Informe sobre Operaciones de Conjuntos Difusos

1. Operación Complemento

En la implementación de la operación complemento de conjuntos difusos se recibe como parámetro un conjunto difuso con notación `[(Double, Int)]`, que es un conjunto de pares. El primer dato de tipo `Double` representa el grado de pertenencia (que se sitúa entre 0 y 1), y el segundo dato de tipo `Int` es el elemento del conjunto.

Como primer paso, se recorre el conjunto y para cada elemento de tipo `Double` (grado de pertenencia), se resta dicho valor de 1, obteniendo el complemento del conjunto. El complemento de un conjunto difuso está dado por $1 - \text{grado de pertenencia}$. Finalmente, se agrega a un nuevo conjunto (conjunto complemento) las parejas `[(complemento, elemento)]`.

1.1. Ejemplo de ejecución

Considere el siguiente conjunto difuso:

```
[(0.8, 2), (0.5, 1), (0.3, 3), (0.7, 5)]
```

Hallando el complemento:

```
[(1 - 0.8, 2), (1 - 0.5, 1), (1 - 0.3, 3), (1 - 0.7, 5)]
```

El conjunto complemento es:

```
[(0.2, 2), (0.5, 1), (0.7, 3), (0.3, 5)]
```

2. Operación Inclusión

La función `inclusion` toma dos conjuntos de tuplas de tipo `Set[(Double, Int)]` como parámetros y devuelve un valor booleano que indica si todos los elementos del primer conjunto están "contenidos" en el segundo conjunto bajo ciertas condiciones. La función verifica que para cada elemento de un conjunto, el valor asociado de tipo `Double` en el primer conjunto sea menor o igual que el valor correspondiente en el segundo conjunto.

2.1. Descripción de la Función

La función `inclusion` tiene la siguiente definición:

```
def inclusion( cd1: Set[(Double, Int)], cd2: Set[(Double, Int)]) : Boolean = {
    val mapaCd2 = cd2.toMap
    cd1.forall { case (grado1, elemento) =>
      mapaCd2.get(elemento) match {
        case Some(grado2) => grado1 <= grado2
        case None => false
      }
    }
}
```

2.2. Parámetros

- **cd1**: Un conjunto de tuplas de tipo `Set[(Double, Int)]`, donde cada tupla contiene:
  - Un valor de tipo `Double` (denominado `grado1`).
  - Un valor de tipo `Int` (denominado `elemento`).
- **cd2**: Otro conjunto de tuplas de tipo `Set[(Double, Int)]`, similar a `cd1`.

2.3. Funcionamiento de la Función

1. La función comienza convirtiendo el conjunto `cd2` a un mapa `mapaCd2`:

   `val mapaCd2 = cd2.toMap`

   Este paso convierte el conjunto `cd2` en un mapa de tipo `Map[Int, Double]`, donde las claves son los elementos de tipo `Int` y los valores asociados son los valores de tipo `Double` correspondientes a esos elementos.

2. Luego, para cada tupla `(grado1, elemento)` en `cd1`, la función busca el valor `grado2` en `mapaCd2` asociado a `elemento`. Se realiza una verificación con la siguiente estructura:

   `mapaCd2.get(elemento) match {`

   Si el valor `elemento` está presente en `mapaCd2`, la función compara `grado1` con `grado2`:

   - Si `grado1 <= grado2`, se continúa con la siguiente tupla de `cd1`.
   - Si `elemento` no está presente en `mapaCd2`, se retorna `false` inmediatamente.

3. La función utiliza `forall`, una función de orden superior que evalúa si todos los elementos de `cd1` cumplen con la condición. Si al menos un elemento no cumple con la condición, se retorna `false`. Si todos los elementos cumplen la condición, la función retorna `true`.

3. Ejemplos de Funcionamiento

A continuación, se presentan algunos ejemplos para ilustrar cómo funciona la función `inclusion`.

3.1. Ejemplo 1: Caso donde todos los elementos cumplen con la condición

Considerando los siguientes conjuntos:

$$\texttt{cd1} = \{(1{,}2, 3), (2{,}5, 1), (3{,}0, 2)\}$$

$$\texttt{cd2} = \{(3{,}0, 1), (2{,}5, 3), (4{,}0, 2)\}$$

El resultado de ejecutar `inclusion(cd1, cd2)` sería `true`, porque:

- El elemento $(1{,}2, 3)$ está en `cd2` con un grado de 2,5, y $1{,}2 \leq 2{,}5$.
- El elemento $(2{,}5, 1)$ está en `cd2` con un grado de 3,0, y $2{,}5 \leq 3{,}0$.
- El elemento $(3{,}0, 2)$ está en `cd2` con un grado de 4,0, y $3{,}0 \leq 4{,}0$.

3.2. Ejemplo 2: Caso donde uno de los elementos no cumple con la condición

Considerando los siguientes conjuntos:

$$\texttt{cd1} = \{(2{,}5, 3), (3{,}5, 2), (4{,}0, 1)\}$$

$$\texttt{cd2} = \{(3{,}0, 1), (2{,}5, 2), (5{,}0, 3)\}$$

El resultado de ejecutar `inclusion(cd1, cd2)` sería `false`, porque:

- El elemento $(2{,}5, 3)$ está en `cd2` con un grado de 5,0, y $2{,}5 \leq 5{,}0$.
- El elemento $(3{,}5, 2)$ está en `cd2` con un grado de 3,0, pero $3{,}5 \leq 3{,}0$ es falso.
- La función retorna `false` inmediatamente cuando encuentra el primer falso.

3.3. Ejemplo 3: Caso donde un elemento no existe en `cd2`

Considerando los siguientes conjuntos:

$$\texttt{cd1} = \{(1{,}5, 4), (2{,}5, 3)\}$$

$$\texttt{cd2} = \{(1{,}5, 4), (3{,}0, 5)\}$$

El resultado de ejecutar `inclusion(cd1, cd2)` sería `false`, porque:

- El elemento $(1{,}5, 4)$ está en `cd2` con un grado de 4,0, y $1{,}5 \leq 4{,}0$.
- El elemento $(2{,}5, 3)$ no está en `cd2`, por lo que la función retorna `false` inmediatamente.