
`"F" : ["estado1","estado2", ... , "estadoN"]`,
  - a. El nombre de cada estado debe de estar entre commillas.
4. Indicar estado de inicio, ejemplo:  
`"I" : "estado1"`,
  - a. El nombre del estado inicial NO debe de estar entre corchetes.
5. Indicar las aristas que existen en el autómata.  
`"E" : [Arista1,Arista2, ... , AristaN]`,
  - a. Cada arista sigue el siguiente formato:  
`["Fuente","CaracterLeido","SimbPilaExtraido","Destino","SimPilaInsertado"]`
    - b. No olvidar colocar comas entre cada arista.
    - c. El simbolo  $\lambda$  se debe representar con l.
6. No olvidar comas entre cada conjunto.

NOTA: El nombre del archivo puede ser de cualquier extensión.

### Descripcion de archivo de gramática generada (salida).

El archivo que describe la gramática generada por el autómata de pila cumple con un formato JSON (Java Script Object Notation) y debe de seguir las siguiente sintaxis.

```
{"grammar_rules_list":
  [
    [{"S"}, [{"f", "l", "h"}]],
    [{"f", "l", "f"}, [{"l"}]],
    [{"g", "c", "h"}, [{"c", ["h", "l", "h"}]],
    [{"g", "c", "g"}, [{"b", ["g", "l", "f"], ["f", "c", "g"}]],
  ]
}
```

1. La descripción de la gramática debe de estar contenida en dos llaves {}
2. Indicar conjunto de reglas, ejemplo :  
`Grammar_rule_list : [regla1,regla2, ... ,reglaN]`
3. Cada regla sigue alguno de los siguientes patrones  
`[NoTerminal] , [Terminal]`

Ejemplo:

```
[["f", "l", "f"], ["l"]]
["f", "l", "f"] – NoTerminal
["l"] – Terminal
```

Se puede traducir como:

$$\langle f, \lambda, f \rangle \rightarrow \lambda$$

`[NoTerminal], [NoTerminal]`

Ejemplo:

$[[\text{"S"}], [\text{"f"}, \text{"l"}, \text{"h"}]],$   
 $[\text{"S"}] - \text{NoTerminal}$   
 $[\text{"f"}, \text{"l"}, \text{"h"}] - \text{NoTerminal}$

El estado inicial siempre estará denotado como "S"

Se puede traducir como:

$$S \rightarrow \langle f, \lambda, h \rangle$$

$[No\ terminal], [Terminal], [NoTerminal]]$

Ejemplo:

$[[\text{"g"}, \text{"c"}, \text{"h"}], [\text{"c"}, [\text{"h"}, \text{"l"}, \text{"h"}]]],$

Donde:

$[\text{"g"}, \text{"c"}, \text{"h"}] - \text{NoTerminal}$   
 $\text{"c"} - \text{Terminal}$   
 $[\text{"h"}, \text{"l"}, \text{"h"}] - \text{NoTerminal}$

Se puede traducir como:

$$\langle g, c, h \rangle \rightarrow c \langle h, \lambda, h \rangle$$

$[No\ terminal], [Terminal], [NoTerminal], [NoTerminal]]$

Ejemplo:

$[[\text{"g"}, \text{"c"}, \text{"g"}], [\text{"b"}, [\text{"g"}, \text{"l"}, \text{"f"}], [\text{"f"}, \text{"c"}, \text{"g"}]]],$

Donde:

$[\text{"g"}, \text{"c"}, \text{"g"}] - \text{NoTerminal}$   
 $\text{"b"} - \text{Terminal}$   
 $[\text{"g"}, \text{"l"}, \text{"f"}] - \text{NoTerminal}$   
 $[\text{"f"}, \text{"c"}, \text{"g"}] - \text{NoTerminal}$

Se puede traducir como:

$$\langle g, c, g \rangle \rightarrow b \langle g, \lambda, f \rangle \langle f, c, g \rangle$$

### Ejemplo de uso típico.

Ejecutar desde línea de comandos.

Python3 PAToGo.py Automata.json Gramatica.json

Ejemplo:

```
192:Tarea9 raziel$ python3 PAToG.py PA1.json G1.json
```

El script genera la gramática y también imprime información en la consola de la gramática generada.

## Ejemplos de autómatas de pila.

En el directorio se encuentran 3 ejemplos de autómatas de pila que pueden ser utilizados con el script.

### **Autómata de pila1 (PA1.json):**

Acepta cadenas que inicien con una “c” seguida de cualquier cantidad de “b” y termina con otra “c”

Ejemplo:

“cbbbc”

### **Autómata de pila 2 (PA2.json):**

Acepta cadenas que inicien con una “x” seguida de cualquier combinacion de “x” o “y” siempre y cuando tengan la misma cantidad de “x” que de “y” y la cantidad de “x” a la izquierda siempre sea mayor que la cantidad de “y” y termina con una “y”.

Ejemplo:

“xxyxyxyyy”

### **Autómata de pila 3 (PA3.json):**

Acepta cadenas con cantidades pares de “x”

Ejemplo:

“xxxx”

## Programa 2.2. Evaluador de una gramática

**Alumno: Raziel López Escamilla.**

### Descripción general.

GrammarEvaluation.py es un script que evalúa una cadena de acuerdo con reglas gramaticales específicas, este script necesita como parámetros de entrada el nombre del archivo que contiene las reglas gramaticales y la cadena que se desea evaluar, idealmente se debe utilizar la gramática generada por el programa PAToG.py debido a su formato específico (para más información acerca del formato de las reglas gramaticales, consulta las especificaciones del script PAToG.py), como salida el programa indicará si la cadena es aceptada por el lenguaje de las reglas gramaticales y en caso de serlo indica la forma en como deben utilizarse las reglas gramaticales para aceptar la cadena de entrada.

### Contenido básico para la ejecución de script.


- GrammarEvaluation.py
  - Script principal encargado de evaluar la gramática.
- pushdownAutomaton.py
  - Clase que define objeto de tipo PushdownAutomaton.
- Grammar.py
  - Clase que define objeto de tipo Grammar.
- G1.json
  - Ejemplo de gramática.
- G2.json
  - Ejemplo de gramática.
- G3.json
  - Ejemplo de gramática.

### Ejemplo de uso típico.

Ejecutar desde línea de comandos.

```
Python3 GrammarEvaluation.py [Gramatica] [cadena]
```

Ejemplo:



```
Phoenix:Tarea2_3 raziel$ python3 GrammarEvaluation.py G2.json xxyxyxyxyy
```

El script indica si la cadena es aceptada por la gramática, y en caso de serlo también imprime la secuencia de reglas que se utilizan para poder comprobar que acepta la cadena.

Ejemplo de salida:

G3 contiene las reglas gramaticales de un lenguaje que acepta solo cantidades pares de x.

```
python3 GrammarEvaluation.py G3.json xx
```

```
Grammar File name: G3.json
String for evaluation: xx
Importing rules from G3.json ...
evaluating input , this might take a while ...
eureka!
lambda = 'l'
Initial rule:
['S'] -> [['a', 'l', 'c']]
rule used:
    ['a', 'l', 'c'] -> ['l', ['b', '#', 'c'], ['c', 'l', 'c']]
substitution:
['S'] -> [['b', '#', 'c'], ['c', 'l', 'c']]
rule used:
    ['b', '#', 'c'] -> ['x', ['b', '@', 'b'], ['b', '#', 'c']]
substitution:
['S'] -> ['x', ['b', '@', 'b'], ['b', '#', 'c'], ['c', 'l', 'c']]
rule used:
    ['b', '@', 'b'] -> ['x', ['b', 'l', 'b']]
substitution:
['S'] -> ['x', 'x', ['b', 'l', 'b'], ['b', '#', 'c'], ['c', 'l', 'c']]
rule used:
    ['b', 'l', 'b'] -> ['l']
substitution:
['S'] -> ['x', 'x', ['b', '#', 'c'], ['c', 'l', 'c']]
rule used:
    ['b', '#', 'c'] -> ['l', ['c', 'l', 'c']]
substitution:
['S'] -> ['x', 'x', ['c', 'l', 'c'], ['c', 'l', 'c']]
rule used:
    ['c', 'l', 'c'] -> ['l']
substitution:
['S'] -> ['x', 'x', ['c', 'l', 'c']]
rule used:
    ['c', 'l', 'c'] -> ['l']
substitution:
['S'] -> ['x', 'x']
```

```
python3 GrammarEvaluation.py G3.json xxx
```

```
Grammar File name: G3.json
String for evaluation: xxx
Importing rules from G3.json ...
evaluating input , this might take a while ...
String not valid for this grammar rules
```