Teoría de Conjuntos

Como el lector se pudo dar cuenta los autómatas no son más que una herramienta matemática, y para ello necesitamos conocer la base en la cual se sustenta y esa es “LA TEORÍA DE CONJUNTOS”

Que es un conjunto:

Un conjunto es una colección de elementos, el conjunto puede tener “ninguno”, “uno” o “muchos” elementos… NOTA: los elementos de un conjunto no deben repetirse.

Para indicar un conjunto lo escribimos entre llaves { }

Ejemplo 01 - conjunto de las vocales

_vocales = {‘a’, ‘e’, ‘i’, ‘o’, ‘u’}

Ejemplo 02 - Conjunto de los números

_numeros = {1,2,3,4,5,6,7,8,9,0}

OPERACIÓN 1 - PERTENENCIA

Para los conjuntos existen varias operaciones y para ello vamos empezar con la más simple que es la pertenencia… y entonces siempre hay 2 posibilidades para todo conjunto:

- Un elemento “?” **pertenece** al conjunto.
- Un elemento “?” **no** pertenece al conjunto.

Ejemplos:

- Para el conjunto de las _vocales la letra ‘Z’ no pertenece.
- Para el conjunto de los _numeros la letra ‘Z’ no pertenece.
- Para el conjunto de las _vocales la letra ‘A’ no pertenece, y esto es debido a que tenemos que distinguir entre mayúsculas o minúsculas.
- Para el conjunto de los números el carácter ‘7’ no pertenece, y esto es debido a que tenemos que distinguir el 7 como un número al ‘7’ como un carácter.

OPERACIÓN 2 - UNIÓN

Es una operación elemental en la cual se unen 2 conjuntos y el resultante es un nuevo conjunto con los elementos de los conjuntos anteriores.

A u B: Son todos los elementos que están en A y en B.

Ejemplo:

```
_vowels = {'a', 'e', 'i', 'o', 'u'}
_numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}

_vowels_union_numbers = _vowels | _numbers
print(f"Vowles UNION NUMBERS: {_vowels_union_numbers}")
```

```
Vowles UNION NUMBERS: {'u', 'o', 0, 1, 2, 3, 4, 5, 'a', 'e', 6, 7, 8, 9, 'i'}
```

Supongamos que tenemos 2 autómatas:

L1: Son todas las palabras que terminan en “**a**”.
L2: Son todas las palabras que empiezan por “**b**”.

Entonces la unión de ambos conjuntos (**L1** u **L2**) serán todas las palabras que empiezan por “b” o terminan en “a”.

OPERACIÓN 3 - INTERSECCIÓN

Esta operación toma lo que 2 conjuntos tienen en común y crea un nuevo conjunto.

A ∩ B: Son todos los elementos que están en A y en B.

Ejemplo:

```
_numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}
_other = {0, '1', 7, 'u'}

_numbers_intersection_other = _numbers & _other
print(f"Number INTERSECTION Other: {_numbers_intersection_other}")
```

```
Number INTERSECTION Otjer: {0, 7}
```

OPERACIÓN 4 - DIFERENCIA

Esta operación toma todos los elementos que hacen parte de uno pero no del otro. Ósea de un conjunto se eliminan todos los elementos que coinciden con el otro.

A - B: Son todos los elementos de A que no están en B.

Ejemplo:

```
_vowels = {'a', 'e', 'i', 'o', 'u'}
_other = {0, '1', 7, 'u'}

_vowels_subtraction_other = _vowels - _other
print(f"Vowels SUBTRACTION Other: {_vowels_subtraction_other}")
```

```
Vowels SUBTRACTION Other: {'i', 'e', 'o', 'a'}
```

OPERACIÓN 5 - SUB-CONJUNTO

Más que una operación es decir que todos los elementos de un conjunto ya se encuentran en otro.

A ⊆ B: Todo B está contenido en A.

Ejemplo:

```
_numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}
_even_numbers = {2,4,6,8,0}

_even_is_contained_in_numbers = _even_numbers.issubset(_numbers)
print(f"Even is contained in numbers? RTA: {_even_is_contained_in_numbers}")
```

```
Even is contained in numbers? RTA: True
```

OPERACIÓN 6 - POTENCIA

Es un nuevo conjunto con todas las combinaciones posibles de otro conjunto.

Siendo A: {a, b} Entonces
P(A): {{}, {a}, {b}, {a, b}}

Ejemplo:

```
def generateElementsEnumeration(set):
    _order = []

    for i in set:
        _order.append(i)

    return _order


def generateCombinatory(arr, n):
    result = []

    def backtrack(start, current):
        if len(current) == n:
            result.append(current.copy())
            return

        for i in range(start, len(arr)):
            current.append(arr[i])
            backtrack(i + 1, current)
            current.pop()

    backtrack(0, [])
    return result


def generatePowerSet(_set):
    PA = [set()]
    return _generatePowerSet(len(_set), PA, generateElementsEnumeration(_set))


def _generatePowerSet(iterator, PA, enum):
    if iterator == 0:
        return PA

    for i in generateCombinatory(enum, iterator):
        PA.append(set(i))
```

```
    iterator = iterator - 1
    return _generatePowerSet(iterator, PA, enum)


_set = {'a', 'b'}
print(f"Power SET of {_set}:")
print(generatePowerSet(_set))
```

```
Power SET of {'b', 'a'}:
[set(), {'b', 'a'}, {'b'}, {'a'}]
```

OPERACIÓN 7 - COMPLEMENTO

Se podría ver como el resto del universo, osea más allá de negar (Que en algunos casos se cumple) se interpreta como todo lo del universo excepto el conjunto original.

$\overline{A} = U - A$

Ejemplo:

```
_vowels = {'a', 'e', 'i', 'o', 'u'}
set_a = {'a'}

print(f"Complement of SET A: {_vowels - set_a}")
```

```
Complement of SET A: {'o', 'e', 'u', 'i'}
```

Conjuntos en Autómatas

Para un autómatas empezaremos describiendo los conjuntos más obvios:

- El conjunto _alfabeto
- El conjunto de _estados
- El conjuntos de _estadosFinales
- El conjunto de _transiciones

Cada autómata puede definirse como una quíntupla… en donde cada elemento es un conjunto.

En autómatas existen 3 clases de operaciones:

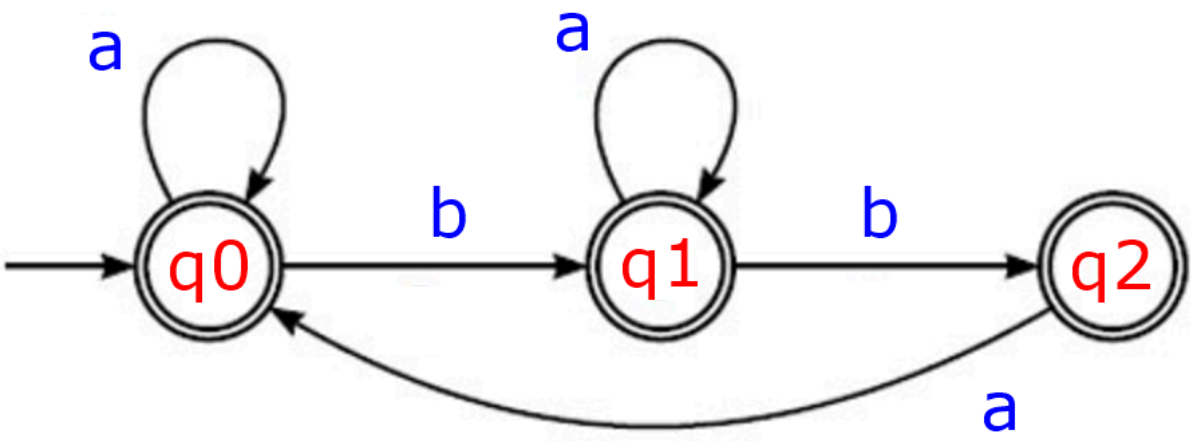
OPERACIÓN 0.1 - PERTENENCIA DE SÍMBOLO A ALFABETO

Un símbolo puede ser procesado por un autómata sólo si pertenece a su alfabeto. Por ejemplo si tenemos el siguiente alfabeto:

_alphabet = {‘a’, ‘b’}

Entonces la cadena ‘ababba’ es válida debido a que cada uno de sus elementos está contenido en el alfabeto. Por otra parte la cadena ‘aaacbba’ NO ES VÁLIDA debido a que contiene el elemento ‘c’ el cual no pertenece al alfabeto.

OPERACIÓN 0.2 - SUBCONJUNTO DE ESTADOS FINALES PERTENECIENTES A TODOS LOS ESTADOS



Todo autómata contiene conjuntos de estados para este caso Q = {q0, q1, q2} y F = {q0, q1, q2} por ello se puede concluir que F⊆Q y se puede evidenciar que si en caso de:

F = {q3} pues entonces F ⊄ Q

Osea para este último caso si suponemos que F que contuviera el estado “q3” pues entonces no pertenecería a el conjunto de estados Q debido a que Q contiene “q0, q1, q2”.

OPERACIÓN 1 - UNIÓN DE LENGUAJES

Supongamos que tenemos 2 lenguajes:

_L1 = {0, 1}
_L2 = {0,1,2,3,4,5,6,7,8,9}

Entonces para la cadena “010101” cumpliria con ambos lenguajes… pero para la cadena “9000” solo cumpliria con _L2 por ello la unión entre lenguajes nos dará un tercer conjunto que de todos modos está contenido en _L2.

Otro ejemplo sería:

Supóngase que se tiene 2 Autómatas:

- AFD1: Reconoce todas las palabras que empiezan por ‘a’.
- AFD2: Reconoce todas las palabras que terminan en ‘b’.

La unión de los lenguajes de los autómatas AFD1 y AFD2 serían todas las palabras que empiezan en ‘a’ o terminan en ‘b’

OPERACIÓN 2 - INTERSECCIÓN DE LENGUAJES

La intersección de lenguajes es la operación por la cual dos lenguajes se tienen que cumplir al tiempo. Por ejemplo suponga que se tiene 2 lenguajes:

_L1 = {‘ab’, ‘ba’, ‘aa’}
_L2 = {‘aa’, ‘bb’, ‘ab’}

La intersección de ambos lenguajes sería:

_L3 = _L1 y _L2
_L3 = {‘ab’, ‘aa’}

Otro ejemplo sería:

Supóngase que se tienen autómatas:

- AFD1: Reconoce todas las palabras que tienen un número par de ‘a’
- ADF2: Reconoce todas las palabras que tiene al menos una ‘b’

La unión de los lenguajes de los autómatas AFD1 y AFD2 serían todas las palabras que empiezan contienen un número par de ‘a’ y por lo menos 1 ‘b’ o más.

OPERACIÓN 3 - COMPLEMENTO

$$A = (Q, \Sigma, \delta, q_0, F)$$

```python
class Automaton:
    def __init__(self, states, alphabet, transitions, q0, F):
        self.states = states
        self.alphabet = alphabet
        self.transitions = transitions
        self.q0 = q0
        self.F = F
        self.pivot = q0
```

El complemento de un lenguaje ‘LC’ en autómatas es el conjunto de todas las cadenas del alfabeto las cuales no pertenecen a ‘Σ’.

Ejemplo:

_ALPHABET = {‘a’, ‘b’}
_L1 = {‘a’} # Todas las palabras que sean solo ‘a’ … {a, aa, aaa, a…}

Entonces su complemento es:

LC = Σ - _L1

Osea LC = {vacío, ‘ab’, ‘ba’} # Todas las palabras que contengan B y el vacío.