

INFORME TALLER 2: FUNCIONES DE ALTO ORDEN
FUNDAMENTOS DE PROGRAMACIÓN FUNCIONAL Y CONCURRENTE

GABRIEL URAZA GARCIA – 2359594
DIEGO FERNANDO LENIS DELGADO – 2359540
SEBASTIAN CERON OROZCO - 2266148
INGENIERIA DE SISTEMAS - 3743

CARLOS ANDRES DELGADO SAAVEDRA
DOCENTE

UNIVERSIDAD DEL VALLE

8/11/2024

TULUA VALLE

INFORME DE PROCESOS:

Pila de llamadas en la función grande:

```
java.lang.Exception: Stack trace
  at java.base/java.lang.Thread.dumpStack(Unknown Source)
  at taller.ConjuntosDifusos.$anonfun$grande$1(ConjuntosDifusos.scala:15)
  at taller.ConjuntosDifusos.auxiliar$1(ConjuntosDifusos.scala:44)
  at taller.ConjuntosDifusos.inclusion(ConjuntosDifusos.scala:48)
  at taller.ConjuntosDifusos.igualdad(ConjuntosDifusos.scala:52)
  at taller.App$.main(App.scala:36)
  at taller.App.main(App.scala)
```

En esta pila de llamada se puede ver que la función “grande” es llamada por la función igualdad (línea 52) a través de la función inclusión (línea 48)

```
java.lang.Exception: Stack trace
  at java.base/java.lang.Thread.dumpStack(Unknown Source)
  at taller.ConjuntosDifusos.$anonfun$grande$1(ConjuntosDifusos.scala:15)
  at taller.ConjuntosDifusos.$anonfun$union$1(ConjuntosDifusos.scala:30)
  at taller.App$.main(App.scala:38)
  at taller.App.main(App.scala)
```

En esta pila la función grande es llamada por la función anónima de la función unión (línea 30) y unión fue invocada desde el archivo App (línea 38)

Aquí grande fue invocado por la función unión



```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$grande$1(ConjuntosDifusos.scala:15)
    at taller.ConjuntosDifusos.$anonfun$interseccion$1(ConjuntosDifusos.scala:36)
    at taller.App$.main(App.scala:41)
    at taller.App.main(App.scala)
```

En esta pila la función grande fue llamada por la función intersección (línea 36) y luego intersección fue llamada desde App (línea 41)

Aquí la función grande en si es llamada por intersección

Pila de llamadas en la función complemento:



```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$complemento$1(ConjuntosDifusos.scala:24)
    at taller.App$.main(App.scala:21)
    at taller.App.main(App.scala)
```

En esta pila se ve como se llama a la función anónima de la función complemento (línea 24) esta función es llamada por el main del archivo App en la línea 21

```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$complemento$1(ConjuntosDifusos.scala:24)
    at taller.App$.main(App.scala:22)
    at taller.App.main(App.scala)
```

Aquí sucede lo mismo que en la anterior, solo que en el archivo app la función es llamada desde la línea 22

```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$complemento$1(ConjuntosDifusos.scala:24)
```

En esta última parte la función se esta ejecutando independientemente del main, solo se esta ejecutando en la clase de ConjuntosDifusos

Pila de llamadas en la función Unión:

```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$union$1(ConjuntosDifusos.scala:27)
    at taller.App$.main(App.scala:38)
    at taller.App.main(App.scala)
```

En esta pila se muestra la función anónima que hay dentro de unión (línea 27) y se llama a la función unión desde el main en la línea 38.

```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$union$1(ConjuntosDifusos.scala:27)
    at taller.App$.main(App.scala:39)
    at taller.App.main(App.scala)
```

Aquí sucede lo mismo solo que es llamada por otra línea del main, en este caso la línea 39

```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$union$1(ConjuntosDifusos.scala:27)
```

En la anterior es la ejecución por si sola de la función anónima dentro de unión

Pila de llamadas en la función intersección:

```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$interseccion$1(ConjuntosDifusos.scala:33)
    at taller.App$.main(App.scala:41)
    at taller.App.main(App.scala)
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$interseccion$1(ConjuntosDifusos.scala:33)
    at taller.App$.main(App.scala:42)
    at taller.App.main(App.scala)
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.$anonfun$interseccion$1(ConjuntosDifusos.scala:33)
```

En este caso ocurre lo que ha ocurrido en las demás funciones, se invoca a la función anónima que está dentro de la función intersección (Línea 33) y se llama desde el main en distintas líneas, en este caso desde la línea 41 y 42, y luego se invoca a la función anónima dentro de la misma intersección

Pila de llamadas en la función Inclusión:

```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.auxiliar$1(ConjuntosDifusos.scala:39)
    at taller.ConjuntosDifusos.inclusion(ConjuntosDifusos.scala:46)
    at taller.ConjuntosDifusos.igualdad(ConjuntosDifusos.scala:50)
    at taller.App$.main(App.scala:36)
    at taller.App.main(App.scala)
```

En esta pila se muestra como inclusión es llamado desde la función auxiliar, al mismo tiempo inclusión es llamado por igualdad, entonces inclusión forma parte de un conjunto de operaciones que evalúan la igualdad de conjuntos en el contexto de conjuntos difusos.

Cabe resaltar que esta pila se llama muchísimas veces, esto simplemente es un fragmento de la pila completa, ya que es el mismo fragmento repetido muchas veces, debido a la recursión de las funciones.

Pilas de llamada en igualdad:

```
java.lang.Exception: Stack trace
    at java.base/java.lang.Thread.dumpStack(Unknown Source)
    at taller.ConjuntosDifusos.igualdad(ConjuntosDifusos.scala:49)
```

Aquí simplemente se llama a la función dentro de conjuntos difusos.

Output general del código:

```
Grado de pertenencia de 2: 0.4444444444444444
Grado de pertenencia de 12: 0.7346938775510203
Grado de pertenencia de 22: 0.81
Complemento del grado de pertenencia de 2: 0.5555555555555556
Complemento del grado de pertenencia de 12: 0.26530612244897966
Complemento del grado de pertenencia de 22: 0.18999999999999995
Grado de pertenencia de 2 en la Union: 0.2962962962962962
Grado de pertenencia de 12 en la Union: 0.629737609329446
Grado de pertenencia de 22 en la Union: 0.7290000000000001
Grado de pertenencia de 2 en la Interseccion: 0.125
Grado de pertenencia de 12 en la Interseccion: 0.421875
Grado de pertenencia de 22 en la Interseccion: 0.5477084898572503
El conjunto grande 1 esta incluido en el conjunto grande 2?: true
El conjunto 1 es igual al 2?: false
```

La ejecución se hizo evaluando los valores (2), (6) y (9).

En general, las pilas de llamada mas extensas fueron las de la función grande y la función inclusión, debido a la recursividad de estas, la pila mostrada en cada una de estas es solo una parte de la pila de llamadas debido a que es tan extensa.

PRUEBAS DE SOFTWARE:

Ejecución de procesos generados por cada programa

PROCESOS función grande:

ejecucion paso a paso

```
{6 / (6 + 1)}^2
{6 / (7)}^2
36.0/49.0
Grado de pertenencia de 6: 0.7346938775510203
* Terminal will be reused by tasks, press any key to continue
```

funcion

```
13 def grande(d: Int, e: Int) : ConjDifuso = {
14   (n: Int) => {
15     pow((n.toDouble/(n+d).toDouble),e.toDouble)
16   }
17 }
18
```

PROCESOS función complmento:

ejecucion paso a paso

```

1 - {taller.ConjuntosDifusos$$Lambda/0x000002792c001800@35bbe5e8(6)}
{6 / (6 + 1)}^2
{6 / (7)}^2
36.0/49.0
1- 0.7346938775510203
{6 / (6 + 1)}^2
{6 / (7)}^2
36.0/49.0
Complemento del grado de pertenencia de 6: 0.26530612244897966
* Terminal will be reused by tasks, press any key to close it.

```

funcion

```

def complemento(c: ConjDifuso) : ConjDifuso = {
  (n : Int) => {
    | 1 - c(n)
  }
}

```

PROCESOS función union:

ejecucion paso a paso

```
{6 / (6 + 2)}^3
{6 / (8)}^3
216.0/512.0
{6 / (6 + 1)}^3
{6 / (7)}^3
216.0/343.0
Pertenencia de elemento 6 a AuB0.629737609329446
* Terminal will be reused by tasks, press any key to close it.
```

funcion

```
def union(cd1: ConjDifuso, cd2: ConjDifuso): ConjDifuso = {
  (n : Int) => {
    math.max(cd1(n), cd2(n))
  }
}
```

PROCESOS función intersección:

ejecucion paso a paso

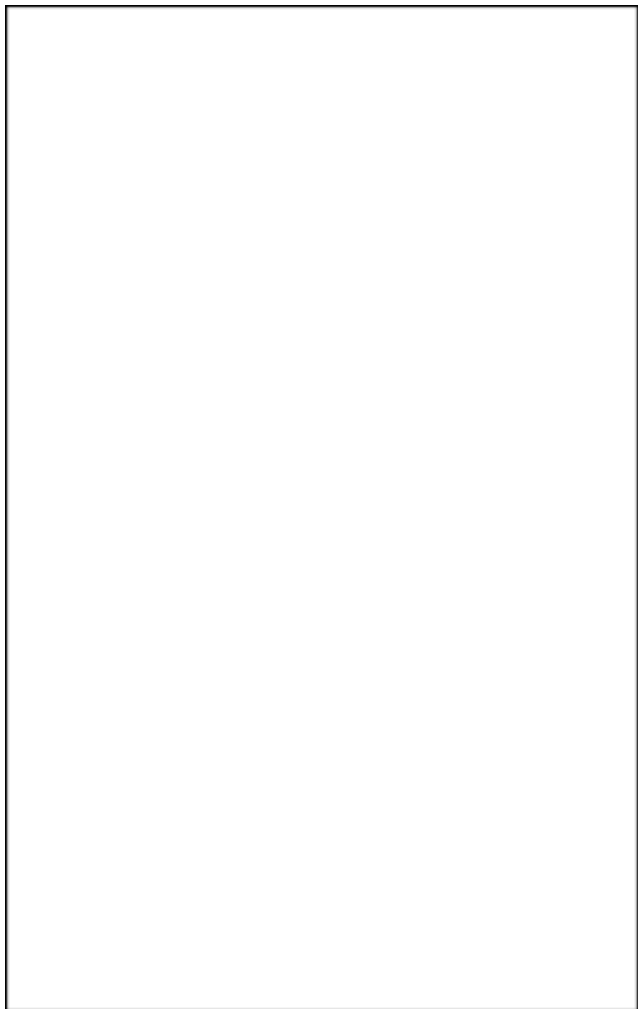
```
{6 / (6 + 2)}^3
{6 / (8)}^3
216.0/512.0
{6 / (6 + 1)}^3
{6 / (7)}^3
216.0/343.0
Pertenencia de elemento 6 a AnB0.421875
* Terminal will be reused by tasks, press any key to close it.
```


funcion

```
def interseccion(cd1: ConjDifuso, cd2: ConjDifuso): ConjDifuso = {
  (n: Int) => {
    math.min(cd1(n), cd2(n))
  }
}
```

PROCESOS función inclusion:

ejecucion paso a paso - ejecutada hasta n = 10



0	4	8
n > 10	n > 10	n > 10
false	false	false
$\{0 / (0 + 2)\}^3$	$\{4 / (4 + 2)\}^3$	$\{8 / (8 + 2)\}^3$
$\{0 / (2)\}^3$	$\{4 / (6)\}^3$	$\{8 / (10)\}^3$
0.0/8.0	64.0/216.0	512.0/1000.0
$\{0 / (0 + 1)\}^3$	$\{4 / (4 + 1)\}^3$	$\{8 / (8 + 1)\}^3$
$\{0 / (1)\}^3$	$\{4 / (5)\}^3$	$\{8 / (9)\}^3$
0.0/1.0	64.0/125.0	512.0/729.0
1	5	9
n > 10	n > 10	n > 10
false	false	false
$\{1 / (1 + 2)\}^3$	$\{5 / (5 + 2)\}^3$	$\{9 / (9 + 2)\}^3$
$\{1 / (3)\}^3$	$\{5 / (7)\}^3$	$\{9 / (11)\}^3$
1.0/27.0	125.0/343.0	729.0/1331.0
$\{1 / (1 + 1)\}^3$	$\{5 / (5 + 1)\}^3$	$\{9 / (9 + 1)\}^3$
$\{1 / (2)\}^3$	$\{5 / (6)\}^3$	$\{9 / (10)\}^3$
1.0/8.0	125.0/216.0	729.0/1000.0
2	6	10
n > 10	n > 10	n > 10
false	false	false
$\{2 / (2 + 2)\}^3$	$\{6 / (6 + 2)\}^3$	$\{10 / (10 + 2)\}^3$
$\{2 / (4)\}^3$	$\{6 / (8)\}^3$	$\{10 / (12)\}^3$
8.0/64.0	216.0/512.0	1000.0/1728.0
$\{2 / (2 + 1)\}^3$	$\{6 / (6 + 1)\}^3$	$\{10 / (10 + 1)\}^3$
$\{2 / (3)\}^3$	$\{6 / (7)\}^3$	$\{10 / (11)\}^3$
8.0/27.0	216.0/343.0	1000.0/1331.0
3	7	11
n > 10	n > 10	n > 10
false	false	true
$\{3 / (3 + 2)\}^3$	$\{7 / (7 + 2)\}^3$	 Terminal will
$\{3 / (5)\}^3$	$\{7 / (9)\}^3$	
27.0/125.0	343.0/729.0	
$\{3 / (3 + 1)\}^3$	$\{7 / (7 + 1)\}^3$	
$\{3 / (4)\}^3$	$\{7 / (8)\}^3$	
27.0/64.0	343.0/512.0	

funcion

```
def inclusion(cd1: ConjDifuso, cd2: ConjDifuso) : Boolean = {  
  @tailrec  
  def auxiliar(n: Int): Boolean = {  
    if (n > 1000) true  
    else{  
      if (cd1(n) > cd2(n)) false  
      else auxiliar(n + 1)  
    }  
  }  
  auxiliar(0)  
}
```

PROCESOS función igual:

ejecucion paso a paso - ejecutada hasta n = 10

```
0
n > 10
false
{0 / (0 + 1)}^3
{0 / (1)}^3
0.0/1.0
{0 / (0 + 2)}^3
{0 / (2)}^3
0.0/8.0
1
n > 10
false
{1 / (1 + 1)}^3
{1 / (2)}^3
1.0/8.0
{1 / (1 + 2)}^3
{1 / (3)}^3
1.0/27.0
A > B
* Terminal will be reused
```

la ejecucion inicialmente es igual a inclusion A en B

Posteriormente B en A, para el valor 0 ambas son iguales, 0, a partir de n1, son diferentes y no se cumple esta segunda condicion -> son diferentes

CODIGO

```
def igualdad(cd1: ConjDifuso, cd2: ConjDifuso) : Boolean = {
  inclusion(cd1, cd2) && inclusion(cd2, cd1)
}
```

INFORME DE CORRECCIÓN:

Argumentacion correccions

funcion pertenece:

La funcion pertenece exclusivamente espera a recibir parametros para la funcion característica, un elemento a evaluar para una expresion de tipo conjunto difuso, que dependiendo de la expresion determinara el grado de pertenencia

funcion grande


```
(n: Int) => {  
    pow((n.toDouble/(n+d).toDouble),e.toDouble)  
}
```

recibe 2 enteros para la funcion característica, que es un cociente impropio en el que el denominador es +d unidades mayor que el numerador, y toda la expresion esta elevada a la e-sima potencia, lo que da mayor precision para numeros de mayor tamaño

si $a > b \rightarrow a/b > 1$, $b > a \rightarrow a/b < 1$

para aplicar otras funciones es necesario determinar un metodo para crear un coeficiente que tambien sea una fraccion impropia

funcion complemento

dado que la pertenencia a un conjunto difuso es un decimal menor a 1 que entre mas se aproxime a 1 mas perteneces, al sustraer el grado de pertenencia de un elemento a un conjunto a 1 ($1-f$) se obtiene un valor alejado de 1, que corresponde con cuanto pertenece al conjunto complemento.

ej:

n pertenece a A = 0,4

este tiene baja pertenencia, por lo que debe pertenecer mas al complemento, lo cual se verifica con

$$1 - 0,4 = 0,6$$

funcion union,

en esta se realiza la comparación para 2 funciones características y se toma el valor maximo para determinar que elementos estan dentro de la union, dado que los elementos que pertenezcan a ambos deben tener el mayor grado de pertenencia

funcion interseccion

al contrario de la funcion anterior, los pertenencientes a este conjunto son los valores menos afines a cualquiera de los dos conjuntos, por tanto los elementos comunes son aquellos que pertenecen en menor grado a cada conjunto individualmente

funcion inclusion

```
def inclusion(cd1: ConjDifuso, cd2: ConjDifuso) : Boolean = {  
  @tailrec  
  def auxiliar(n: Int): Boolean = {  
    if (n > 1000) true  
    else {  
      if (cd1(n) > cd2(n)) false  
      else auxiliar(n + 1)  
    }  
  }  
  auxiliar(0)  
}
```

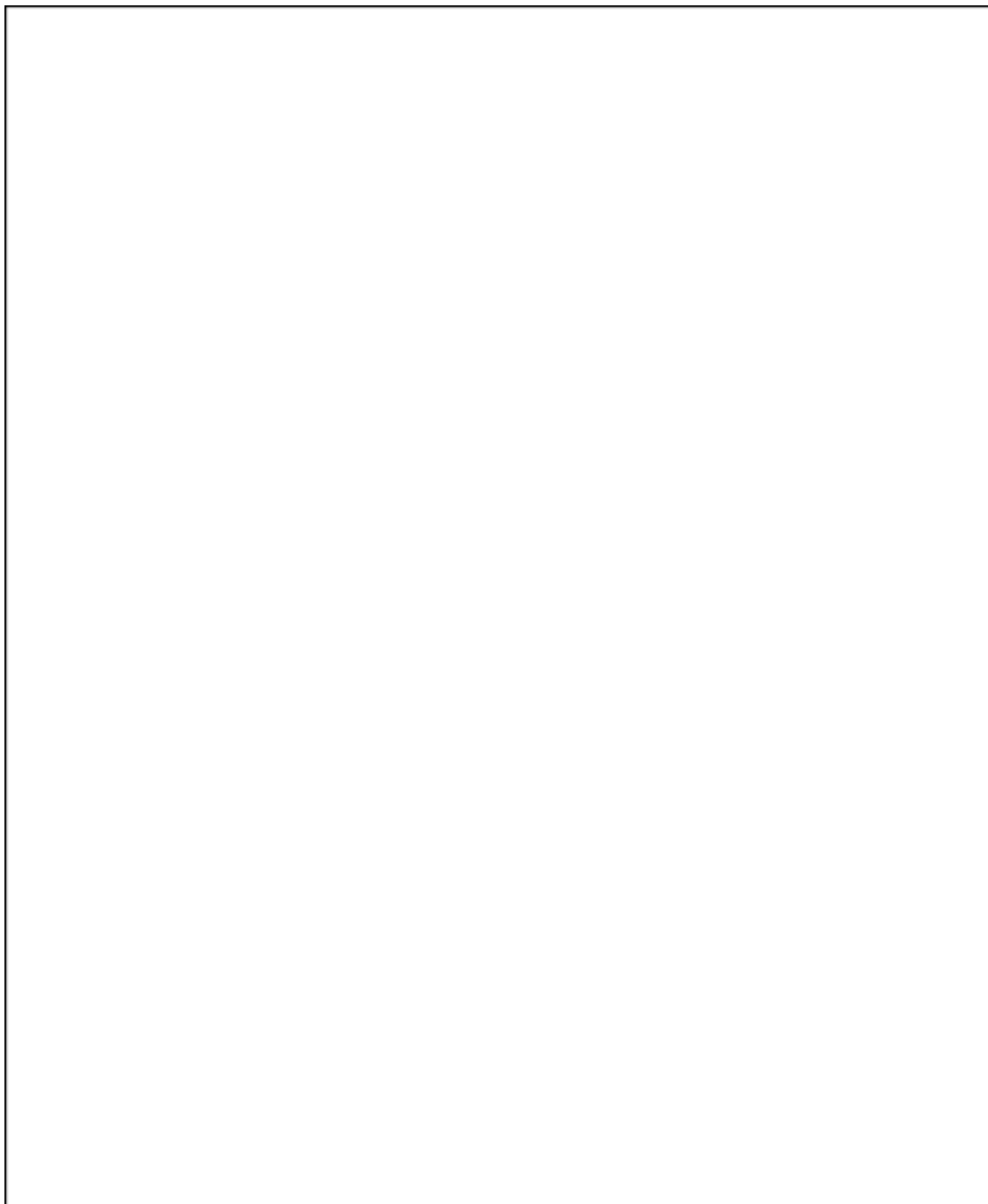
esta funcion implementa recursion de cola para evaluar cada elemento desde $n = 0$ hasta 1000, si el grado de pertenencia para el primer conjunto es mayor para el otro, esto quiere decir que hay un elemento al menos que no esta contenido, al exceder el grado de pertenencia para el otro conjunto, si esto sucede se interrumpe el reto de evaluaciones y se arroja false, de lo contrario se sigue evaluando.

correcciones -> se puede esperar que un valor bastante grande de n arroje que el conjunto 1 no esta contenido e 2, sin embargo para funciones como las empleadas en el ejercicio se puede inferir que no hay forma de que ante el crecimiento exponencial del denominador, un conjunto con un parametro b menor al conjunto que se verifica si esta contenido en el arroje falsea

funcion igualdad

esta funcion evalua que ambas expresiones tanto a contenido b como b contenido a se cumplan, ya que de ser iguales se cumplira que un conjunto sea subconjunto de si mismo

Casos de prueba - adjunto en ConjuntosDifusosTest.scala





CONCLUSIONES:

Los conjuntos difusos son un ejemplo de aplicación de funciones definidas a trozos, pueden ser útiles para tareas de categorización, ya que se puede establecer un umbral a partir del cual el objeto debe tomar cierto comportamiento.

Este tipo de conjuntos permite