Inteligencia Artificial para Videojuegos – Capítulo 4

Introducción

En esta unidad veremos una técnica de Inteligencia Artificial conocida como Fuzzy Logic o Lógica Difusa. La misma nos permitirá crear comportamientos y tomar decisiones en nuestro videojuego a partir de reglas expresadas en un lenguaje coloquial. Gracias a esto nuestros personajes podrán comportarse de forma mucho más creíble y "humana". Para ello primero haremos un repaso de la lógica convencional o de Bool para luego introducir los conceptos de conjuntos difusos. Finalmente vamos a explicar cómo funciona el sistema de inferencia Fuzzy y mostraremos un ejemplo de uso del mismo.

Lógica Difusa

¿Porqué?

Alguna vez les pasó que estaban programando el comportamiento de un personaje en un juego y quien les daba las indicaciones de como se debería comportar éste les dio instrucciones como:

a. Si esta **MUY RODEADO** de enemigos **PODEROSOS** y tiene **POCAS VIDAS** -> **HUIR RAPIDO**.
b. Si esta **MUY RODEADO** de enemigos **DEBILES** y tiene **POCAS VIDAS** -> **HUIR LENTO**.
c. Si esta **POCO RODEADO** de enemigos **DEBILES** y tiene **MUCHAS VIDAS** -> **ATACAR RAPIDO**.

Esas instrucciones hacen mucho sentido para un humano… pero como le explicamos, nosotros los programadores, al personaje que es estar **MUY RODEADO** o uno **POCO RODEADO**?
Evidentemente no es sencillo comunicarle a un programa conceptos tan imprecisos o difusos ya que la lógica que entiende una computadora es la lógica convencional, donde un personaje o esta rodeado o no lo esta, pero no puede estar algo rodeado. Sin embargo seria conveniente poder expresarle a la computadora reglas como las que vimos arriba por varios motivos:

- Al crear una inteligencia artificial que controle un elemento de un juego muy probablemente entrevistemos a alguien que sepa controlar ese elemento bien, por ejemplo si estamos programando un bot para un juego como el Quake seguramente hablaríamos con un jugador experto (en realidad en la practica muchas veces terminaremos siendo nosotros mismos) y le preguntaremos en cada situación que es lo que hace, a lo que seguramente nos contestara con términos difusos como si un enemigo esta MUY LEJOS, LEJOS, MUY CERCA, CERCA.
- Nuestras inteligencias artificiales se comportaran de forma más humana, porque para ellas las situaciones no se evaluaran de forma absoluta, por ejemplo diciendo si esta cerca o no lo esta, sino que para ella un enemigo podrá estar algo cerca.

Imaginemos que queremos programar como ya sabemos hacerlo si un enemigo esta a distancia **CERCANA**, **MEDIA** o **LEJANA**. Lo que haríamos (sin conocer el tema de esta unidad) es separar en rangos, por ejemplo si el enemigo esta entre 0 y 20 unidades esta a distancia **CERCANA**; si el enemigo esta entre 20 y 40 unidades esta a distancia **MEDIA**; si el enemigo esta entre 40 y 60 unidades esta a distancia **LEJANA**. En fin resumiendo:

$$distancia \begin{cases} 0 - 20\ unidades\ \boldsymbol{CERCANA} \\ 20 - 40\ unidades\ \boldsymbol{MEDIA} \\ 40 - 60\ unidades\ \boldsymbol{LEJANA} \end{cases}$$

Pero si el enemigo esta a 19.99 unidades según los rangos que definimos arriba el enemigo estaría a distancia CERCANA de forma absoluta a pesar de que a 0.1 unidades mas estaría a distancia MEDIA. Es decir que a pesar de que el enemigo estuviese a 2.0 unidades de distancia o 19.99 unidades para nuestro personaje será lo mismo porque a ambas distancias esta a distancia

CERCANA. Claro podría refinar mas el problema creando más intervalos, por ejemplo una nueva categoría entre $10 - 20$ unidades. Pero el problema de fondo sigue estando, y es que el enemigo se considera que esta de forma absoluta dentro de una de las categorías que definimos.

En cambio pensemos como razonamos los humanos. Cuando consideramos los términos lingüísticos como "cerca" y "lejos" o "suavemente" y "fuertemente" pensamos que los límites entre estas categorías son vagos, es decir por ejemplo podemos pensar que entre 20-30 metros comienza a dejar de ser cerca y comienza a ser lejos de forma gradual. Esto es porque los humanos consideramos que el valor tiene un ***grado de pertenencia*** a cada conjunto. Entonces en nuestro ejemplo anterior a 19.99 unidades de distancia en nuestra mente diríamos que esta asociada a distancia CERCANA pero en mayor medida a distancia MEDIA. De esto se trata esta unidad, una lógica donde un valor no pertenece de forma absoluta a una categoría sino que en general pertenece a varios conjuntos.

Lógica Difusa…

Entonces la lógica difusa o también llamada borrosa permite que una computadora pueda razonar términos lingüísticos como LEJOS o CERCA (siempre que pongamos un termino lingüístico será en mayúsculas) y reglas lingüísticas de forma similar a como lo hace un humano.

Para lograr esto estudiaremos un proceso llamado inferencia basado en reglas difusas o sistema difuso. El esquema de este es:

ILUSTRACIÓN 1: ESQUEMA SIMPLIFICADO DE SISTEMA DE DIFUSO

Vemos a grandes rasgos que es cada elemento:

- V: **El valor de entrada al sistema. Desde ahora nos referiremos a él como valor nítido, solo para remarcar que es un valor normal.**
- $Fuzzificación$: **En este proceso se calcula que tanto pertenece el valor de entrada V a cada conjunto.**
- $Regla\ difusa$: **Se evalúan las reglas difusas con el valor fuzzificado de entrada**
- $Defuzzificación$: **En este proceso se calcula que valor de salida se tendrá**
- V': **El valor de salida.**

Conjuntos nítidos

En la teoría de conjuntos clásica que se enseña en la escuela, los elementos del espacio de discurso o Universo (es decir todo el dominio) pertenecen a un conjunto o no pertenecen. Es decir que la pertenencia a un conjunto se da de forma booleana, true pertenece al conjunto, false no pertenece. Desde ahora en más nos referiremos a esta teoría de conjuntos como conjuntos boleanos. Entonces tenemos la siguiente certeza:

Los elementos pueden pertenecer a un conjunto nítido en un 100% o en un 0%, pero jamás de forma parcial.

En la Ilustración 2 podemos ver un ejemplo donde el espacio de discurso son los números enteros entre 1 y 15, y los conjuntos son de números pares (conjunto marrón) y números impares (conjunto verde). En el ejemplo podemos ver que el elemento 3 pertenece un 100% al conjunto de números impares.

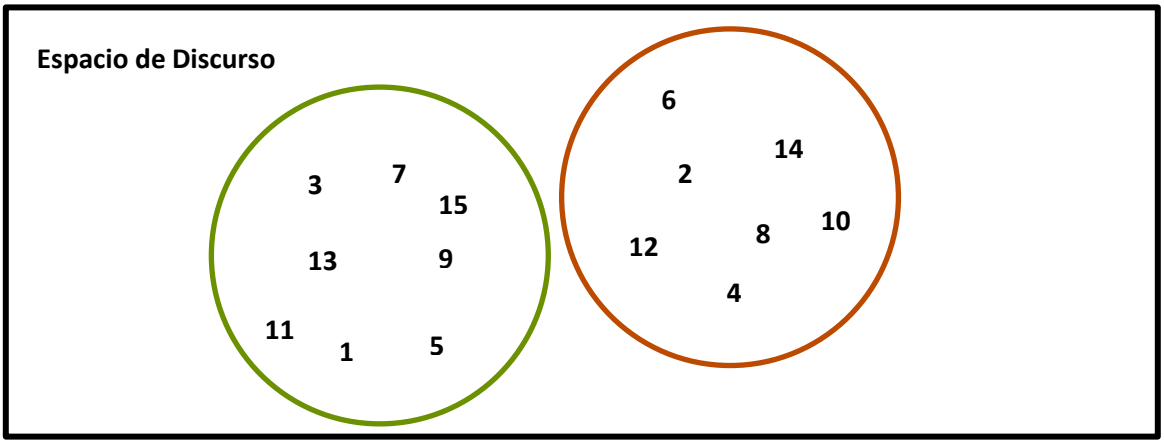

ILUSTRACIÓN 2: EJEMPLO DE CONJUNTOS BOOLEANOS

Operaciones entre conjuntos nítidos

Recordemos las operaciones que podíamos realizar entre conjuntos booleanos para que luego las operaciones análogas en conjuntos difusos tengan sentido. Las operaciones básicas entre conjuntos son:

1. Unión:

   La operación unión entre dos conjuntos nítidos nos da como resultado un conjunto nítido que contiene todos los elementos de los dos conjuntos que operan. Por ejemplo si tenemos dos conjuntos $A\{1,4,5,6\}$ y $B\{1,2\}$, la unión entre A y B que se representa con el símbolo ∪ nos devuelve el conjunto $\{1,2,4,5,6\}$. Resumiendo:

   $$A \cup B = \{1,2,4,5,6\}$$

2. Intersección:

   La operación intersección entre dos conjuntos nítidos nos da como resultado un conjunto nítido que contiene los elementos que están en los dos conjuntos. Por ejemplo si tenemos dos conjuntos $A\{1,4,5,6\}$ y $B\{1,2\}$, la intersección entre A y B que se representa con el símbolo ∩ nos devuelve el conjunto $\{1\}$. Resumiendo:

   $$A \cap B = \{1\}$$

3. Complemento:

   A diferencia de los dos operadores anteriores el complemento es una operación unaria, es decir que no se aplica a dos conjuntos sino que a uno solo. El complemento de un conjunto nítido nos da como resultado un conjunto nítido que contiene todos los elementos del espacio de discurso que no están en el conjunto sobre que se opera. Por ejemplo si tenemos un conjunto $A\{1,4,5,6\}$ y nuestro espacio de discurso es $U\{1,2,3,4,5,6,7,8,9,10\}$ es decir todos los enteros de 1 a 10, el complemento de A que se representa con el símbolo $\bar{A}$ nos devuelve el conjunto $\{2,3,7,8,9,10\}$. Resumiendo:

   $$\bar{A} = \{2,3,7,8,9,10\}$$

Conjuntos difusos

Los conjuntos difusos difieren de los conjuntos que vimos anteriormente en que los elementos ya no pertenecen o no pertenecen de forma absoluta a un conjunto, sino que tiene un grado de pertenencia al conjunto.

Los elementos pueden pertenecer a un conjunto difuso en un cierto porcentaje entre 0% y 100%, al que se le llamara grado de pertenencia del elemento al conjunto difuso, conocido como DOM por sus siglas en ingles.

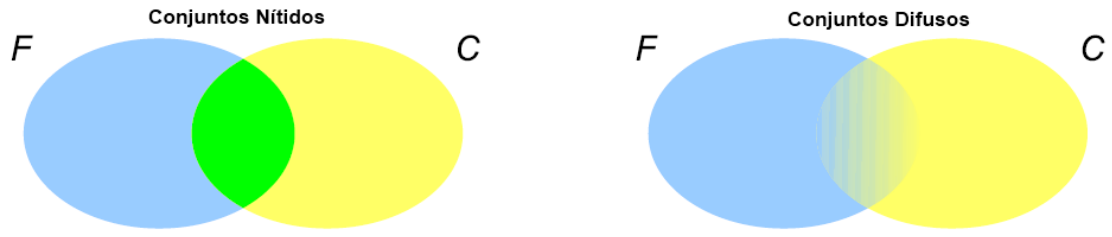

ILUSTRACIÓN 3: COMPARACIÓN ENTRE CONJUNTOS NÍTIDOS Y DIFUSOS

Por ejemplo veamos un ejemplo donde usaremos primero conjuntos nítidos para representar nuestro problema y luego conjuntos difusos. Supongamos que tenemos el coeficiente intelectual

(IQ) y queremos clasificar como TONTA, PROMEDIO o INTELIGENTE (claramente si terminan de leer el apunte y entienden algo están en este conjunto). Supongamos que entre 70 y 90 puntos consideremos que la persona es TONTA, si tiene entre 90 y 110 es PROMEDIO y si tiene entre 110 y 130 es INTELIGENTE. Entonces como estamos trabajando con conjuntos nítidos en esos rangos cada puntaje de IQ pertenece a solo un conjunto, por ejemplo 89 es un IQ que pertenece a TONTA, en cambio 91 es un IQ que pertenece a PROMEDIO. Ahora veamos como podemos graficar el grado de pertenencia a estos conjuntos de cada coeficiente intelectual, recordemos que dijimos que los conjuntos nítidos tenían una pertenencia absoluta o no pertenecían a un conjunto. El grado de pertenencia lo representaremos como un valor entre 0 y 1, donde 0 significa 0% de pertenencia y 1 representa 100% de pertenencia. Entonces podemos representar la pertenencia a los conjuntos nítidos como:

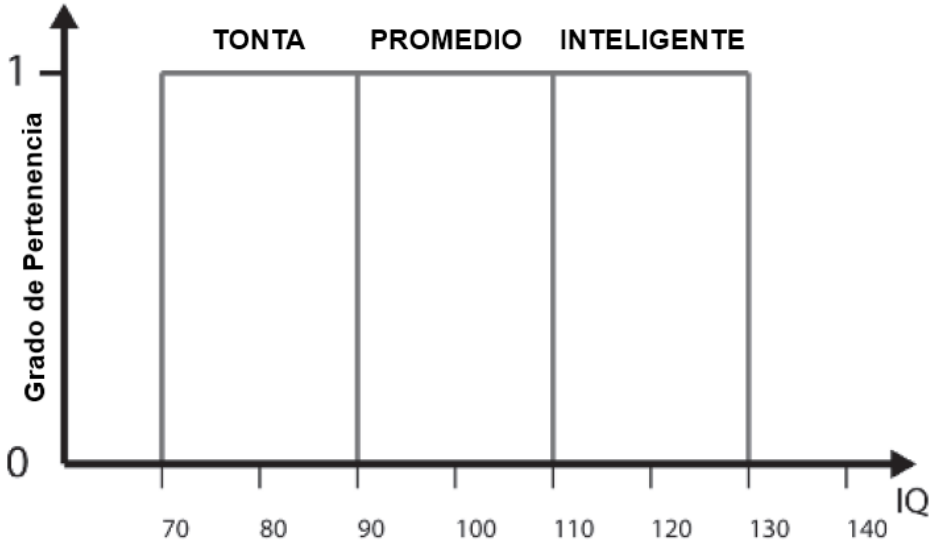

ILUSTRACIÓN 4: GRADO DE PERTENENCIA DEL EJEMPLO EN CONJUNTOS NÍTIDOS

Vemos que siempre en el grafico tendremos rectángulos como la forma de cada conjunto, por ejemplo el conjunto PROMEDIO es un rectángulo entre el valor 90 y el 110. Eso es lo que esperábamos ya que como dijimos cada IQ pertenece a un conjunto con un grado de pertenencia 1 o 0, pero jamás uno intermedio.

En cambio si modelamos el ejemplo con conjuntos difusos tenemos que definir la función de grado de pertenencia del IQ a cada conjunto del problema, que en este caso serán conjuntos difusos. Entonces para definir el conjunto difuso debemos definir una función de grado de pertenencia para cada conjunto, por ejemplo la función de pertenencia del conjunto TONTA es $P_{TONTA}(IQ)$ que para cada valor de IQ asigna un valor entre 0 y 1. De la misma forma para los demás conjuntos definimos las funciones $P_{PROMEDIO}(IQ)$ y $P_{INTELIGENTE}(IQ)$. Lo que deseamos es que en vez de terminar el grado de pertenencia de cada conjunto de forma abrupta como lo hacia en los conjuntos nítidos, el grado decaiga lentamente y el próximo conjunto incremente su grado de pertenencia, de esta forma un valor en el limite como por ejemplo el IQ 109 no esta solo en

PROMEDIO, sino que esta un poco en PRIMEDIO y un poco en INTELIGENTE. Una posible elección de las funciones de pertenencia seria:

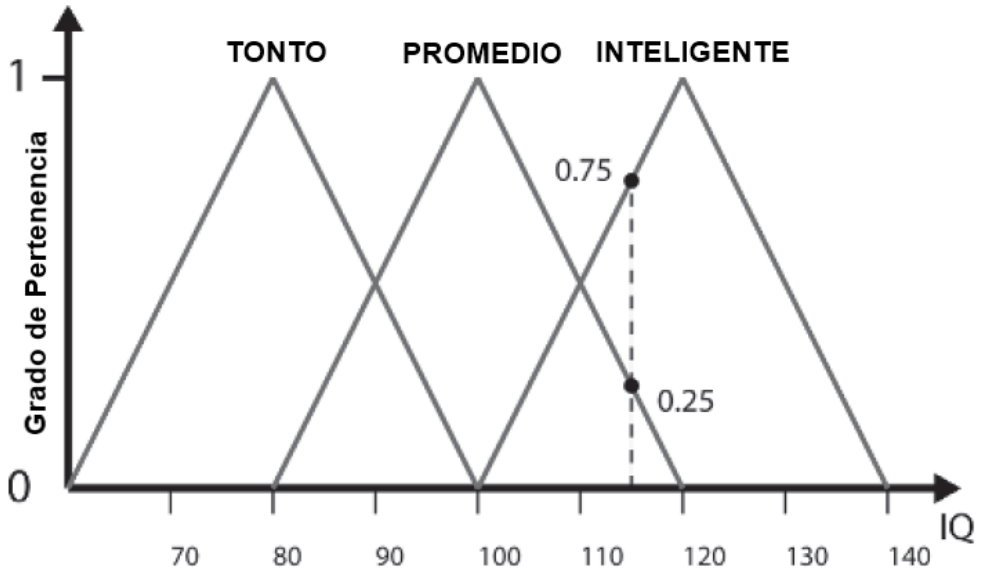

ILUSTRACIÓN 5: GRADO DE PERTENENCIA DEL EJEMPLO EN CONJUNTOS DIFUSOS

En la próxima sección veremos las funciones de pertenencia más comunes. Entonces como podemos ver en la Ilustración 5 un IQ de 115 tiene una pertenencia a los conjuntos difusos:

$P_{TONTO}(115) = 0.0$

$P_{PROMEDIO}(115) = 0.25$

$P_{INTELIGENTE}(115) = 0.75$

Vemos en el ejemplo que la suma del grado de pertenencia de todos los conjuntos difusos para cada IQ suma 1.0, eso no es coincidencia sino que es recomendable que lo hagan. Entonces:

En general la suma del grado de pertenencia de todos los conjuntos difusos para un valor suma 1.

Formas típicas de funciones de membresía

Las funciones de membresía pueden tener cualquier forma que deseemos siempre y cuando controlemos de que entre todas las funciones de membresía no nos pasemos de 1. Pero en general se suele definir funciones con formas simples como triangulares (como el ejemplo anterior) o trapezoidal. Estas formas son útiles ya que

representan lo que en general queremos para nuestros conjuntos difusos, que estos terminen de forma suave y paulatina. En la Ilustración 6 podemos ver varias funciones comunes.

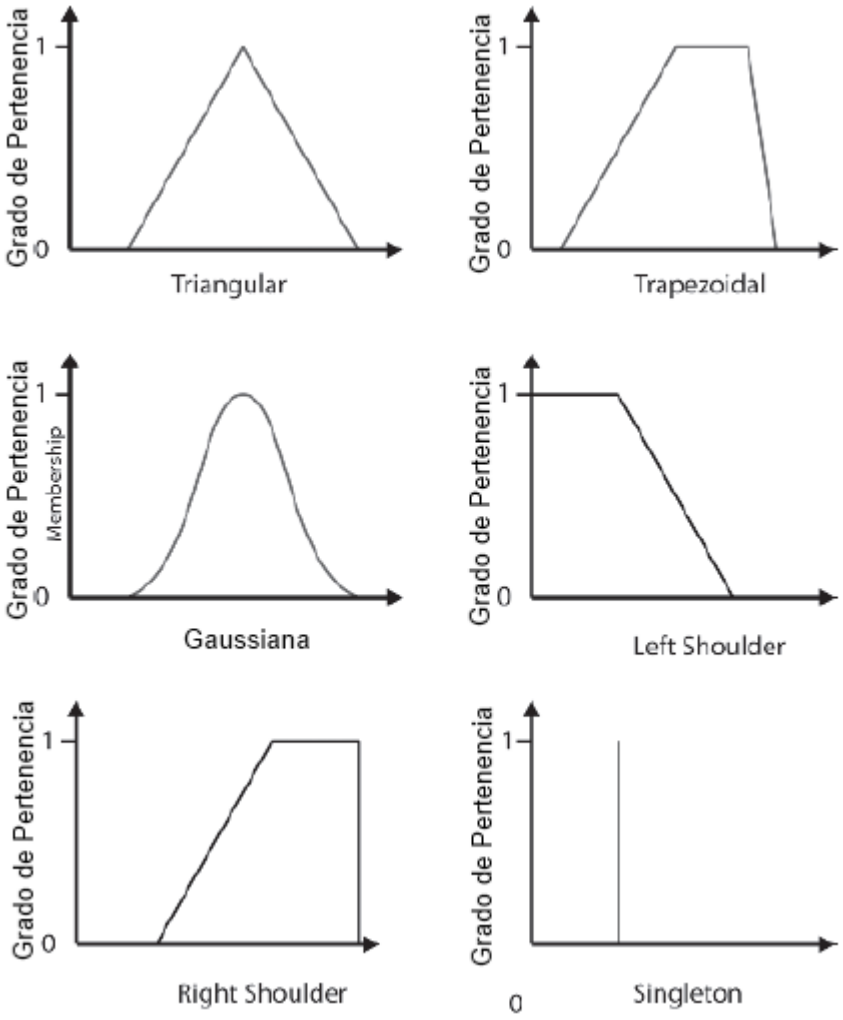

ILUSTRACIÓN 6: FUNCIONES DE PERTENENCIA COMUNES

La función Singleton en realidad no es difusa ya que su único elemento tiene una pertenencia total o ninguna, pero no obstante de ser un caso degenerado es útil en la practica.

Operaciones entre conjuntos difusos

Ahora que ya sabemos lo que es un conjunto difuso veamos como se adaptan las operaciones entre conjuntos que vimos para los conjuntos nítidos a estos conjuntos difusos:

1. Unión:

   Como vimos en los conjuntos nítidos la unión es idéntica a un OR entre los conjuntos. Acá se mantiene esto, pero en general un OR lo realizamos entre valores

boleanos, mientras que con conjuntos difusos tendremos pertenencias entre 0 y 1. Por esto definiremos a la operación OR entre valores difusos como:

$$PROMEDIO \cup INTELIGENTE = PROMEDIO\ OR\ INTELIGENTE$$
$$= max\{P_{PROMEDIO}(x), P_{INTELIGENTE}(x)\}$$

Es decir que tomaremos el máximo de pertenencia para cada valor de la variable, en este caso es el IQ. Podemos ver esto gráficamente en la Ilustración 7.

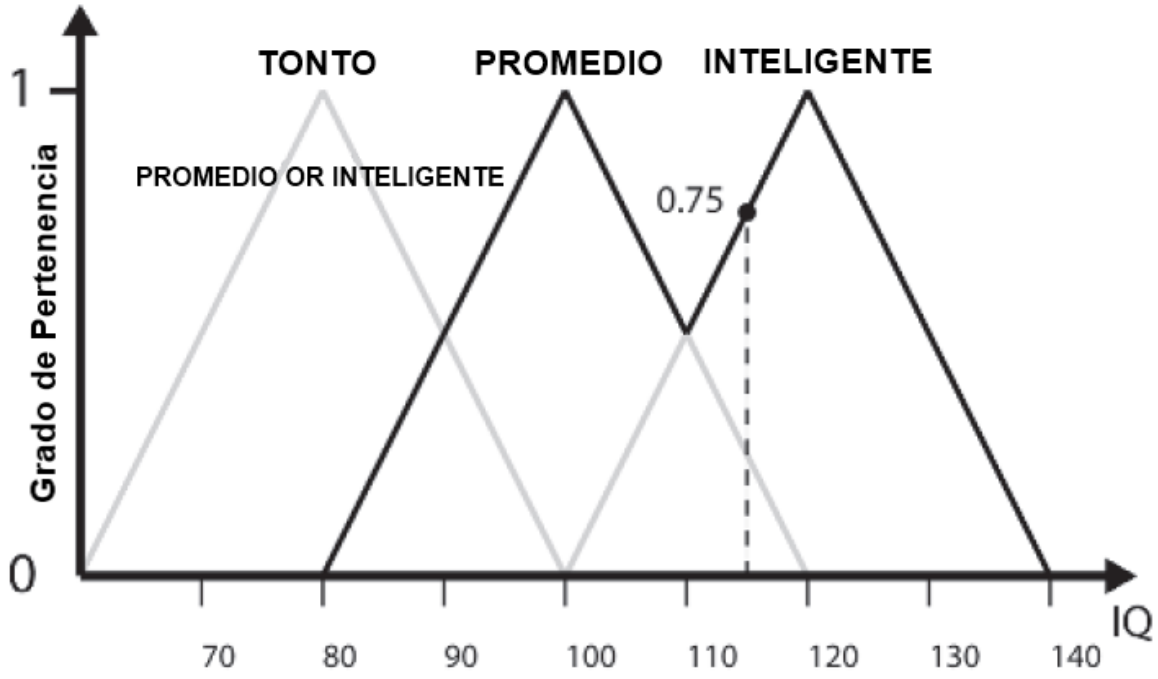

ILUSTRACIÓN 7: EJEMPLO OR DIFUSO

2. Intersección:

Como vimos en los conjuntos nítidos la intersección es idéntica a un AND entre los conjuntos. Por esto definiremos a la operación AND entre valores difusos como:

$$PROMEDIO \cap INTELIGENTE = PROMEDIO\ AND\ INTELIGENTE$$
$$= min\{P_{PROMEDIO}(x), P_{INTELIGENTE}(x)\}$$

Es decir que tomaremos el mínimo de pertenencia para cada valor de la variable, en este caso es el IQ. Podemos ver esto gráficamente en la Ilustración 8 .

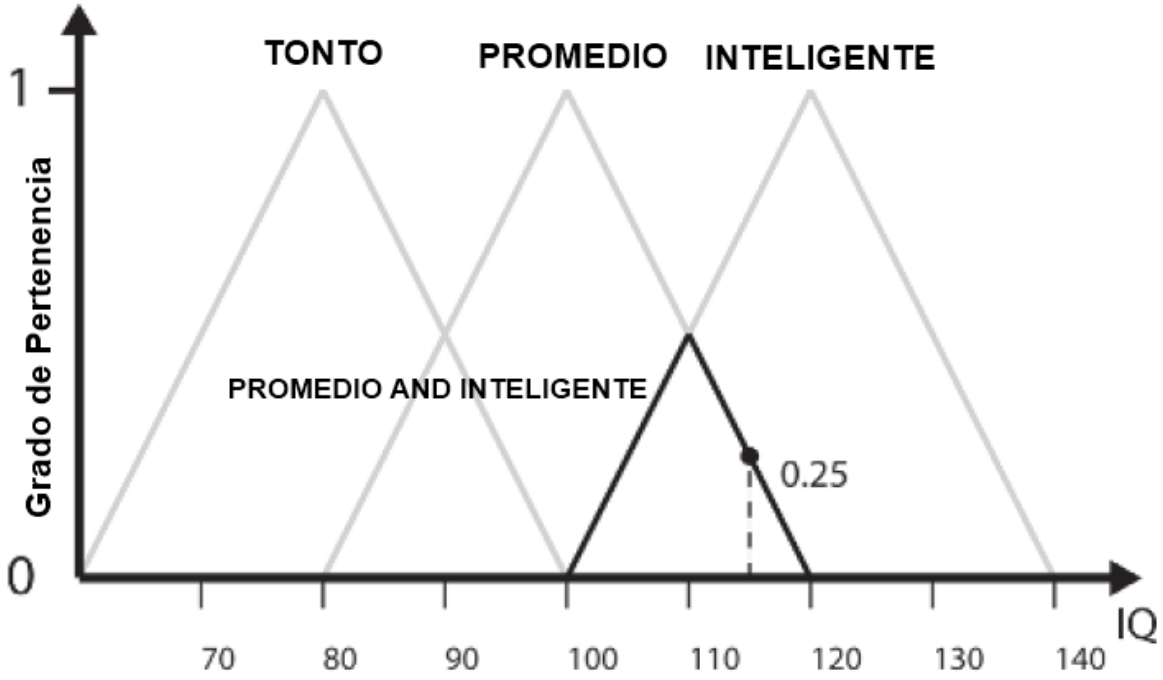

ILUSTRACIÓN 8: EJEMPLO AND DIFUSO

3. Complemento:

   El complemento como vimos equivale a el NOT lógico. Para adaptar la operación a conjuntos difusos haremos:

$$\overline{PROMEDIO} = NOT\ PROMEDIO = 1 - P_{PROMEDIO}(x)$$

   Es decir que tomaremos el complemento a 1 de la pertenencia para cada valor de la variable, en este caso es el IQ. Podemos ver esto gráficamente en la Ilustración 9.

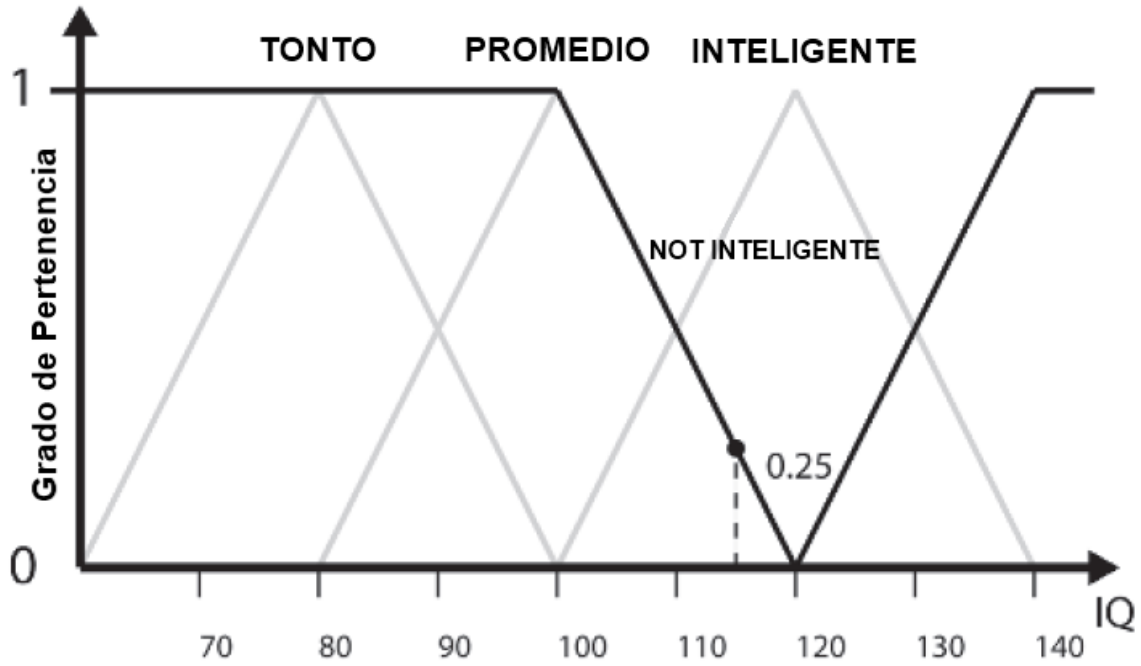

ILUSTRACIÓN 9: EJEMPLO NOT DIFUSO

Variables Lingüísticas Difusas o Fuzzy Linguistic Variables (FLV)

Hasta ahora hablamos de conjuntos difusos separados, pero grupos de conjuntos difusos están relacionados de cierta forma como por ejemplo los conjuntos TONTO, PROMEDIO e INTELIGENTE. Por otro lado otros conjuntos difusos como CERCANO y LEJANO no tienen nada que ver con los conjuntos anteriores. Para agrupar estos conjuntos difusos definiremos las FLV como sigue:

> ***Una FLV es un grupo de uno o más conjuntos difusos que representan un mismo concepto y tienen un mismo dominio.***

Por ejemplo en el ejemplo del coeficiente intelectual podemos decir que los tres conjuntos difusos pertenecen a una misma FLV que podemos llamar $\boldsymbol{IQ}$ (la escribiremos con letra negrita). Esto es correcto según la definición ya que entre los tres representan un mismo conjunto que es el coeficiente intelectual de una persona y además tienen un mismo dominio que es el valor del coeficiente intelectual. Entonces la FLV de ese ejemplo será:

$$\boldsymbol{IQ} = \{TONTO, PROMEDIO, INTELIGENTE\}$$

Es importante saber que en una misma FLV podemos mezclar conjuntos difusos con diferentes funciones de pertenencia, siempre y cuando tengamos cuidado de que ningún valor del dominio tenga una pertenencia de mas de 1 entre todos los conjuntos difusos. Por ejemplo podemos definir un FLV que modele la dirección a la que se avanza teniendo como dominio el grado de la dirección como:

$$\boldsymbol{DIRECION} = \{MUY_{IZQUIERDA}, IZQUIEDA, CENTRO, DERECHA, MUY\_DERECHA\}$$

Y el grafico de funciones de pertenencia de la FLV podría ser:

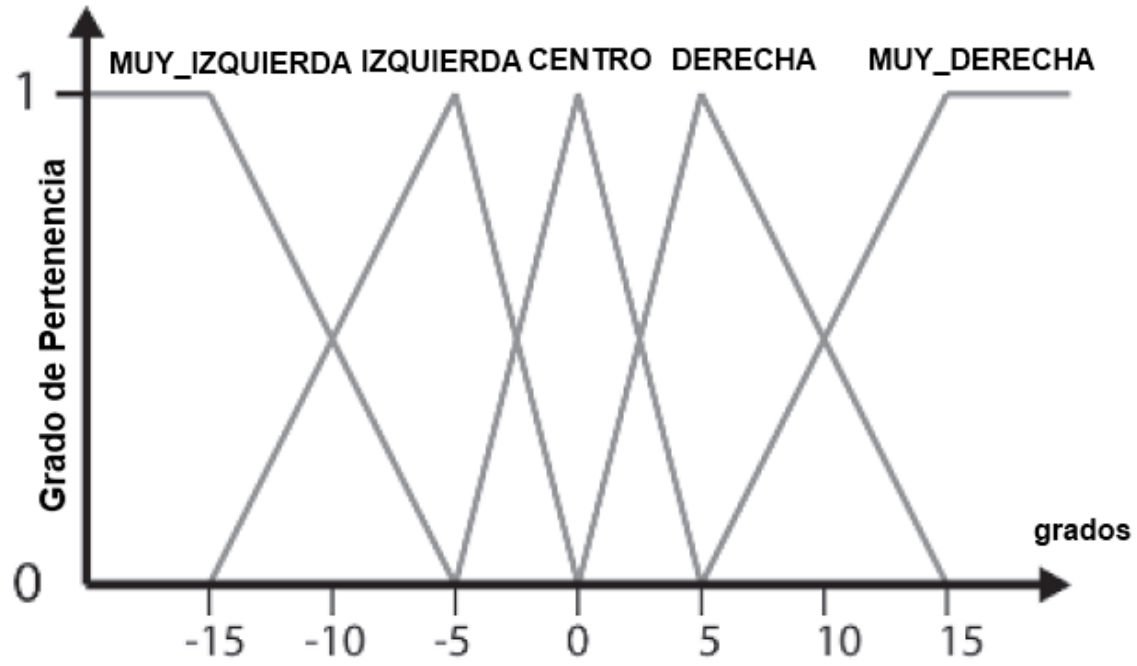

ILUSTRACIÓN 10: EJEMPLO FLV *DIRECCION*

Reglas Difusas

Hasta ahora hablamos mucho sobre los conjuntos difusos, sus operaciones y como se agrupan en FLV… pero todavía no sabemos como usar esos conceptos para lograr una lógica. Acá es donde hacen su entrada triunfal las reglas difusas. Una regla difusa esta compuesta por dos partes: un antecedente y un consecuente de la siguiente forma:

IF *antecedente* THEN *consecuente*

Como seguramente ya imaginaran el antecedente representa la condición bajo la cual se desencadena el consecuente, es decir si sucede el antecedente luego se da el consecuente. Pero acá no hay nada de nuevo, hemos estado planteando condiciones de estas todo el tiempo. La diferencia en las reglas difusas es que a diferencia de las reglas lógicas tradicionales donde el

consecuente se activa o no activa es que en las reglas difusas el consecuente se activa en cierto grado dependiendo de la activación del antecedente.

Entonces el antecedente estará compuesto por un conjunto difuso o por una combinación de conjuntos difusos que operan entre sí. El grado de activación del consecuente se dará por la activación que tenga este antecedente luego de las operaciones. Un sistema difuso estará compuesto por varias de estas reglas, que serán tantas como tan compleja sea nuestra lógica, la cantidad de FLV que tengamos y de cuantos conjuntos difusos tengan estas. Por ejemplo las reglas difusas para el ejemplo que dimos al comenzar la unidad de un personaje que necesita saber cuando huir y cuando atacar a sus enemigos pueden expresarse como:

IF MUY_RODEADO AND PELIGROSOS AND POCAS_VIDAS THEN HUIR_RAPIDO

IF MUY_RODEADO AND DEBILES AND POCAS_VIDAS THEN HUIR_LENTO

IF POCO_RODEADO AND DEBILES AND MUCHAS_VIDAS THEN ATACAR_RAPIDO

Veamos además que ahí tenemos varias FLV, una para la medida de cuan rodeado esta el personaje que seguramente su dominio será sobre la cantidad de enemigos en la cercanía, otra FLV será que tan fuertes son los enemigos que lo rodean que podría ser por ejemplo el promedio de vida que tienen esos enemigos, y otra FLV para las vidas del personaje que tendrá como dominio los puntos de vida del personaje. Además se tendrá dos FLV para los conjuntos difusos que se activan en las reglas donde una FLV es para saber que tan rápido huir donde el dominio podría ser simplemente un valor de escala y luego el personaje huiría tomando como medida de la prioridad que le tiene que dar ese valor y una FLV para saber que tanto empeño debe ponerle al ataque que podría tener una escala igual que la huida. El ejemplo seguramente podría tener mas reglas y tal vez los conjuntos difusos en los FLV para lograr mejores resultados.

Hasta ahora hemos visto la mitad del proceso que implica un sistema difuso ya que llegamos hasta la formulación de las reglas difusas como podemos ver en la Ilustración 1. Además estudiamos suficientes conceptos teóricos como para comenzar a ver como se usan en la práctica… y suficientes conceptos como para perder a más de uno seguramente, pero no desesperen porque la lógica difusa es una herramienta muy útil y su uso no es complejo una vez que ya se sabe como funciona. Por esto a continuación comenzaremos viendo un ejemplo de aplicación de lógica difusa done iremos diseñando nuestro sistema, y una vez que hallamos terminado de formular las reglas difusas seguiremos explicando los pasos que faltan sobre el mismo ejemplo.

Un caso de Aplicación: Sistema de selección de armas

Imaginemos que estamos queriendo desarrollar un juego estilo Quake o Crimsonland donde debemos realizar la inteligencia artificial de un bot que posee armas con las que ataca a sus enemigos. Como nuestro juego tendrá muchas armas y todas ellas pueden ser usadas por el bot, éste deberá decidir que arma le conviene usar en cada momento. En comienzo podemos pensar que esto no es en realidad un problema, pero en realidad es una decisión compleja que depende

de muchas variables. Por ejemplo imaginemos que nuestro bot esta muy cerca de un enemigo, elegir un arma que cause una explosión como un lanzacohetes es una mala elección porque nos causara daño a nosotros mismos o elegir un rifle de francotirador también será una mala elección porque tiene una cadencia de disparo muy baja. Además las municiones de las armas no son ilimitadas, así que si un arma para una situación es la mas propicia pero tenemos muy poca munición para ella seguramente no será una buena elección porque podemos quedarnos sin munición en el medio de la balacera (y posiblemente el resto de los jugadores no serán lo suficientemente caballeros para esperar a que recargue :P). Incluso mas allá de estos dos parámetros que dijimos existen muchos otros, por ejemplo si estamos en un ambiente cerrado y con paredes muy cerca elegir un lanzacohetes no será buena elección y elegir un arma especial como un lanza llamas seria aconsejable. En fin nuestro bot debe tener todas esas cosas en cuenta si quiere aparentar humanidad y ser un desafío en el juego. Nos concentraremos en este ejemplo en desarrollar un sistema difuso para la selección de armas que usará un bot.

Este ejemplo es muy fácil de resolver usando lógica difusa ya que es fácil formular reglas difusas para el problema. Tendremos un sistema difuso para cada arma que tengamos en nuestro juego y cada uno de los sistemas difusos tendrá en cuenta dos parámetros del estado del bot para decidir que serán la cantidad de munición que tenga el bot del arma que modele y la distancia al enemigo mas cercano. La salida del sistema difuso será un valor que nos dirá que tan deseable es esta arma para la situación en la que se encuentra el bot. Podemos ver lo que explicamos en la Ilustración 11. Entonces luego de que tenemos los sistemas difusos para cada arma podemos evaluar cual es la deseabilidad para cada uno con los parámetros actuales del bot y el arma que obtenga como resultado es el arma que debe usar el bot en este momento. Un detalle que si bien no es fundamental verlo ahora pero si es interesante es que esta evaluación para elegir la mejor arma la ejecutara varias veces el bot a lo largo del juego. Pero no es necesario que se ejecute cada frame del juego, podría ejecutar esta evaluación tal vez cada un segundo o cada treinta frames lo que ayudaría a que la inteligencia artificial no consuma muchos recursos y además que no cambie de arma demasiado seguido el bot si se encuentra en una situación donde cambios muy pequeños pueden hacer que cambie el arma que tiene la máxima deseabilidad.

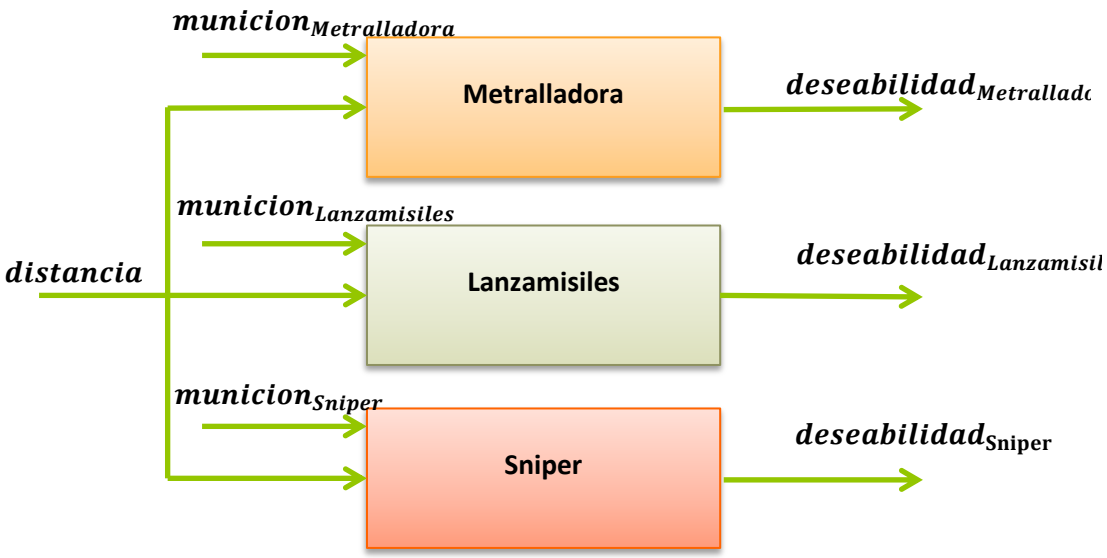

ILUSTRACIÓN 11: SISTEMA DE SELECCIÓN DE ARMAS. $distancia$ ES LA $distancia$ AL ENEMIGO MÁS CERCANO, $municion_{\propto}$ ES LA CANTIDAD DE MUNICIÓN DEL ARMA $\propto$ Y $deseabilidad_{\propto}$ ES UN VALOR QUE MIDE QUE TAN ÚTIL ES EL ARMA $\propto$ EN LA SITUACIÓN ACTUAL.

Nosotros nos concentraremos en desarrollar el sistema difuso del lanzamisiles, el resto de las armas se pueden resolver de forma muy similar. Incluso la entrada de distancia será la misma para todas las armas, solo cambiaran las entradas de municiones y por supuesto las reglas difusas ya que algunas armas son deseables un una situación y otras no.

Diseño de las FLV

Antes de comenzar con cualquier otra cosa es necesario identificar cuales serán las FLV de nuestro sistema. Recordemos que estas serán las FLV modelaban un concepto sobre el cual luego crearemos conjuntos difusos. En nuestro ejemplo tendremos tres FLV, dos de entrada y una de salida:

1. **DESEABILIDAD**: Esta FLV será la salida del sistema difuso y modelara el valor de que tan deseable es el arma. El dominio de esta será un valor numérico entre 0 y 100, donde un numero grande será mas deseable (valga la redundancia) que uno pequeño.
2. **DISTANCIA**: Esta FLV será una de las entradas del sistema difuso y modelara la distancia a la que se encuentra el enemigo más cercano. El dominio de esta será la distancia al enemigo en unidades que pueden ser por ejemplo metros o tal vez simplemente unidades de distancia de nuestro mundo (o incluso pixeles si es un juego 2D).
3. **MUNICION**: Esta FLV será la otra entrada del sistema difuso y modelara la cantidad de municiones que tendrá el bot.

Como dijimos antes el FLV DESEABILIDAD y DISTANCIA estarán definidos de la misma forma para todos los sistemas difusos de armas, el único que cambiara es el de MUNICION ya que para un arma como la Metralladora 100 balas es poco pero para una como el Lanzamisiles es muchísimo. Ya veremos esto mas claro a continuación cuando definamos los conjuntos difusos para cada FLV.

Creación de Conjuntos difusos

Antes de comenzar a crear conjuntos difusos y ya que este es un ejemplo práctico vemos dos reglas que si bien no son obligatorias son típicas de un buen diseño de conjuntos difusos:

i. Para cada valor del dominio del FLV la suma del grado de pertenencia de todos los conjuntos debe ser aproximadamente uno.
ii. Una línea vertical sobre el FLV no debería cruzar más de dos conjuntos difusos.

Podemos ver ejemplos de malos diseños según esas reglas en la Ilustración 12.

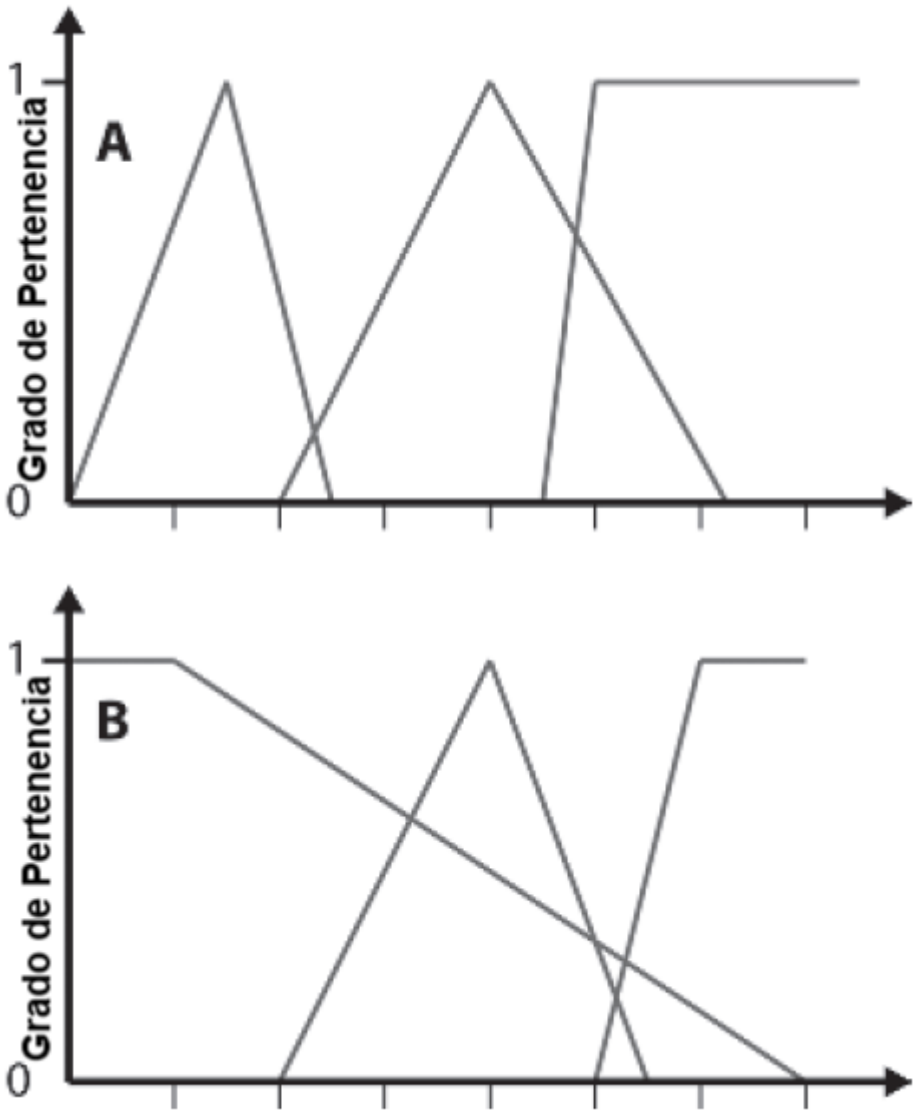

ILUSTRACIÓN 12: FLVS CON UN MAL DISEÑO. A POSEE ELEMENTOS DEL DOMINIO CON GRADO DE PERTENENCIA MAYOR A UNO. B TIENE PUNTOS DONDE SE SUPERPONEN MAS DE DOS CONJUNTOS DIFUSOS.

Conjuntos de DESEABILIDAD

El la FLV **DESEABILIDAD** proponemos tres conjuntos difusos que abarcaran los valores desde 0 a 100. Estos conjuntos son: INDESEABLE, DESEABLE y MUY_DESEABLE donde por supuesto el primer conjunto difuso abarcara los valores que se consideran en cierta medida indeseables, el segundo los valores que se consideran deseables y los últimos aquellos que son muy recomendables. Proponemos además que tengas las funciones de pertenencia con forma left shoulder, triangular y right shoulder respectivamente. Entonces la distribución que proponemos (cumpliendo las recomendaciones i e ii) son:

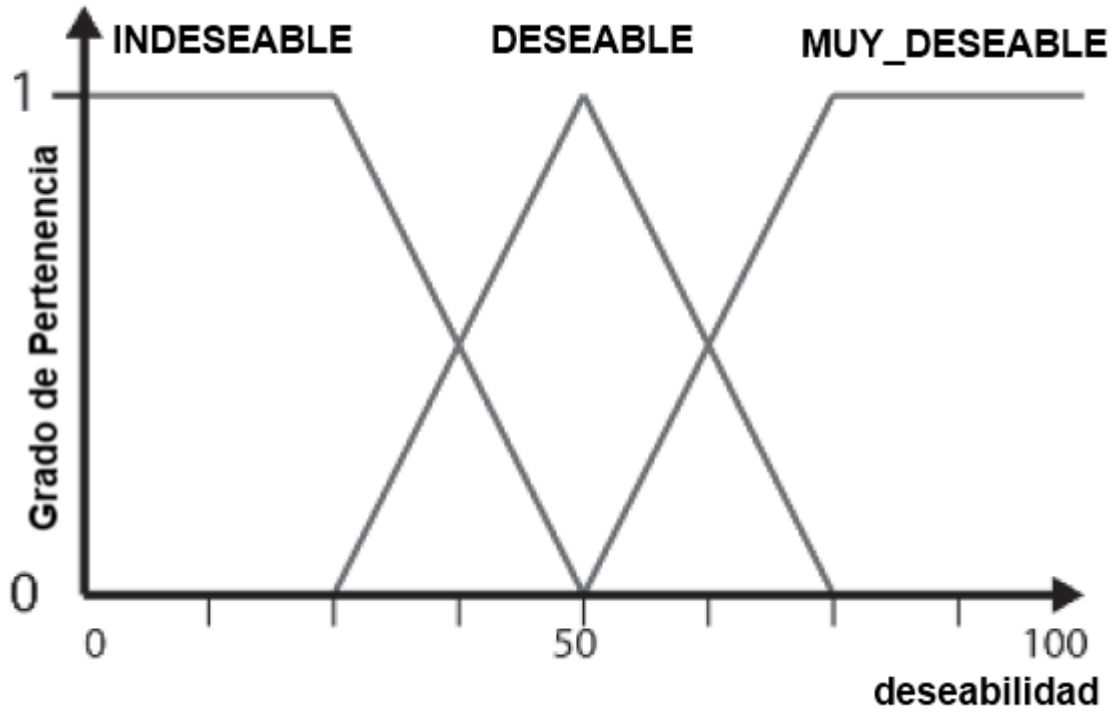

ILUSTRACIÓN 13: CONJUNTOS DIFUSOS DE LA FLV DESEABILIDAD

De esta forma los valores bajos los abarcan INDESEABLE y se tiene la precaución de darle un grado de pertenencia de 1 al mínimo de nuestro dominio que es 0. De forma análoga el conjunto MUY_DESEABLE abarca los valores altos teniendo la precaución de asignarle grado de pertenencia 1 al valor máximo de nuestro dominio que es 100. Al mismo tiempo el conjunto difuso DESEABLE esta entre estos dos, logrando que al recorrer de izquierda a derecha los valores bajos gradualmente dejen de ser indeseables y comiencen a ser deseables y que los valores medios gradualmente dejen de ser deseables para convertirse en muy deseables.

Conjuntos de DISTANCIA

En este caso consideraremos que las distancias están dadas en pixeles por lo que usaremos valores que numéricamente son grandes pero en pantalla representan poca distancia. Proponemos tener tres conjuntos como en el caso anterior estos son: BLANCO_CERCA, BLANCO_MEDIANA y BLANCO_LEJOS. Al igual que el caso anterior propondremos usar funciones de forma de left shoulder, triangular y right shoulder respectivamente. Ahora decidir desde que cantidad de pixeles consideramos que un enemigo esta cerca, a mediana distancia y lejos es muy subjetivo y en realidad lo que posiblemente definiría eso seria la onda expansiva del lanzamisiles... sin mencionar la resolución a la que corra el juego lo cual nos da la idea de que medir distancia en pixeles no es por un comienzo una muy buena idea, pero no nos preocupemos por esos detalles y sigamos. La distribución que usaremos será:

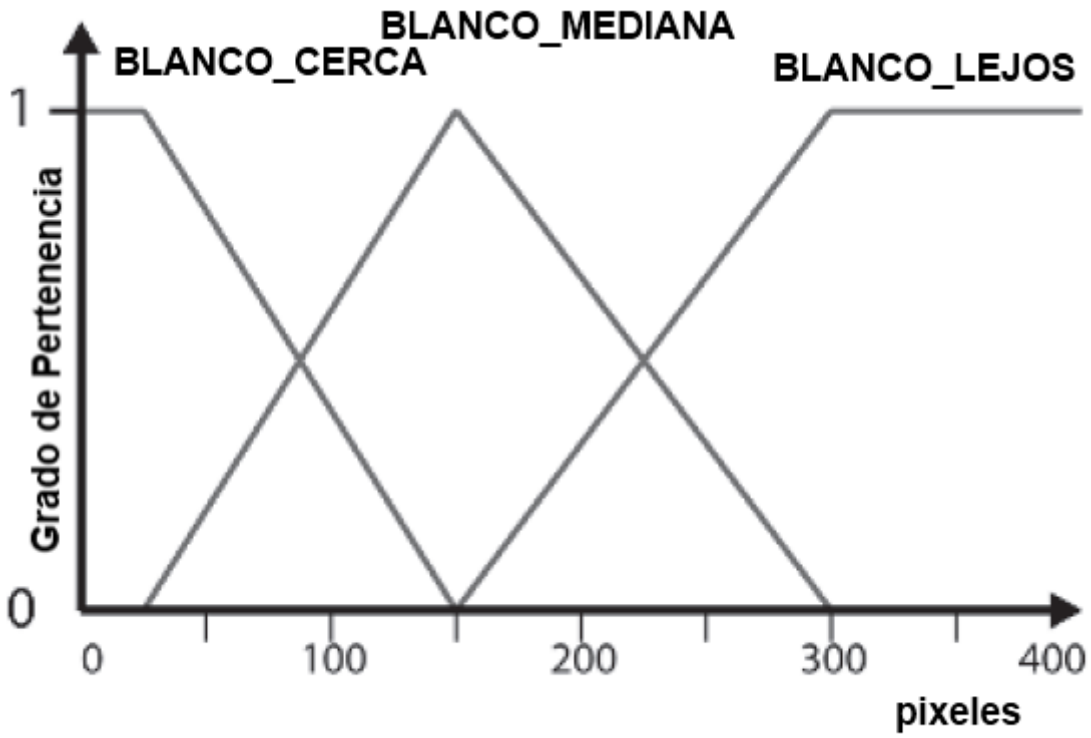

ILUSTRACIÓN 14: CONJUNTOS DIFUSOS DE LA FLV DISTANCIA

Conjuntos de MUNICION

Esta ultima FLV tiene conjuntos difusos que como dijimos antes son muy dependientes del arma que estamos modelando, lo que hará que usemos valores pequeños de cantidad de munición ya que como seguramente los misiles son pesados y difíciles de llevar, a menos que nuestro bot sea Guybrush Threepwood. Nuestro bot podrá llevar entre 0 y 40 misiles. Proponemos tres conjuntos que será: MUNICION_POCA, MUNICION_OK y MUNICION_LLENA. En el segundo conjunto lo ubicaremos en los valores que consideramos que se pueden considerar en cierta medida como seguros como para entablar batalla y el primero sobre los valores que presentan cierto grado de peligro. La distribución que proponemos es:

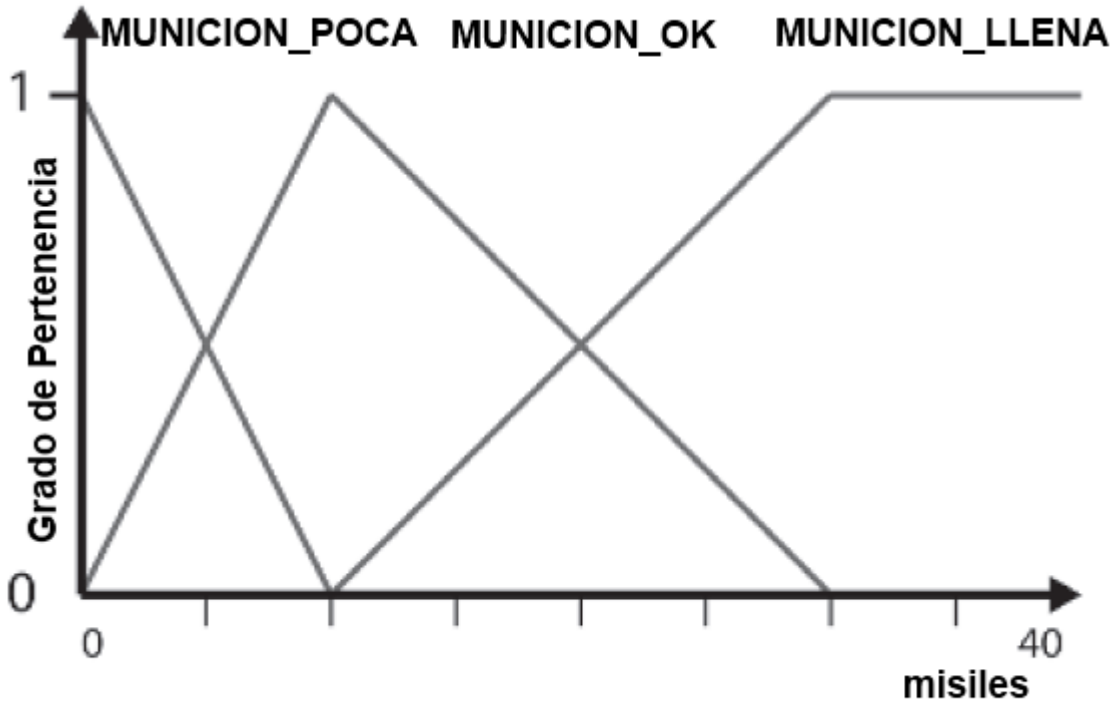

ILUSTRACIÓN 15: CONJUNTOS DIFUSOS DE LA FLV MUNICION

Notemos que proponemos un conjunto MUNICION_POCA que solo tiene un elemento con grado de pertenencia de 1 que es el valor 0, para el resto tiene una cantidad menor y decreciente con la cantidad de munición. Esta decisión se tomo ya que como dijimos para nosotros (nuestras reglas difusas que definiremos a continuación) tener un valor con un valor alto de pertenencia a este conjunto es algo crítico.

Inferencia Difusa

Por fin ya tenemos todos nuestros conjuntos difusos definidos. Ahora nos queda formular las reglas difusas que controlaran la lógica del sistema (una tarea crítica por supuesto). Antes de definir nuestras reglas veamos como evaluaremos las reglas difusas, a lo que se le llama inferencia difusa, para así a continuación poder plantear las reglas y poder entender como obtener resultados con ellas.

1. Por cada regla difusa:
    a. Por cada antecedente calcular el grado de pertenencia del valor ingresado.
    b. Calcular el término del antecedente usando las pertenencias de 1.a y en usar este como la activación del consecuente.
2. Combinar todas las consecuencias de las reglas en un solo conjunto difuso.
3. Desfuzzificar el conjunto calculado en 2 para obtener el valor nítido de salida.

No se asusten por no entender exactamente a implican esos pasos, en los próximos pasos plantearemos las nueve reglas (podrían ser mas, pero con estas podemos tener buenos resultados) que propondremos para nuestro sistema. Para ver la evaluación de las reglas supongamos que la entrada al sistema difuso fue:

- $distancia$, el bot tiene su enemigo más cercano a 200 pixeles de distancia.
- $munucion$, el bot tiene en este momento 8 misiles.

Las reglas que proponemos son:

1. IF *BLANCO_LEJOS* AND *MUNICION_LLENA* THEN *DESEABLE*
2. IF *BLANCO _ LEJOS* AND *MUNICION _OK* THEN *INDESEABLE*
3. IF *BLANCO _ LEJOS* AND *MUNICION _ POCA* THEN *INDESEABLE*
4. IF *BLANCO _MEDIA* AND *MUNICION _ LLENA* THEN *MUY_DESEABLE*
5. IF *BLANCO _ MEDIA* AND *MUNICION _OK* THEN *MUY_DESEABLE*
6. IF *BLANCO _ MEDIA* AND *MUNICION _ POCA* THEN *DESEABLE*
7. IF *BLANCO _CERCA* AND *MUNICION _ LLENA* THEN *INDESEABLE*
8. IF *BLANCO _ CERCA* AND *MUNICION _OK* THEN *INDESEABLE*
9. IF *BLANCO _ CERCA* AND *MUNICION _POCA* THEN *INDESEABLE*

Estas reglas se decidieron como hemos venido diciendo usando nuestra experiencia como jugadores y nada mas. Por ejemplo sabemos que si estamos cerca del enemigo el lanzamisiles es una mala elección, por eso las últimas tres reglas. A continuación explicaremos la lógica de las dos primeras reglas y mostraremos como se evalúan para el caso de ejemplo.

Regla 1:

En esta primera regla diremos que si el enemigo esta lejos y tenemos varios misiles entonces elegir el lanzamisiles es una buena alternativa. Esto es:

IF BLANCO_LEJOS AND MUNICION_LLENA THEN DESEABLE

Para la entrada de distancia al objetivo de 200, el grado de pertenencia al conjunto difuso BLANCO_LEJOS es 0.33. Para la entrada de munición de 8, el grado de pertenencia al conjunto difuso MUNICION_LLENA es 0. Como vimos el operador AND resulta en el mínimo de la pertenencia de los dos conjuntos:

IF $P_{\text{BLANCO\_LEJOS}}(200)$ AND $P_{\text{MUNICION\_LLENA}}(8)$ THEN DESEABLE

IF 0.33 AND 0 THEN DESEABLE

IF $min\{0.33,0\}$ THEN DESEABLE

IF 0 THEN DESEABLE

Entonces la activación del conjunto difuso consecuente DESEABLE será0. Es decir que no se activara DESEABLE. Gráficamente esto es:

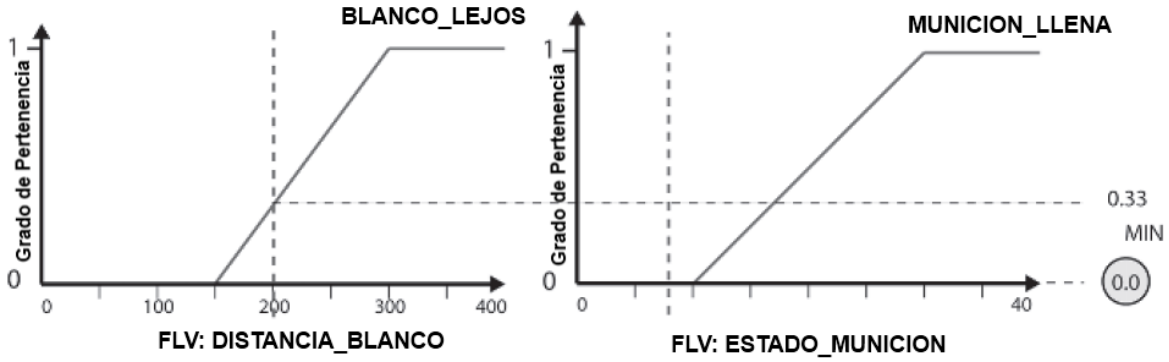

ILUSTRACIÓN 16: CALCULO GRAFICO DE LA ACTIVACIÓN DEL CONSECUENTE PARA LA REGLA 1. VEMOS DISTANCIA 200 TIENE PERTENENCIA 0.33 PARA EL CONJUNTO BLANCO_LEJOS Y MUNICION 8 TIENE PERTENENCIA 0 PARA EL CONJUNTO MUNICION_LLENA. ADEMÁS EL AND ES LA OPERACIÓN MINIMO.

Regla 2 :

En esta segunda regla diremos que si el enemigo esta lejos y tenemos varios una cantidad más restringida de misiles entonces elegir el lanzamisiles no es tan buena idea ya que si esta lejos fácilmente podemos errar misiles ya que es fácil esquivar un misil y así desperdiciar los pocos misiles que tenemos. Esto es:

IF BLANCO_LEJOS AND MUNICION_OK THEN INDESEABLE

De forma análoga al cálculo de la regla 1 para una distancia de 200, el grado de pertenencia para el conjunto BLANCO_LEJOS es 0.33. Mientras que para la entrada de munición de 8, el grado de pertenencia al conjunto difuso MUNICION_OK es 0.78. La operación AND produce el mínimo de pertenencias, por lo que la activación del conjunto difuso consecuente INDESEABLE es de 0.33. Gráficamente esto es:

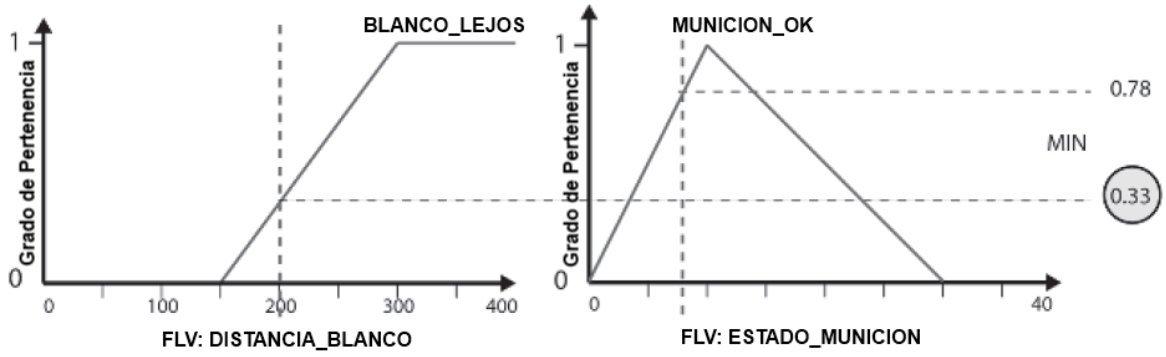

ILUSTRACIÓN 17: CALCULO GRAFICO DE LA ACTIVACIÓN DEL CONSECUENTE PARA LA REGLA 2. VEMOS DISTANCIA 200 TIENE PERTENENCIA 0.33 PARA EL CONJUNTO BLANCO_LEJOS Y MUNICION 8 TIENE PERTENENCIA 0.78 PARA EL CONJUNTO MUNICION_OK. ADEMÁS EL AND ES LA OPERACIÓN MINIMO.

Seguimos el cálculo para el resto de las reglas y calculamos la activación del conjunto consecuente de cada regla. Podemos representar estos valores de activación de forma resumida en lo que se conoce como Matriz Asociativa Difusa o Fuzzy Associative Matrix (FAM) de la siguiente forma:

MUNICION_LLENA MUNICION_OK MUNICION_POCA	BLANCO_CERCA	BLANCO_MEDIA	BLANCO_LEJOS
	INDESEABLE **0**	**DESEABLE** **0.2**	**INDESEABLE** **0.2**
	INDESEABLE **0**	**MUY_DESEABLE** **0.67**	**INDESEABLE** **0.33**
	INDESEABLE **0**	**MUY_DESEABLE** **0**	**INDESEABLE** **0**

Si un par de conjuntos difusos no están relacionados por una regla entonces consideraríamos que no activan ningún conjunto, en nuestro ejemplo consideramos todas las combinaciones de conjuntos por lo que no sucede. Entonces vemos que el conjunto DESEABLE fue activado con una

activación de 0.2 y el conjunto MUY_DESEABLE con una activación de 0.67. Por otro lado el conjunto INDESEABLE fue activado dos veces con activaciones diferentes, acá podemos hacer dos cosas:

i. **Sumar las dos activaciones y truncar a 1. Es decir en el ejemplo seria:**
$$\boldsymbol{Truncar(0.2 + 0.33) = Truncar(0.53) = 0.53}$$ **ya que no se pasa de 1**

ii. **Hacer un OR entre las dos activaciones (tomar el máximo). Es decir en el ejemplo seria:**
$$\boldsymbol{0.2\ OR\ 0.33 = max\{0.2, 0.33\} = 0.33}$$

Las dos alternativas son posibles pero notros usaremos la segunda, es decir tomaremos el máximo.

Entonces los niveles de activación que calculamos será:

INDESEABLE	0.33
DESEABLE	0.2
MUY_DESEABLE	0.67

El conjunto resultante de cada regla es el consecuente de la regla truncando al valor de su activación. Entonces gráficamente la activación de los conjuntos nos da los siguientes resultados gráficamente:

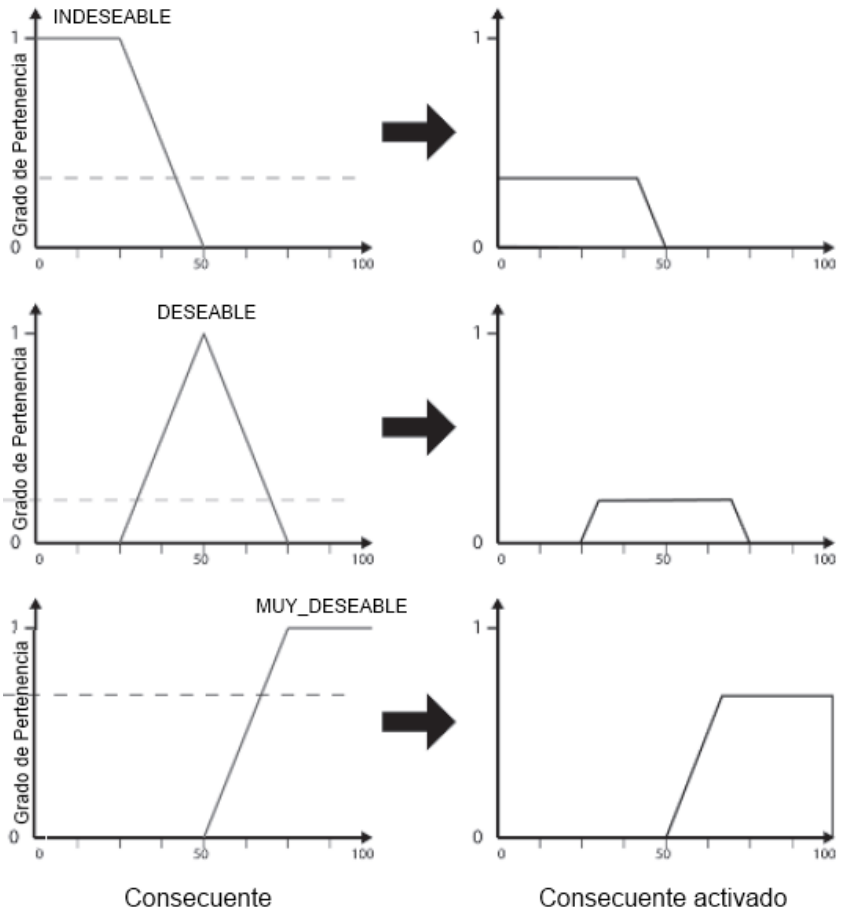

ILUSTRACIÓN 18: ACTIVACIONES

Nos queda combinar todos estos conjuntos activados en un solo conjunto difuso. Esta combinación se puede realizar de muchas maneras. La forma que usaremos, que es la que mas sentido gráficamente hablando tiene es la de combinarlas usando operadores OR, es decir tomando el valor máximo de todos los conjuntos. Gráficamente esto es:

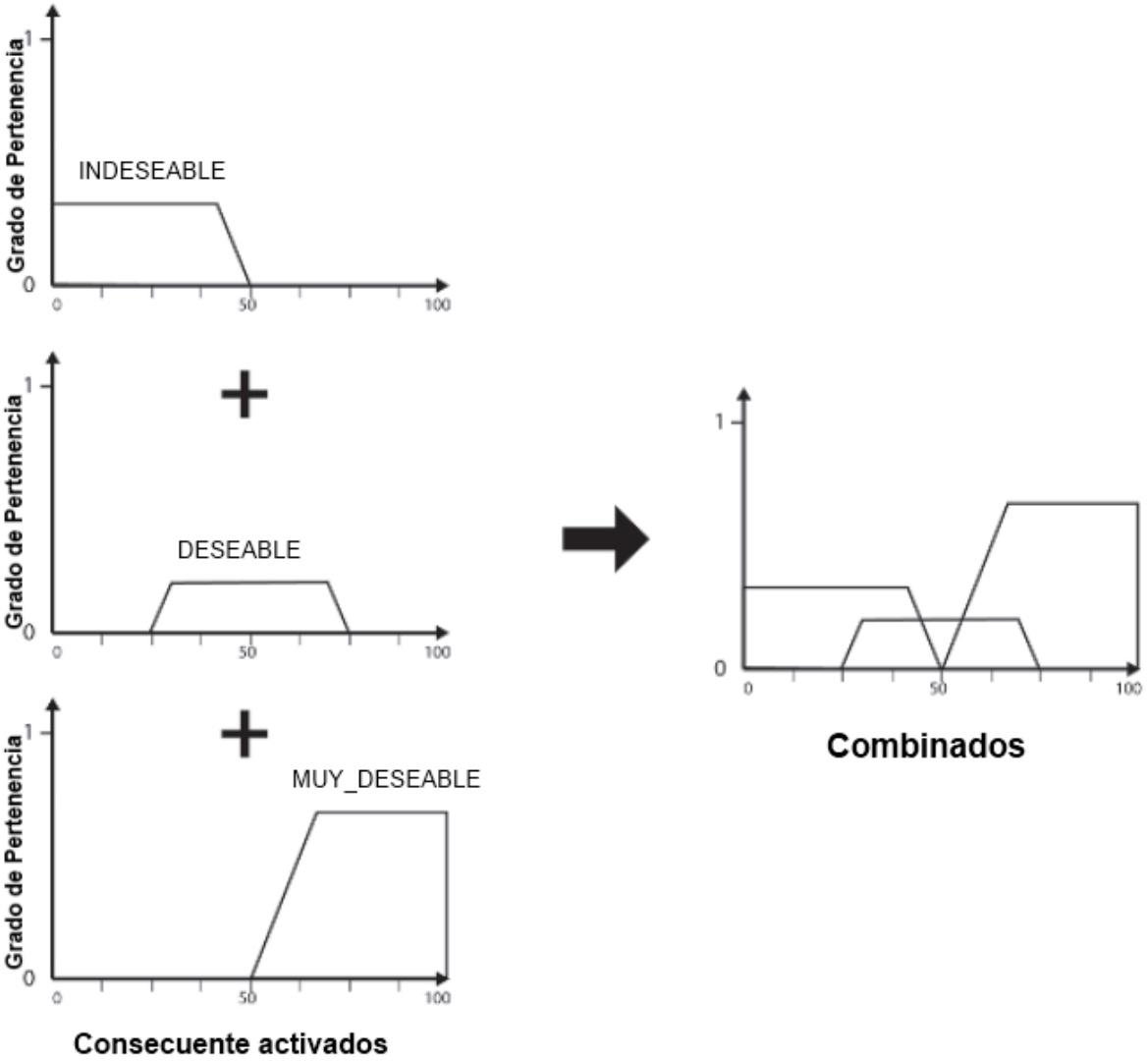

ILUSTRACIÓN 19: COMBINACIÓN

Desfuzzificación

La desfuzzificación como dijimos deberemos en base al resultado de la combinación obtener un único valor nítido. Veremos la técnica más correcta para realizar esta, pero existen otras formas de realizar la desfuzzificación.

Centroide:

Esta técnica es la más correcta pero también la más cara computacionalmente hablando de calcular. Lo que calculamos es el centro de masa o de balance de la figura, la FLV, resultante de la combinación que es el punto donde tanto a la izquierda como a la derecha tiene igual área (o masa). Podemos ver esto en la Ilustración 21. Usar este punto en la desfuzzificación tiene sentido ya que al ser el centro de balance de la figura eso significara que tanto a la derecha como a la izquierda tendrá igual masa o mas específicamente grado de pertenencia, es decir que el valor que elegiremos será el que esta en el medio de los grados de pertenencia. Por ejemplo en la combinación resultante del ejemplo que vimos anteriormente seria:

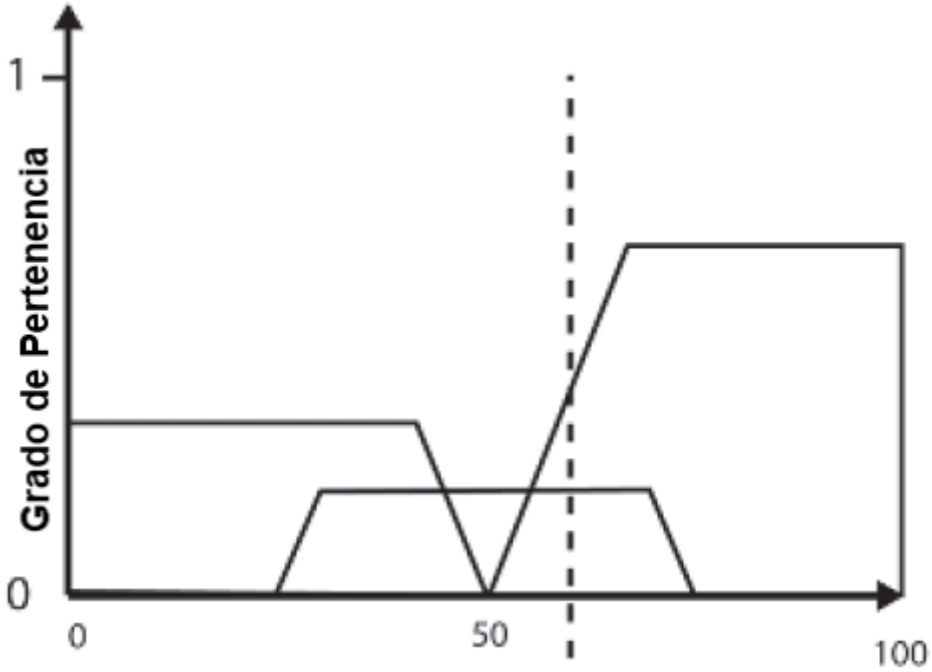

ILUSTRACIÓN 20: LA LÍNEA VERTICAL PUNTEADA ES EL PUNTO DEL VALOR DEL CENTROIDE

Para realizar este cálculo lo que haremos es cortar el dominio en muchas secciones y en cada sección que cortamos calcular el grado de pertenencia al conjunto y pesarlo por el valor del dominio, al resultado de esa operación se la divide por la suma de los grados de pertenencia de todos los cortes. Mientras en mas cortes realicemos, es decir mas puntos tomemos, mas preciso será el resultado, aunque en la practica realizando entre diez y veinte cortes es suficiente. Entonces la formula para el cálculo es:

$$valor = \frac{\sum_{s=minomo\ Dominio}^{s=maximo\ Dominio} s\ DOM(s)}{\sum_{s=minomo\ Dominio}^{s=maximo\ Dominio} DOM(s)}$$

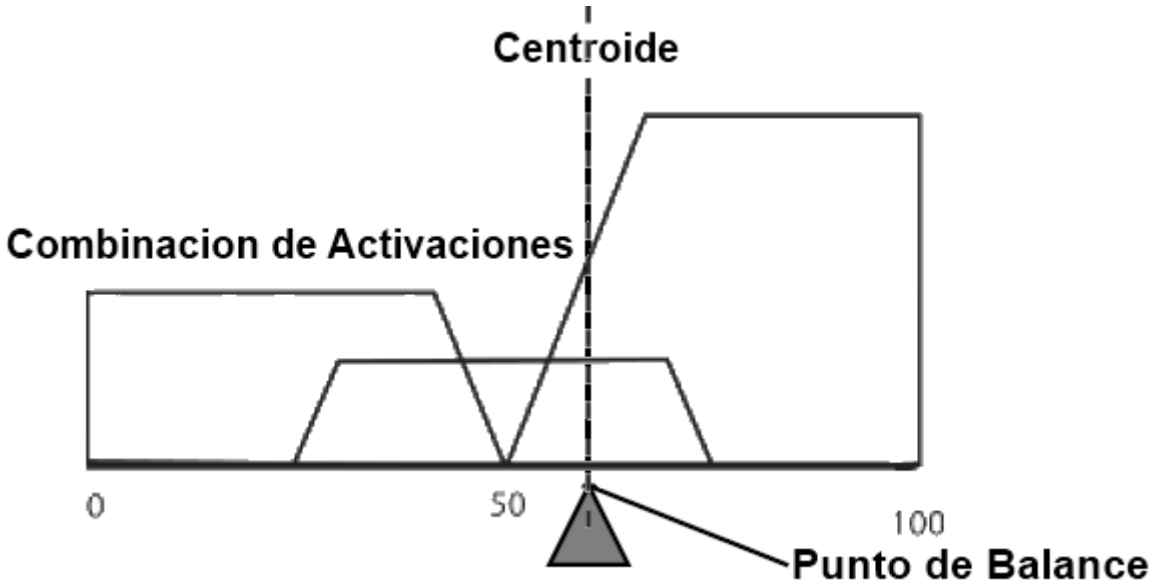

ILUSTRACIÓN 21: EL CENTROIDE ES EL PUNTO DE BALANCE DE LA FIGURA

Donde DOM(s) es la función de pertenencia del valor s y el mínimo y máximo del dominio se refiere al valor mínimo y máximo que puede tomar el dominio del conjunto difuso. Para verlo más claramente hagamos el cálculo de un ejemplo. Tomemos la figura que obtuvimos como la combinación en la etapa anterior. Para realizar el calculo realicemos diez cortes en el dominio, es decir que tomaremos diez puntos de la figura. Podemos ver esto gráficamente en la Ilustración 22.

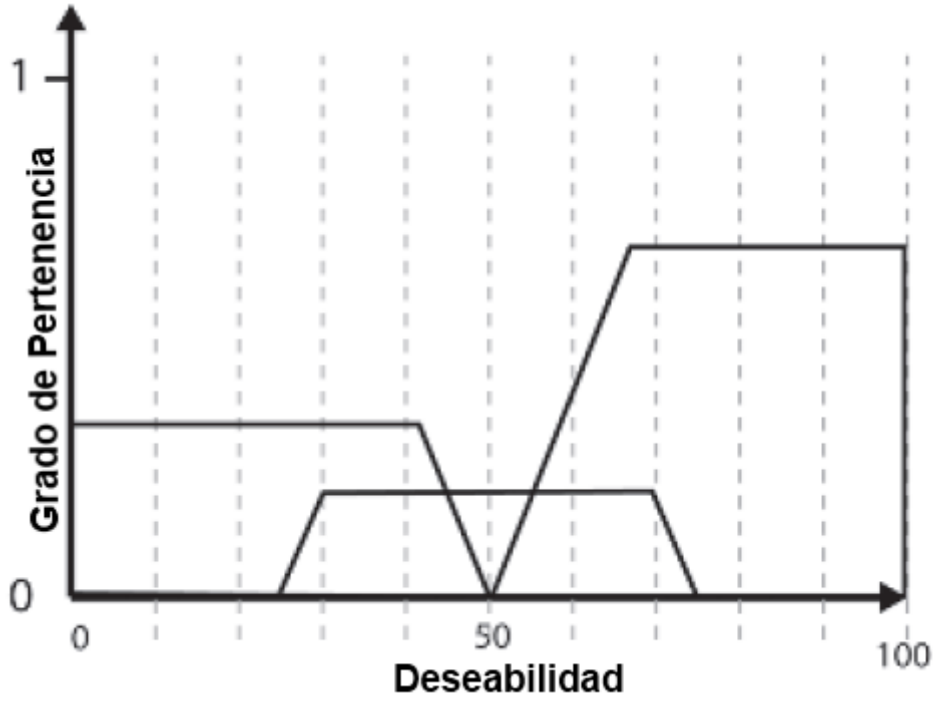

ILUSTRACIÓN 22: DIVISIÓN DEL DOMINIO PARA CALCULAR EL CENTROIDE

s	INDESEABLE	DESEABLE	MUY_DESEABLE	$DOM(s)$	$s\ DOM(s)$
10	0.33	0	0	**0.33**	**3.3**
20	0.33	0	0	**0.33**	**6.6**
30	0.33	0.2	0	**0.53**	**15.90**
40	0.33	0.2	0	**0.53**	**21.20**
50	0	0.2	0	**0.2**	**10**
60	0	0.2	0.34	**0.54**	**32.40**
70	0	0.2	0.67	**0.87**	**60.90**
80	0	0	0.67	**0.67**	**53.60**
90	0	0	0.67	**0.67**	**60.30**
100	0	0	0.67	**0.67**	**67**
Total				**5.34**	**331.20**

Con los cálculos de la tabla es fácil concluir:

$$cantroide = \frac{\sum_{s=minomo\ Dominio}^{s=maximo\ Dominio} s\ DOM(s)}{\sum_{s=minomo\ Dominio}^{s=maximo\ Dominio} DOM(s)} = \frac{331.20}{5.34} \approx 62{,}02$$

Entonces la salida de la desfuzzificación de este ejemplo es que el valore de deseabilidad para el arma en estas circunstancias es de 62.02, si este es mayor a la deseabilidad del resto de armas entonces seria conveniente cambiar y usar esta arma para que el personaje siga llevando el infierno a sus enemigos.

Resumen de Sistema Difuso o Memoria Asociativa Borrosa (FAM)

Los procesos de un sistema difuso no son fáciles de recordar, aunque la parte más desagradable esta en entenderlo en un primer lugar. Por eso escribimos esta sección para resumir todos los pasos que se pueden ver esquematizados en la Ilustración 23. En el esquema la variable x es el valor nítido de entrada al sistema, $\boldsymbol{A}$ es la fuzzificación de la entrada x, $\widetilde{\boldsymbol{A}}_{\boldsymbol{n}}$ es el antecedente de la regla $\boldsymbol{n}$ (puede ser un termino complejo entre conjuntos difusos), $\widetilde{\boldsymbol{B}}_{\boldsymbol{n}}$ es el consecuente de la regla $\boldsymbol{n}$ (puede ser uno o un conjunto de conjuntos difusos), $\widetilde{\boldsymbol{B}}_{\boldsymbol{n}}{}'$ es la activación del consecuente de la regla $\boldsymbol{n}$, $\sum$ representa la combinación de activaciones (nosotros combinamos las activaciones con operaciones OR), $\boldsymbol{B}$ es el conjunto combinado de activaciones, por ultimo y es el valor nítido de salida del sistema.

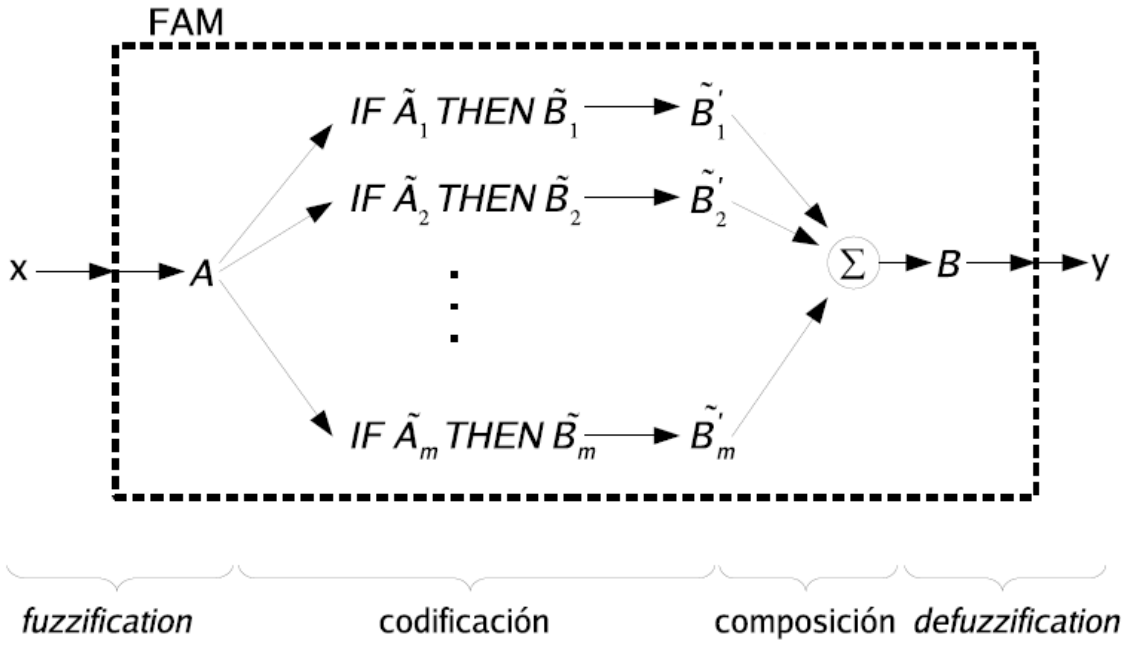

ILUSTRACIÓN 23: RESUMEN DE SISTEMA DIFUSO

Uso del código

Ya conocemos toda la teoría que hay que entender para poder usar la lógica difusa, ahora veamos como utilizar el código que les facilitamos para diseñar sistemas difusos y evaluarlos. Antes que nada hagamos un listado de las clases que hay y que hace cada una, luego veamos como se usan. Entonces tendremos las siguientes clases:

- FuzzyModule: Esta es la clase principal ya que representa a un sistema difuso completo. Dentro de estas crearemos FLV y reglas difusas para el sistema. Al mismo tiempo especificaremos cuales son las entradas y las salidas del sistema. Una vez que hallamos configurado estas propiedades del sistema obtener salidas será tan fácil como poner las entradas de ese momento (por ejemplo en el caso del modulo de selección de armas la distancia al enemigo mas cercano y la cantidad de munición de esa arma) y llamar a una función para obtener las salidas (en el caso del ejemplo del modulo de selección seria solo la deseabilidad del arma).

- FuzzyVariable: Esta clase representa las FLV de nuestro diseño. A objetos de esta clase se le asignaran conjuntos difusos, los que determinaras el grado de pertenencia de cada valor asi como también el máximo y mínimo del dominio de la variable lingüística. Algunos ejemplos de objeto de esta clase seria el FLV **DISTANCIA**, **MUNICION** y **DESEABILIDAD** del ejemplo del selector de armas.

- FuzzySet: Esta clase es en realidad una clase abstracta que representa un conjunto difuso. En la practica por supuesto que jamás instanciaremos un objeto de este tipo, pero si instanciaremos objetos de las clases que heredan de esta que son una clase para cada tipo de conjunto básico que vimos arriba, es decir triángulos, rectángulos, trapecios, right shoulders, left shoulders, etc. Mas adelante veremos estas clases en particular. Lo que tenemos que tener en claro es que todas las clases que hereden de esta se asignaran a un FLV al que pertenezcan, por ejemplo si creamos el conjunto MUNICION_POCA se asignaría al FLV **MUNICION**.
- FuzzyTerm**:** Esta clase al igual que al anterior es abstracta y por esto no se instanciara. Pero las clases hijas que heredan de esta nos permitirán crear reglas difusas, potencialmente complejas, para asignar al sistema difuso. Por ejemplo podremos crear una regla como IF OBJETIVO_LEJOS AND MUNICION_LLENA THEN DESEABLE para nuestro sistema de selección de armas.

Creación de sistema difuso

Crear el sistema difuso será el primer paso a realizar ya que para crear el resto de objetos será necesario hacer llamadas a métodos de un objeto FuzzyModule, ya que tanto las reglas difusas como las variables lingüísticas existen dentro de un sistema difuso. Para crear un objeto de este tipo solo será necesario crearlo sin pasar parámetros al constructor ya que luego a través de métodos iremos creando las reglas difusas y variables lingüísticas del sistema. El código necesario para crear el sistema difuso para selección de armas será:

```
//  Creacion del sistema difuso para selección de armas
FuzzyModule   sisSeleccionLanzamisil;
```

Creación de FLV

Crear la FLV será uno de nuestros primeros pasos ya que los conjuntos difusos se crean sobre una FLV. Para crear un objeto de este tipo sólo es necesario pedirle a un sistema difuso que la cree. Para ello usaremos la función de FuzzyModule:

```
  //crea un objeto FuzzyVariable vacia dentro del objeto FuzzyModule
FuzzyVariable&  FuzzyModule::CreateFLV(const std::string& VarName);
```

Para crear la variable lingüística **DISTANCIA** del ejemplo de selección de armas hacemos:

```
//  Creacion de FLV usando el modulo difuso sisSeleccionLanzamisil
FuzzyVariable& DISTANCIA = sisSeleccionLanzamisil.CreateFLV("DISTANCIA");
```

Creación de conjuntos difusos

Veamos como crear conjuntos difusos usando las clases que heredan de FuzzySet, estas son:

- FuzzySet_Triangle: Representa una función de grado de pertenencia triangular.
- FuzzySet_RightShoulder: Representa una función de grado de pertenencia right shoulder.
- FuzzySet_LeftShoulder: Representa una función de grado de pertenencia left shoulder.

Para crear un conjunto triangular dentro de un FLV llamaremos al método de FuzzyVariable siguiente:

```
FzSet  AddTriangularSet(std::string name,
                        double       minBound,
                        double       peak,
                        double       maxBound);
```

Donde los parámetros a pasarle están mostrados en la Ilustración 24. El parámetros peak es el valor del dominio donde el triangulo tiene grado de pertenencia 1, minBound es el valor del dominio a la izquierda donde el triangulo tiene grado de pertenencia 0, por ultimo maxBound es el valor del dominio a la derecha donde el triangulo tiene grado de pertenencia 0.

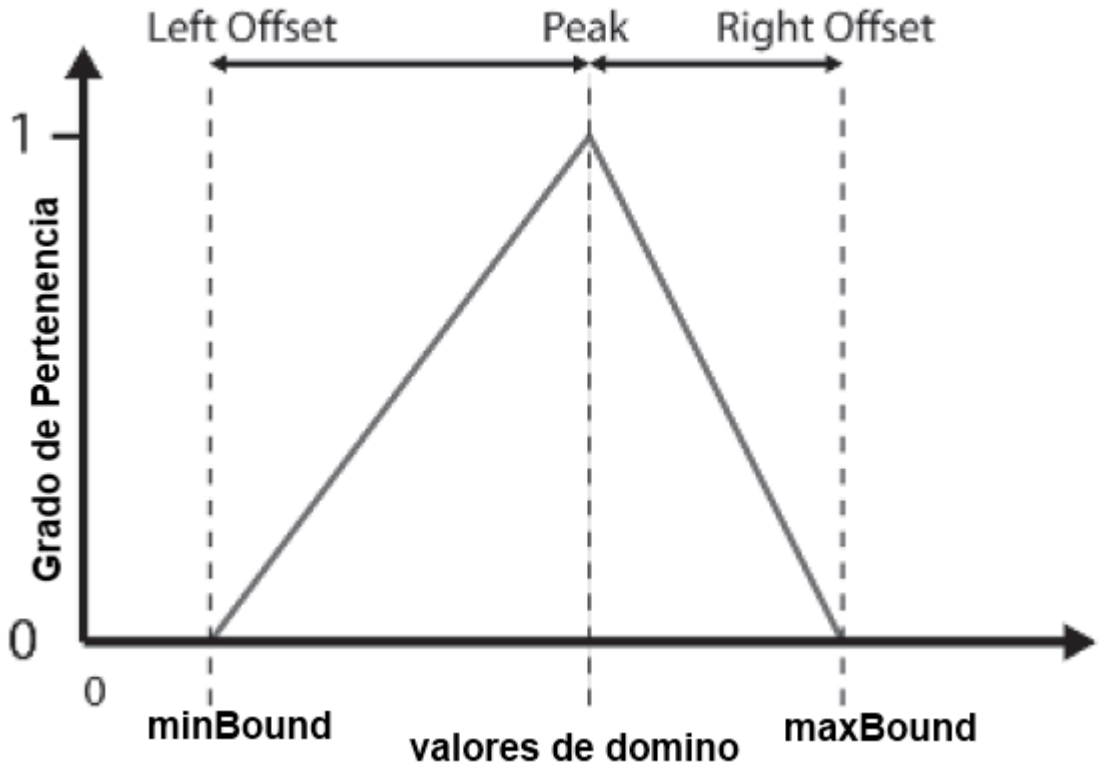

ILUSTRACIÓN 24: PARÁMETROS DE UN TRIANGULO

Para crear un conjunto right shoulder dentro de un FLV llamaremos al método de FuzzyVariable siguiente:

```
FzSet  AddRightShoulderSet(std::string name,
                           double minBound,
                           double peak,
                           double maxBound);
```

Donde los parámetros son muy parecidos a los que se les pasaba al triangulo y podemos ver lo que representan en la Ilustración 25.

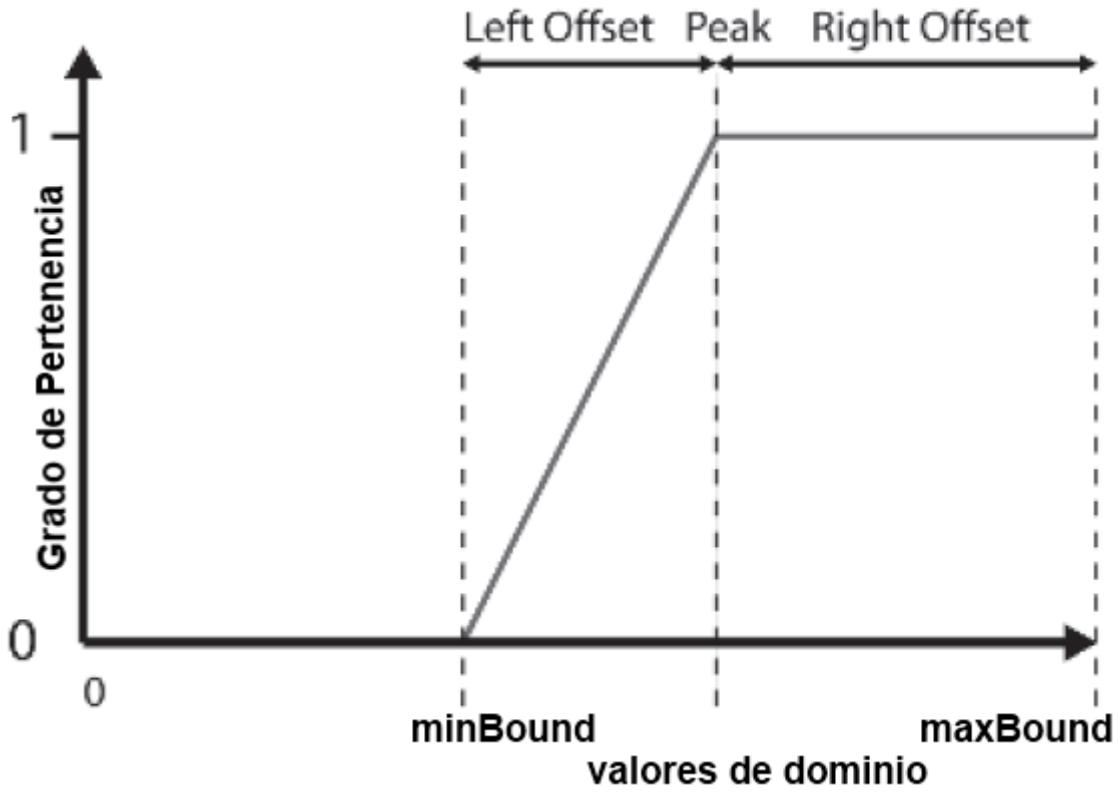

ILUSTRACIÓN 25: PARÁMETROS DE UN RIGHT SHOULDER

Es sencillo entender como se crean los distintos conjuntos difusos con diferentes formas.

Para crear los conjuntos difusos del FLV **DISTANCIA** haríamos lo siguiente:

```
FzSet& DISTANCIA_CERCA =
            DISTANCIA.AddLeftShoulderSet("DISTANCIA_CERCA",0,25,150);
FzSet& DISTANCIA_MEDIA =
            DISTANCIA.AddTriangularSet("DISTANCIA_MEDIA",25,150,300);
FzSet& DISTANCIA_LEJOS =
            DISTANCIA.AddRightShoulderSet("DISTANCIA_LEJOS",150,300,1000);
```

Creación reglas difusas

Recordemos que para crear una regla debemos definir dos términos que son el antecedente y el consecuente para lograr una regla como:

IF *antecedent* THEN *consequent*

Donde el antecedente y el consecuente son o bien un conjunto difuso o varios conjuntos difusos operados con AND, OR y NOT. Para agregar una regla difusa a un sistema difuso llamaremos al siguiente método de FuzzyModule:

```
//agrega una regla difusa con el antecedente y el consecuente que se pasa
void  FuzzyModule::AddRule(FuzzyTerm& antecedente, FuzzyTerm& consecuente);
```

Como antecedente o consecuente se pueden pasar varios parámetros:

- Un conjunto difuso solo.
- Un grupo de conjuntos difusos con operaciones AND, OR y NOT. Donde estas se pasan como se explica a continuación.

Para realizar operaciones AND se usan las siguientes clases:

```
//Realiza la operacion op1 AND op2
FzAND(FuzzyTerm& op1, FuzzyTerm& op2);

// Realiza la operacion op1 AND op2 AND op3
FzAND(FuzzyTerm& op1, FuzzyTerm& op2, FuzzyTerm& op3);

// Realiza la operacion op1 AND op2 AND op3 AND op4
FzAND(FuzzyTerm& op1, FuzzyTerm& op2, FuzzyTerm& op3, FuzzyTerm& op4);
```

Para realizar las operaciones OR se usan las siguientes clases:

```
//Realiza la operacion op1 OR op2
FzOR(FuzzyTerm& op1, FuzzyTerm& op2);

//Realiza la operacion op1 OR op2 OR op3
FzOR(FuzzyTerm& op1, FuzzyTerm& op2, FuzzyTerm& op3);

//Realiza la operacion op1 OR op2 OR op3 OR op4
FzOR(FuzzyTerm& op1, FuzzyTerm& op2, FuzzyTerm& op3, FuzzyTerm& op4);
```

La gracia de la implementación es que esta diseñada de forma tal que usando polimorfismo (AJA, vieron que servia!) se pueden mezclar resultados de operaciones AND y OR como operadores entre ellos y el resultado de cualquiera de estos usarlos como antecedente o consecuente. Podemos ver un ejemplo de esto en la próxima regla (que no tiene sentido lógicamente pero sirve para mostrar esta posibilidad):

```
//Realiza la operación MUNICION_OK OR BLANCO_CERCA AND MUNICION_LLENA
sisSeleccionLanzamisil.AddRule(
             FzOR(MUNICION_OK, FzAND(BLANCO_CERCA, MUNICION_LLENA)),
             Undesirable);
```

Entonces el código para agregar todas las reglas del sistema difuso de selección de armas para el lanzamisiles seria:

```
sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_CERCA, MUNICION_LLENA),INDESEABLE);
sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_CERCA, MUNICION_OK),INDESEABLE);
sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_CERCA, MUNICION_POCA),INDESEABLE);

sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_MEDIA, MUNICION_LLENA),MUY_DESEABLE);
sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_MEDIA, MUNICION_OK),MUY_DESEABLE);
sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_MEDIA, MUNICION_POCA),DESEABLE);

sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_LEJOS, MUNICION_LLENA),DESEABLE);
sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_LEJOS, MUNICION_OK),INDESEABLE);
sisSeleccionLanzamisil.AddRule(FzAND(BLANCO_LEJOS, MUNICION_POCA),INDESEABLE);
```

Bibliografía

"Programming Game AI by Example", Mat Buckland. Wordware Publishing, 2005.

"Computational Intelligence An Introduction" Segunda Edición, Andries P. Engelbrech.

"Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*", Bart Kosko.*

"AI for Game Developers", David M. Bourg, Glenn Seeman. O'Reilly,2004.

"*Fuzzy Logic Primer", Togai InfraLogic, Inc.*

"The C++ Programming Language", Bjarne Stroustrup. Addison Wesley.