Técnicas de FL aplicadas al control automático

Conjuntos difusos y FLC

CHP: 2

Nelson Acosta

INCA – UNCPBA - ARGENTINA

INvestigación en Computación Aplicada

http://www.exa.unicen.edu.ar/inca/

Email: nacosta@exa.unicen.edu.ar

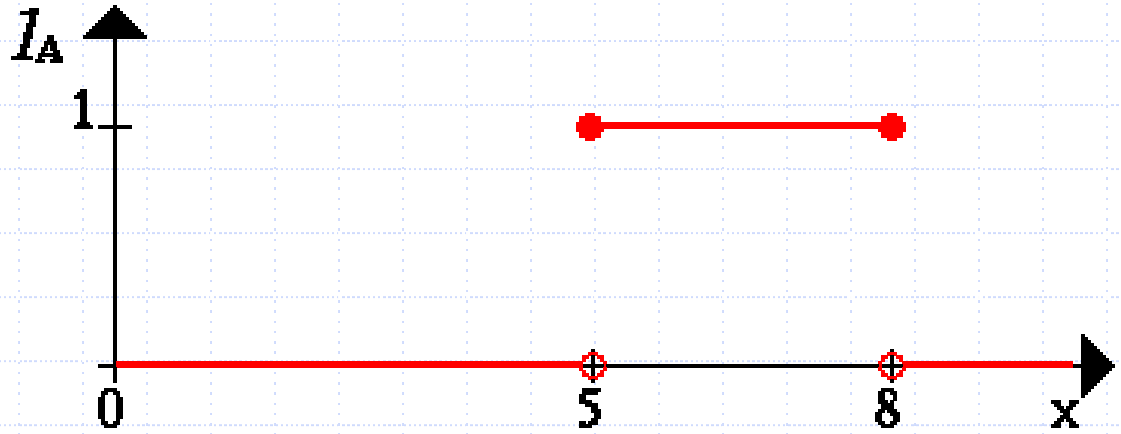
Indice del capítulo 2

- Que es un conjunto difuso.
- Operaciones de conjuntos difusos.
- Como funciona?.
- FLC: Pendulo invertido.

Qué es un conjunto difuso?

Conjuntos:

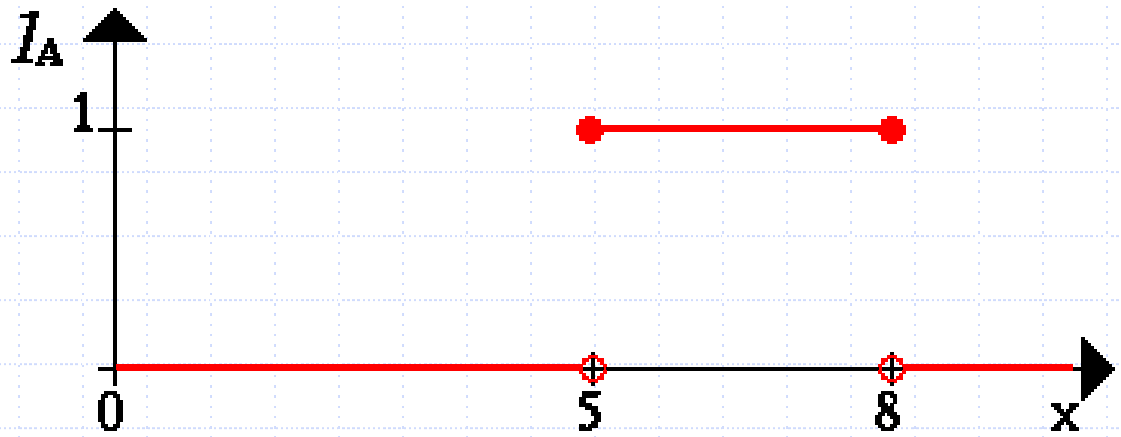
- Si consideramos el conjunto **X** de todos los números reales entre 0 y 10 (*universo de discurso*). Se define el subconjunto **A** de **X** en el rango entre 5 y 8: **A = [5,8]**.
- Se puede ver el conjunto **A** por su función característica (*que asigna 1 o 0 para cada elemento de* **X** *dependiendo si el elemento pertenece o no a* **A**).



Qué es un conjunto difuso? (2)

- Se puede interpretar que al elemento que se le asigne:
  - 1 si pertenece al conjunto **A**
  - 0 si NO pertenece al conjunto **A**

- Este concepto es **suficiente en muchas áreas** de aplicación, pero fácilmente se puede encontrar situaciones donde se **necesita más flexibilidad**.



Qué es un conjunto difuso? (3)

- Ejemplo: **Conjunto de JOVENES**.
- Formalmente:

$$\mathbf{B} = \{ \textit{set of young people} \}$$

- Universo de discurso para **edad**: comienza en 0 y termina en .... ¿ ***límite actual año 2001*** ?
- Si el límite de la juventud está en 20 años, en matemática CRISP tenemos:

$$\mathbf{B} = [0, 20]$$

- **Pregunta.** Alguien pertenece al conjunto JOVEN y el día de su vigésimo cumpleaños, ya no pertenece más al conjunto, **YA NO ES MÁS JOVEN?**.

Qué es un conjunto difuso? (4)

- Una construcción más natural para **B** puede considerar el **relajar la separación** estricta entre joven y no_joven.

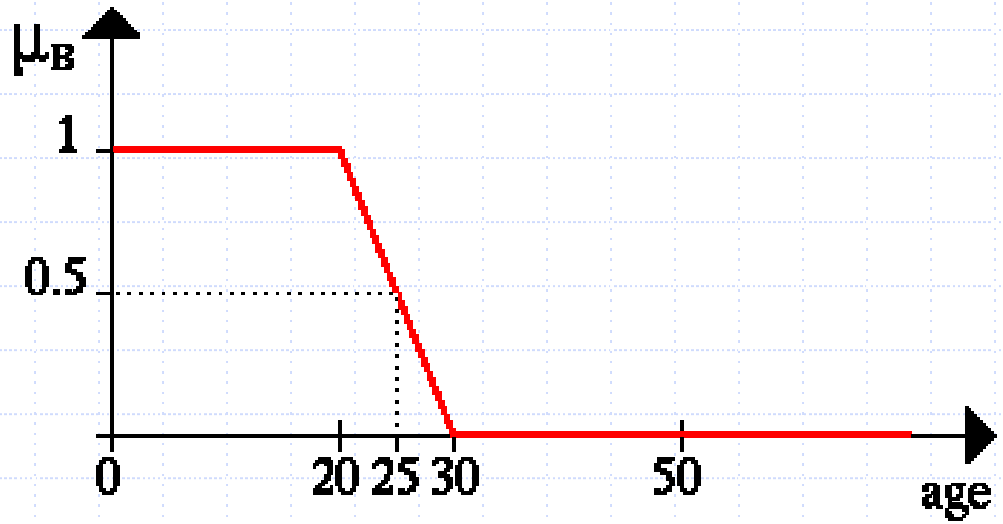
- Se puede permitir la pertenencia Si o NO al conjunto crisp. Pero se suavizará con frases del tipo: '**pertenece un poco menos a B**' o '**casi pertenece a B**' o ...

- Una forma más **generalizada** del concepto de conjunto es el permitir valores entre 0 y 1 (permite las infinitas alternativas) definidas por el intervalo:

$$\mathbf{I} = [0, 1].$$

Qué es un conjunto difuso? (5)

- Ahora la interpretacíón de números asignados a todos los elementos del universo de discurso es mucho más dificultosa.
  - 1 representa la pertenencia a **B**.
  - 0 representa la NO pertenencia a **B**.
  - Todos los demás números, representan la pertenencia gradual a **B**.
- Así la función característica es:



Nota:

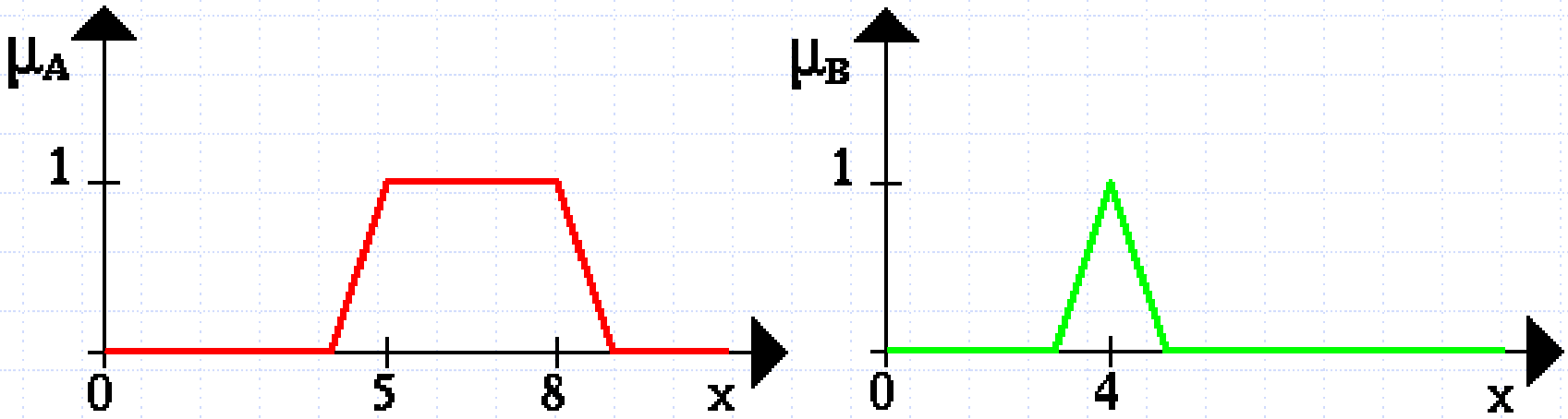
A los 25 años, le corresponde una pertenencia a joven del 50%.

Operaciones con conjunto difusos

- Al igual que las operaciones con conjuntos crisp: **intersección**, **unión** y **negación** de conjuntos.
- **Zadeh** propuso la operación **mínimo para la intersección** y el **máximo para la unión** de 2 conjuntos difusos.
- **Nótese:** los operadores (**máximo** y **mínimo**) coinciden con la lógica crisp cuando los grados de pertenencia son 0 ó 1.

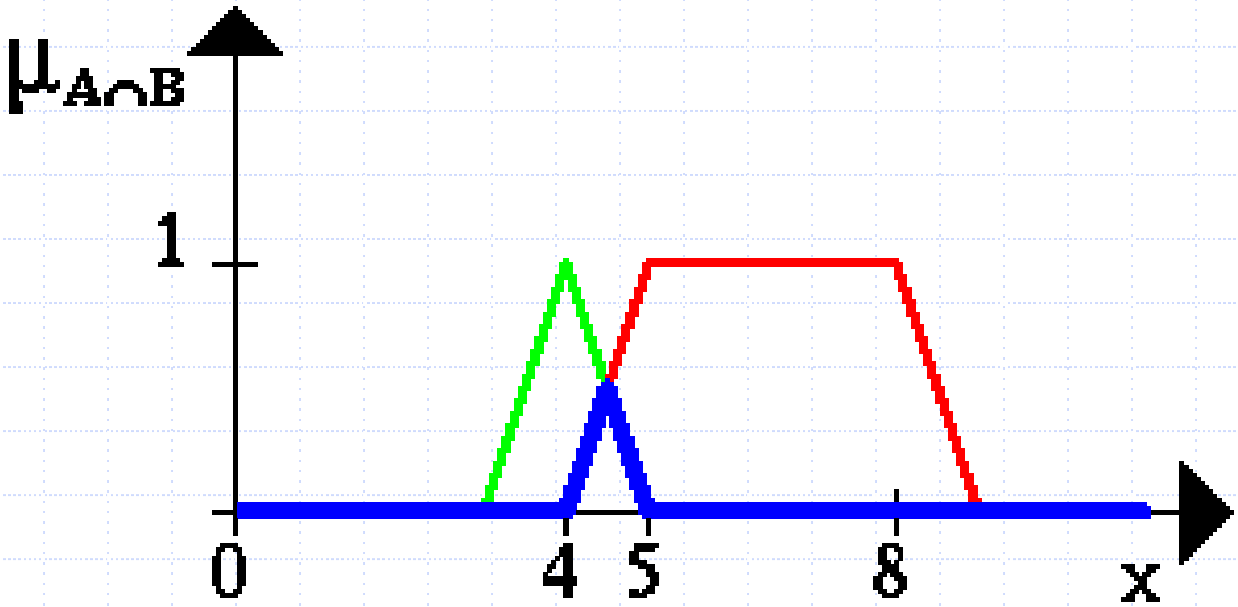
Operaciones con c.d. (2)

- **A** es un intervalo difuso entre 5 y 8.
- **B** es un intervalo difuso alrededor de 4.



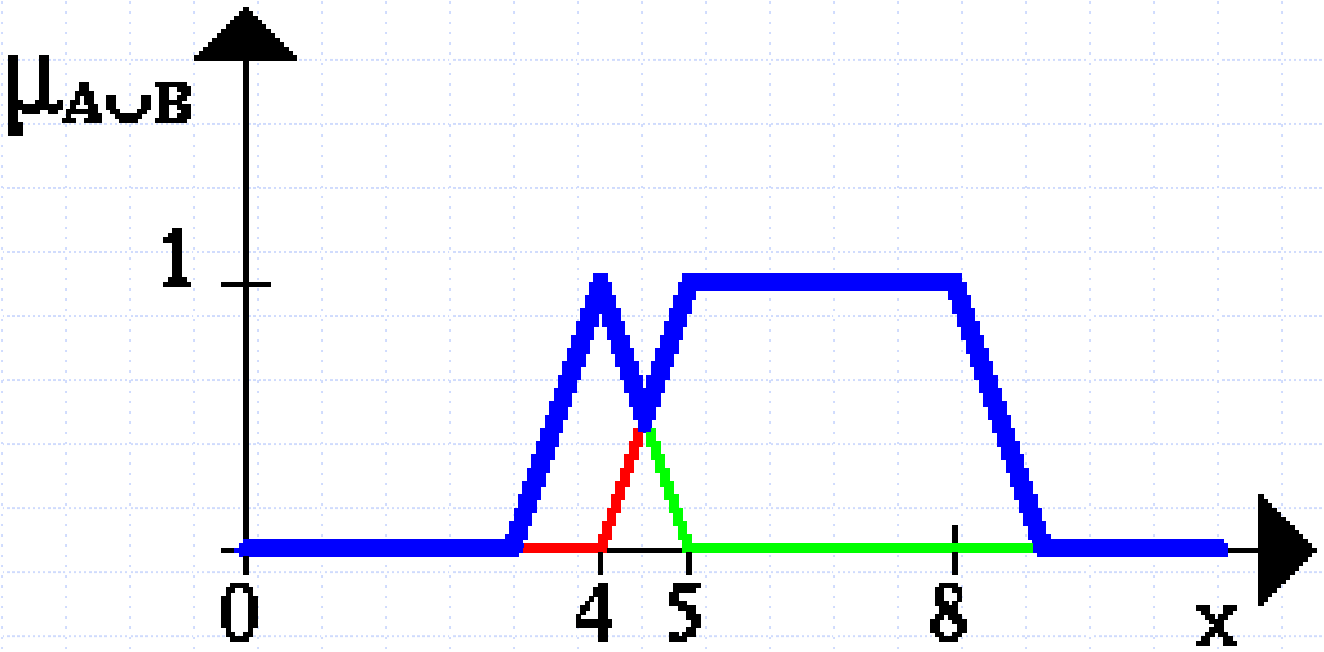
Operaciones con c.d. (3)

- La línea **AZUL** muestra la operación **AND** entre **A** y **B**.



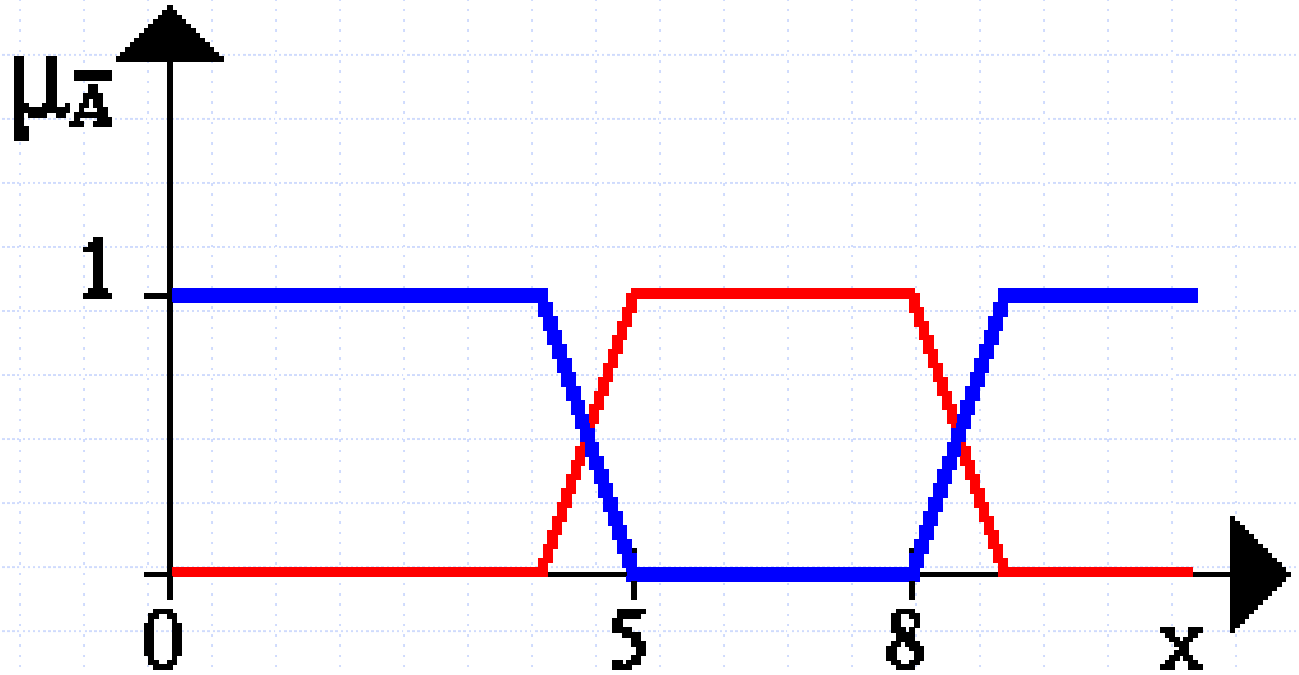
Operaciones con c.d. (4)

- La línea **AZUL** muestra la operación **OR** entre **A** y **B**.



Operaciones con c.d. (5)

- La línea **AZUL** muestra la operación **NOT** (o negación) entre **A** y **B**.



¿ Cómo funciona ?

- Para demostrar como funciona, se presenta un ejemplo: **el control simple de un termostato controlando un calentador**.
- El controlador tiene como entrada la **temperatura de la habitación** y como la salida el ajuste de la **velocidad del calentador**.

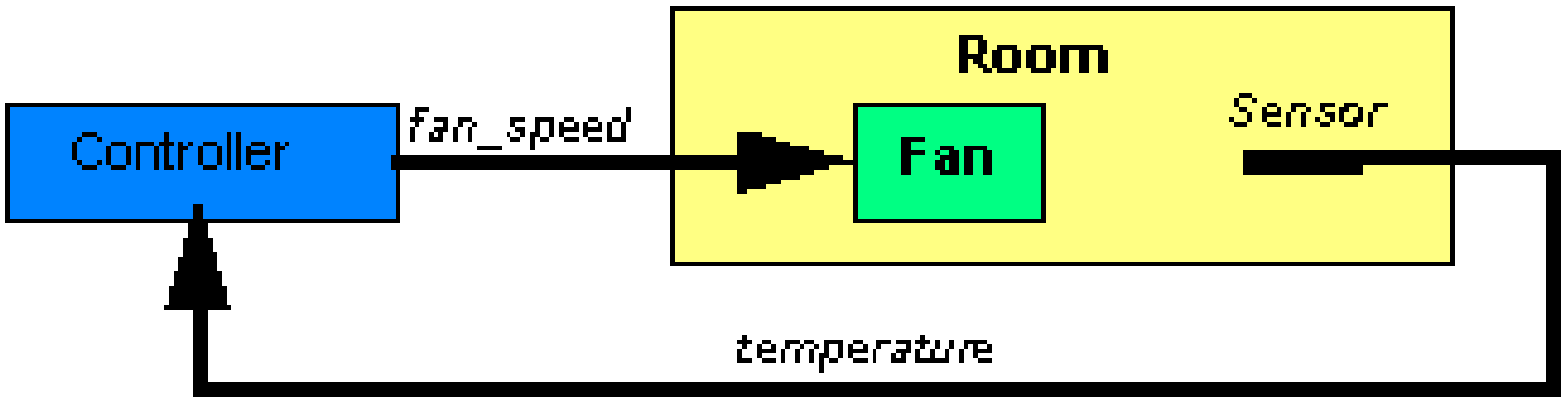


Figure 1: A Simple Temperature Controller

¿ Cómo funciona ? (2)

- Un termostato convencional trabaja como una llave de conexión **ON-OFF**.
- Si lo seteamos a **78°F** el calentador es activado sólo cuando la temperatura cae debajo de los **75°F**; mientras que se apaga cuando alcanza **81°F**.
- **Como resultado**, la temperatura de la habitación resulta **DEMASIADO CALIENTE** o **FRIA.**

¿ Cómo funciona ? (3)

- Un **termostato difuso** trabaja en 'las sombras' o 'en tonalidades', donde la temperatura es tratada como una serie de rangos que se superponen.
- Por ejemplo: 78°F es 60% cálido y 20% caliente.
- El controlador es programado con simples reglas **IF-THEN** de tal forma que el calentador funcione gradualmente.
- **Como resultado**, cuando la temperatura cambia, la velocidad se irá ajustando gradualmente para mantener la temperatura al nivel deseado.

¿ Cómo funciona ? (4)

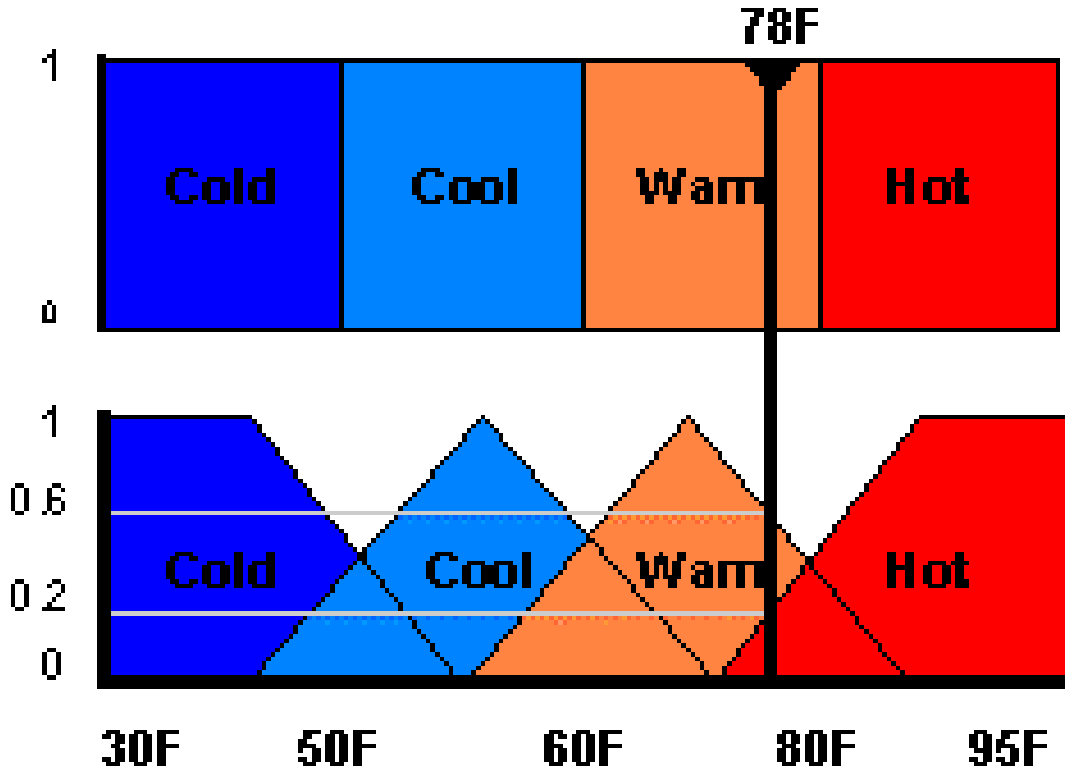


Figure2: Conventionnal and Fuzzy Sets

¿ Cómo funciona ? (5)

- **Caracterizar** los rangos de valores para las variables de entrada y salida del FLC.
- **Asignar rótulos** tales como: *cool* para la temperatura o *high* para el calentador; de tal forma de poder escribir las reglas. Estas variables lingüísticas se refieren a los conjuntos difusos que se superponen (llamadas como funciones de pertenencia).
- Todas las acciones del FLC dependerán de cómo la actual temperatura cae en esos rangos y en cómo las reglas describen el comportamiento del FLC. **La salida del FLC varía continuamente para ajustar la velocidad del calentador**.

¿ Cómo funciona ? (6)

- EL FLC para la temperatura puede ser descrito en 4 reglas:

```
IF temperature IS cold THEN
        fan_speed IS high
IF temperature IS cool THEN
        fan_speed IS medium
IF temperature IS warm THEN
        fan_speed IS low
IF temperature IS hot THEN
        fan_speed IS zero
```

- La salida del FLC **varía continuamente** para **ajustar la velocidad del calentador**.

¿ Cómo funciona ? (7)

- Un FLC funciona de manera similar a un sistema convencional: acepta valores de entrada, hace algunos cálculos y genera una salida.

- Este proceso es llamado **Inferencia Difusa** y funciona en 3 pasos:
  - **Fusificación** o traducción a valores difusos.
  - **Evaluación de Reglas**.
  - **Defusificación** o traducción a valores crisp.

¿ Cómo funciona ? (8)

- **Fusificación**. El valor de temperatura entrado (78°F) es traducido de acuerdo a las funciones de pertenencia:
  - `cold(78°F) = 0`
  - `cool(78°F) = 0`
  - `warm(78°F) = 0.6`
  - `hot( 78°F) = 0.2`

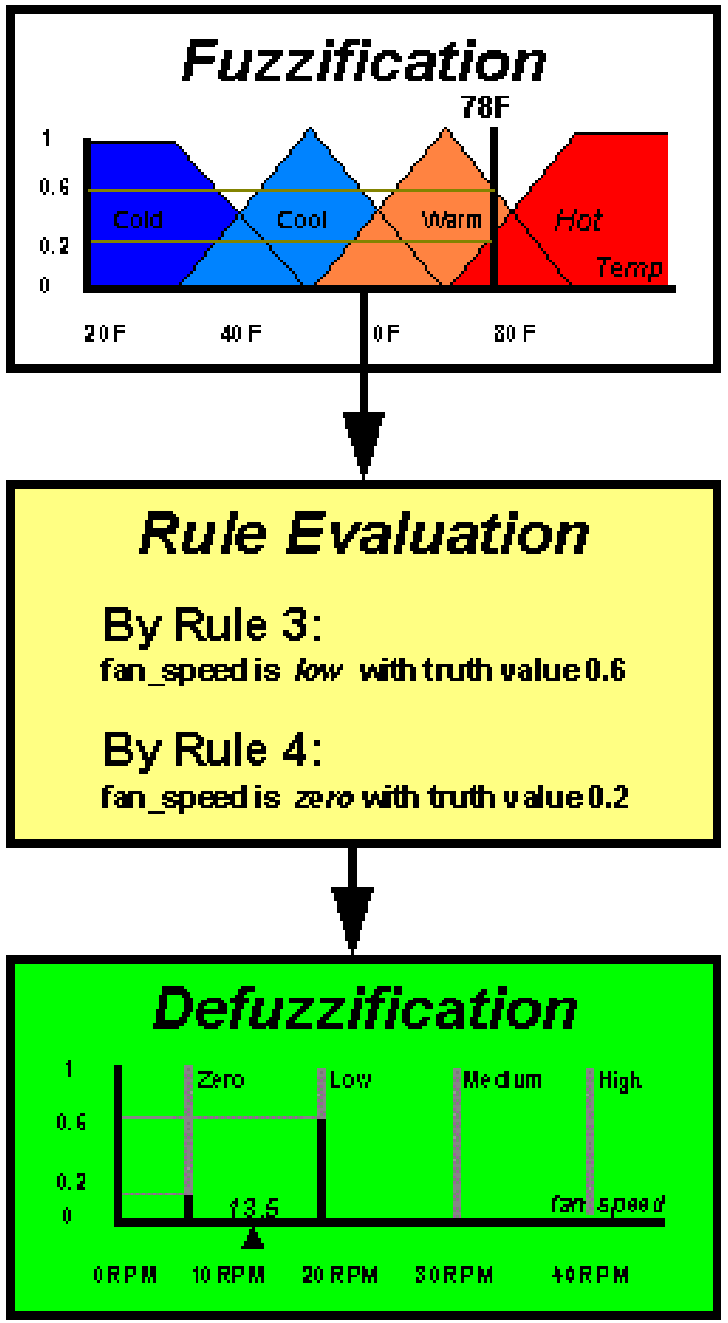


Figure 3 The Fuzzy Inference Process

¿ Como funciona ? (9)

- Durante la **evaluación de reglas**, todas las reglas son evaluadas y algunas son ***fired***. Para **78°F** sólo las **2** reglas últimas son activadas.

```
Regla 3, velocidad low = 60%.
Regla 4, velocidad zero = 20%.
```

- Durante la **defusificación** el **60%** y el **20%** son combinados por el método del **COG** para producir una salida **CRISP** de 13.5 RPM para el calentador.

¿ Como funciona ? (10)

Objetivo:

- FLC simplifica la implementación ?
- Hasta aquí se vió como implementar un FLC de **una sola variable**. En la naturaleza hay sistemas mucho más complejos: con más variables, con gran cantidad de parámetros, muchas variables controladas, etc.
- **EJEMPLO:** Se redefine el FLC para que entre también el valor de humedad.

¿ Como funciona ? (11)

```
IF temperature IS cold AND humidity IS high THEN fan_spd IS high
IF temperature IS cool AND humidity IS high THEN fan_spd IS medium
IF temperature IS warm AND humidity IS high THEN fan_spd IS low
IF temperature IS hot  AND humidity IS high THEN fan_spd IS zero
IF temperature IS cold AND humidity IS  med THEN fan_spd IS medium
IF temperature IS cool AND humidity IS  med THEN fan_spd IS low
IF temperature IS warm AND humidity IS  med THEN fan_spd IS zero
IF temperature IS hot  AND humidity IS  med THEN fan_spd IS zero
IF temperature IS cold AND humidity IS  low THEN fan_spd IS medium
IF temperature IS cool AND humidity IS  low THEN fan_spd IS low
IF temperature IS warm AND humidity IS  low THEN fan_spd IS zero
IF temperature IS hot  AND humidity IS  low THEN fan_spd IS zero
```

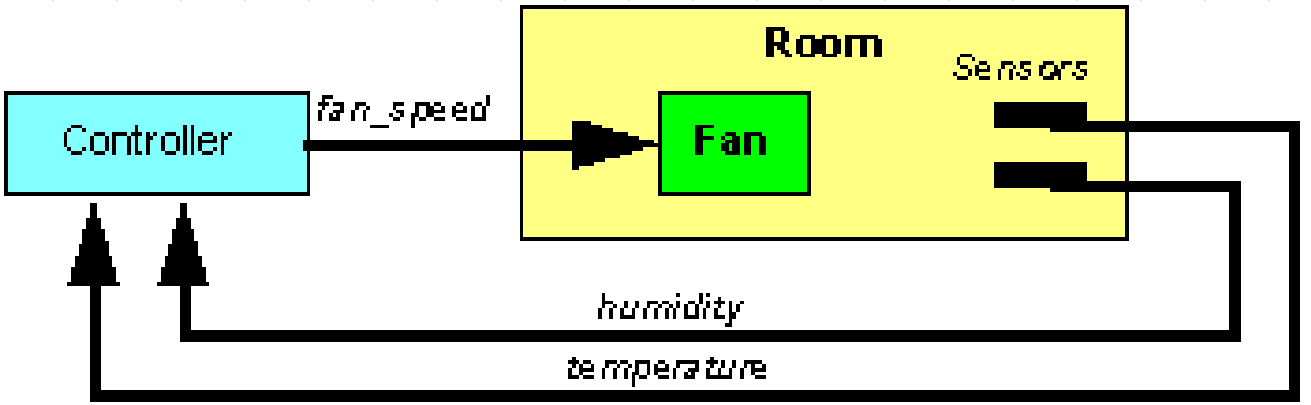


Figure 7: Modified Temperature Controller

¿ Como funciona ? (12)

- Un aproximador lineal requiere manejar separadamente cada entrada, entonces multiplica el esfuerzo de diseño.
- Un aproximador por segmentos requiere varios controladores y su costo se incrementa mucho,.
- Una LUT parece más apropiada pero toma mucho tiempo de desarrollo, debug y ajuste (o puesta a punto). Si cada entrada de 8 bits, la LUT debe tener 64KBytes.

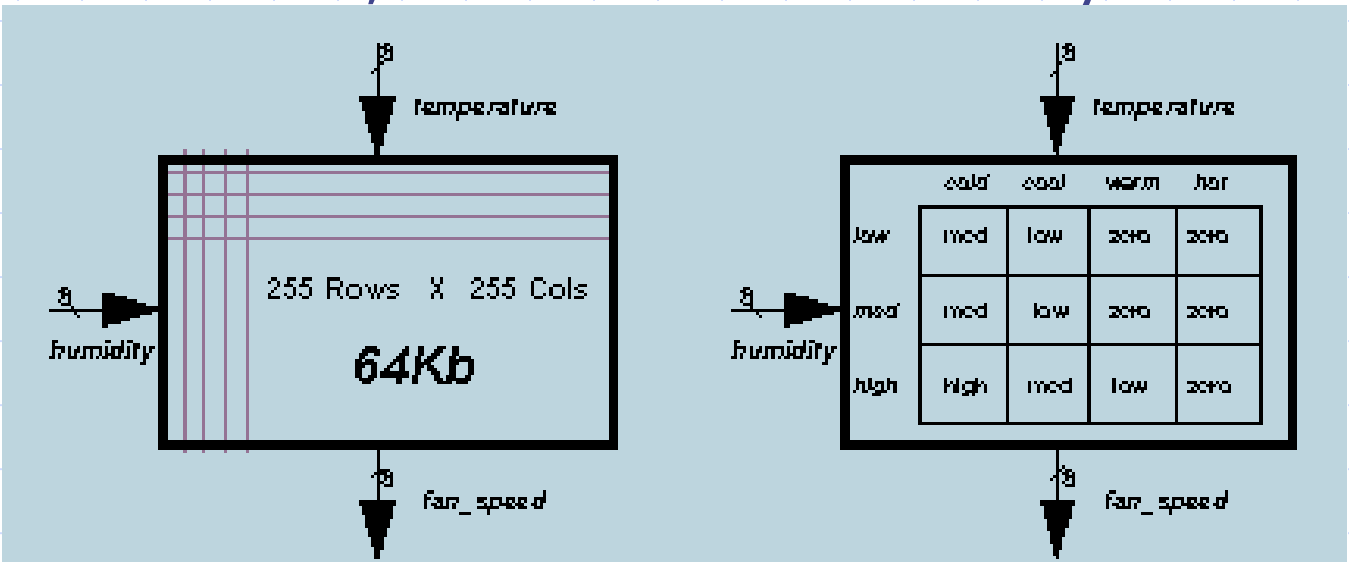


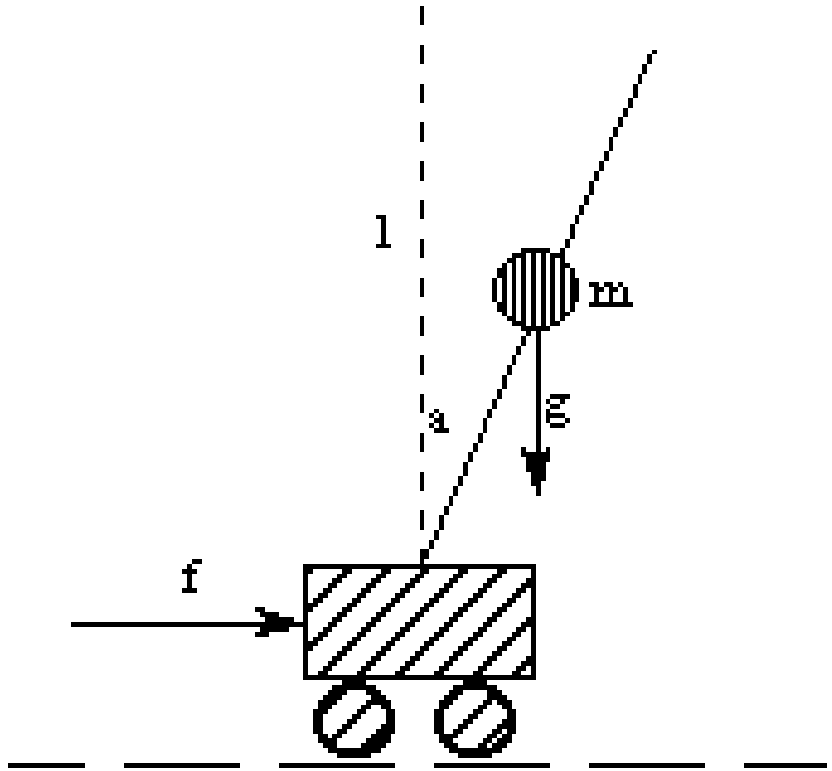
Figure 8: Lookup table versus rules and membership functions

¿ Como funciona ? (13)

- Con FL se puede describir la salida como función de 2 o más entradas, (con operadores AND).
- El **enfoque difuso** requiere menos entradas que una LUT (dependiendo del número de rótulos para cada variable de entrada).
- Las **reglas** son mucho más fáciles de desarrollar, depurar y poner a punto que la LUT.

FLC: Péndulo Invertido

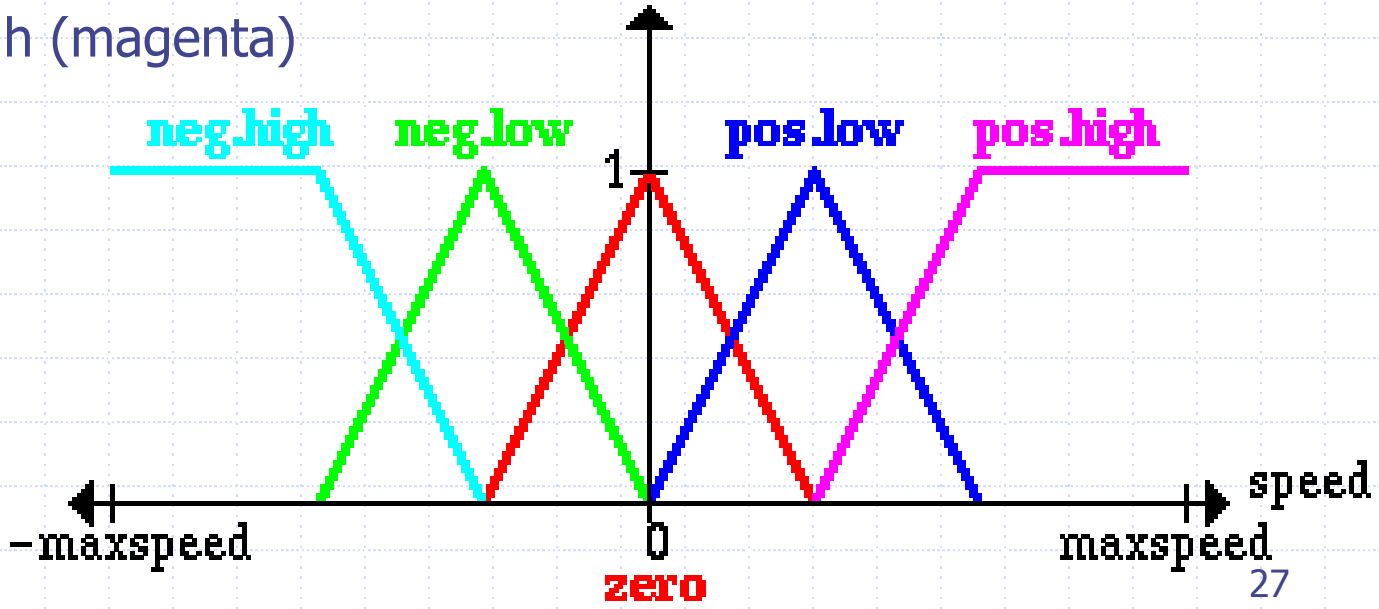
El problema es **balancear** un **péndulo** sobre una plataforma móvil que puede moverse sólo en 2 direcciones (a izquierda o a derecha).



FLC: Péndulo Invertido (2)

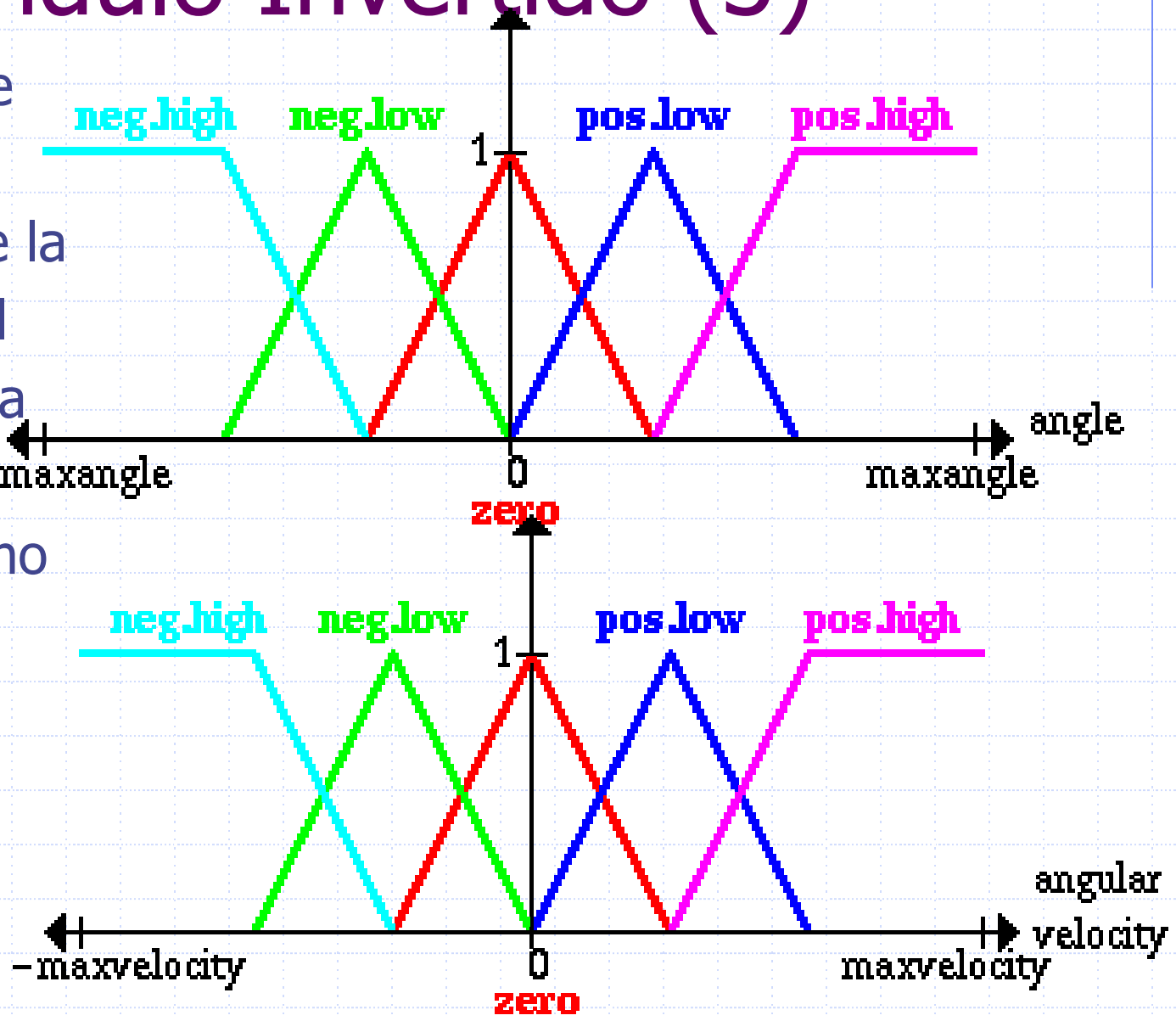
Primero hay que definir (**subjetivamente**) los conjuntos de velocidad: high, low, ...; así se especifican las funciones de pertenencia para los conjuntos difusos.

- negative high (cyan)
- negative low (green)
- zero (red)
- positive low (blue)
- positive high (magenta)



FLC: Péndulo Invertido (3)

Lo mismo debe realizarse para el ángulo entre la plataforma y el péndulo; y para la velocidad angular de dicho ángulo.



FLC: Péndulo Invertido (4)

Para simplificar el problema se supone que:

- **Inicialmente** el péndulo está muy cerca de la posición central (ángulo 0) y no se mueve (velocidad angular 0).
- Así el **mayor ángulo** del péndulo puede ser de 45 grados en cualquier dirección.
- En la situación deseada (**inicial**) no hay nada que hacer (velocidad 0).

FLC: Péndulo Invertido (5)

Para definir el comportamiento (REGLAS):

- Si no hay nada que hacer (velocidad 0).
- Si el péndulo está en ángulo 0, pero moviendose lentamente en dirección positiva.
- Naturalmente, se debe compensar el movimiento del péndulo, moviendo la plataforma en la misma dirección que la pequeña velocidad (del péndulo).

Así, ya tenemos 2 reglas:

```
IF (angle=zero)&(angular_velocity=zero) THEN
                    speed=zero.
IF (angle=zero)&(angular_velocity=pos.low) THEN
                    speed=pos.low
```

FLC: Péndulo Invertido (6)

REGLAS:

		angle				
	speed	NH	NL	Z	PL	PH
	v	NH		NH		
	e	NL		NL	Z	
	l	Z	NH	NL	Z	PL	PH
	o	PL		Z	PL	
	c	PH			PH	

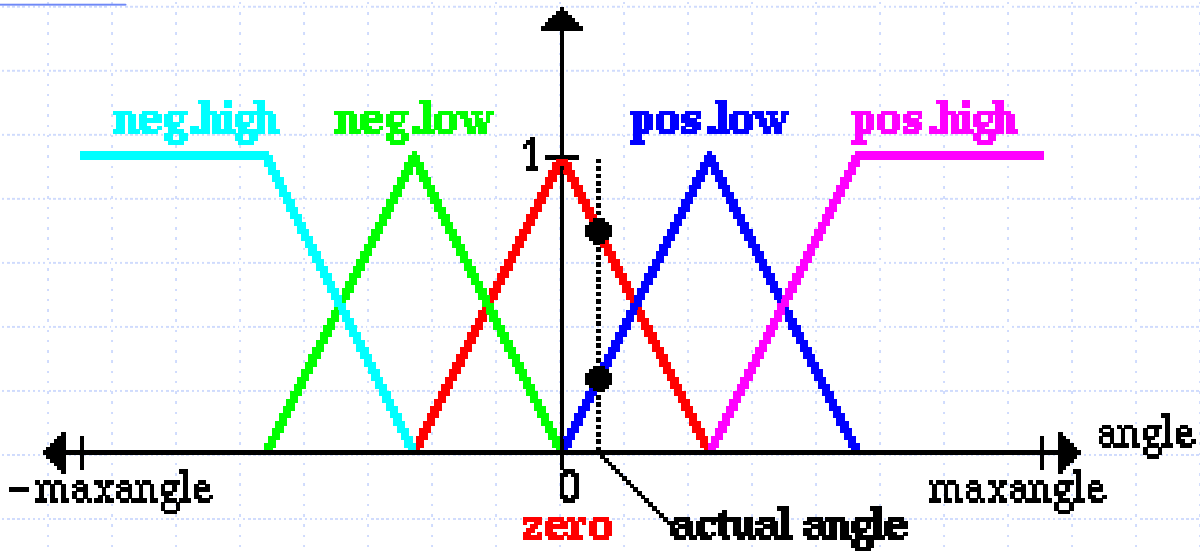
donde:

- NH es la (usual) abreviación para negative high,
- NL para negative low etc.

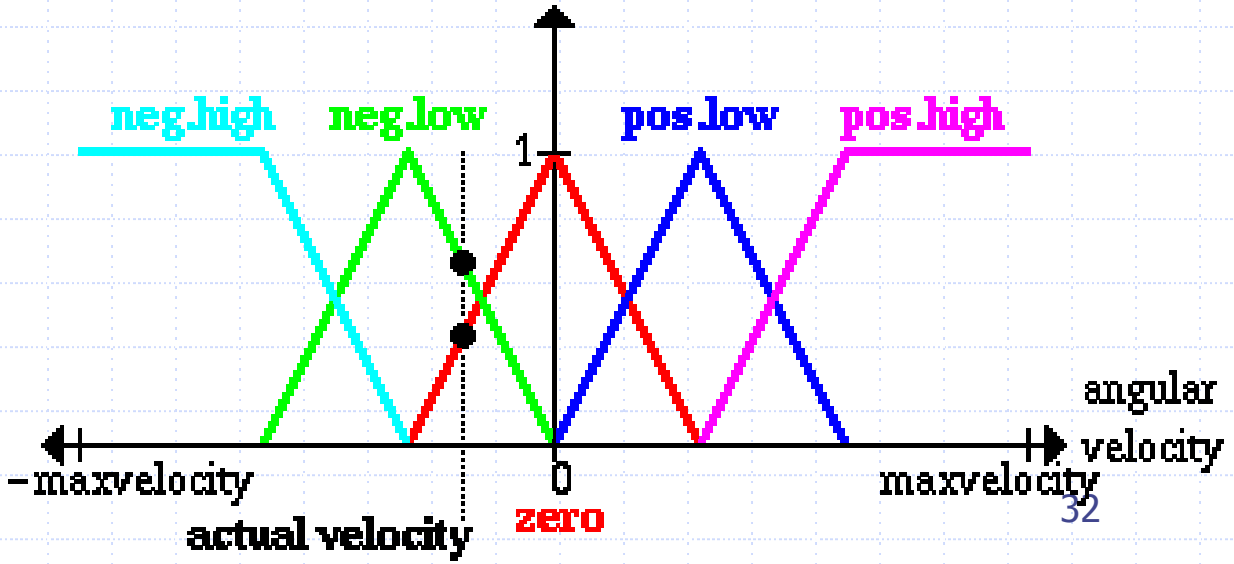
FLC: Péndulo Invertido (7)

Se definen 2 valores explícitos para angle y angular_velocity:

angle



angular_velocity

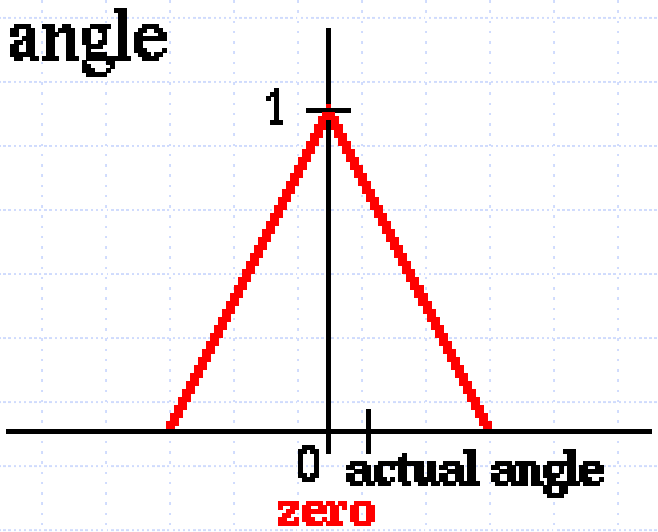


FLC: Péndulo Invertido (8)

Si aplicamos la regla:

IF (angle=zero and angular_velocity=zero) THEN speed=zero

a los valores seleccionados,

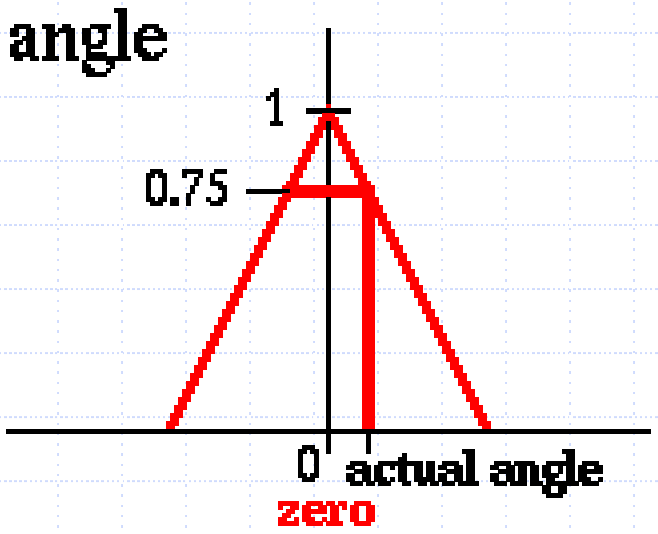


This is the linguistic variable "angle" where we zoom in on the fuzzy set "zero" and the actual angle.

FLC: Péndulo Invertido (9)

IF (angle=zero and angular_velocity=zero) THEN speed=zero

,

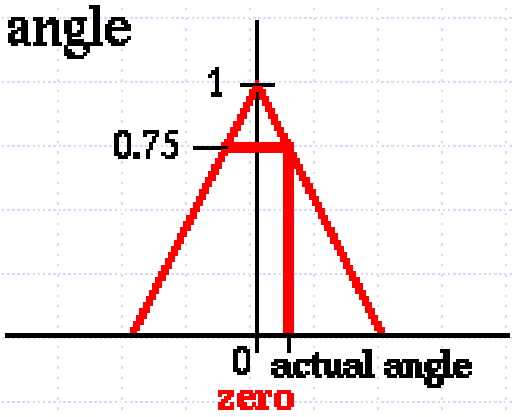


We realize that our actual value belongs to the fuzzy set "zero" to a degree of 0.75.

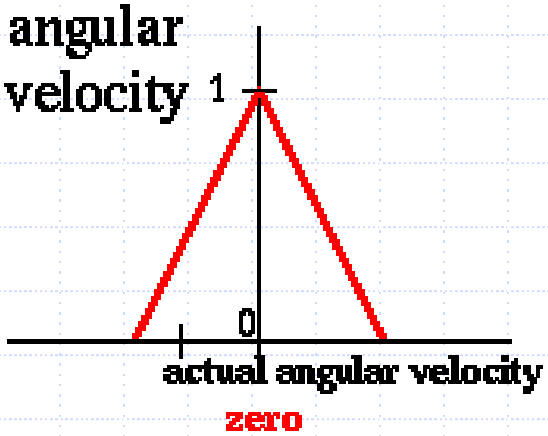
FLC: Péndulo Invertido (10)

IF (angle=zero and angular_velocity=zero) THEN speed=zero

,



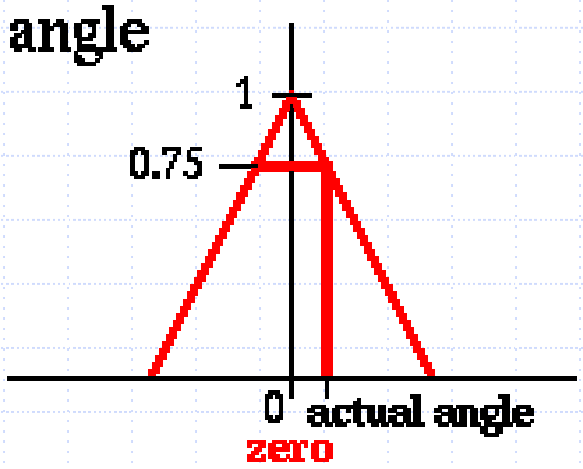
We realize that our actual value belongs to the fuzzy set "zero" to a degree of 0.75.



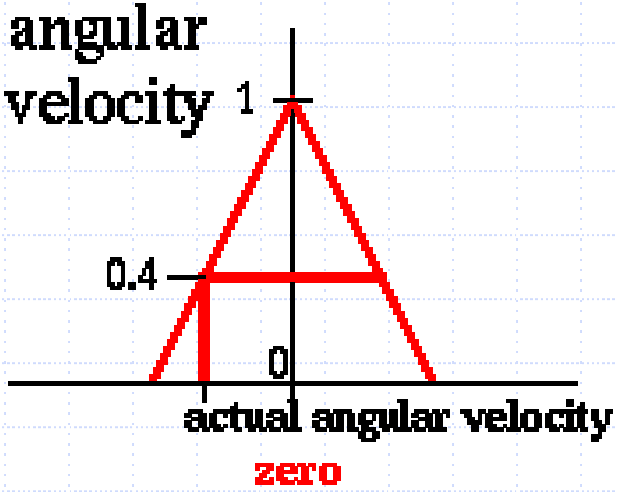
This is the linguistic variable "angular velocity" where we zoom in on the fuzzy set "zero" and the actual angular velocity.

FLC: Péndulo Invertido (11)

IF (angle=zero and angular_velocity=zero) THEN speed=zero



We realize that our actual value belongs to the fuzzy set "zero" to a degree of 0.75.



We realize that our actual value belongs to the fuzzy set "zero" to a degree of 0.4.

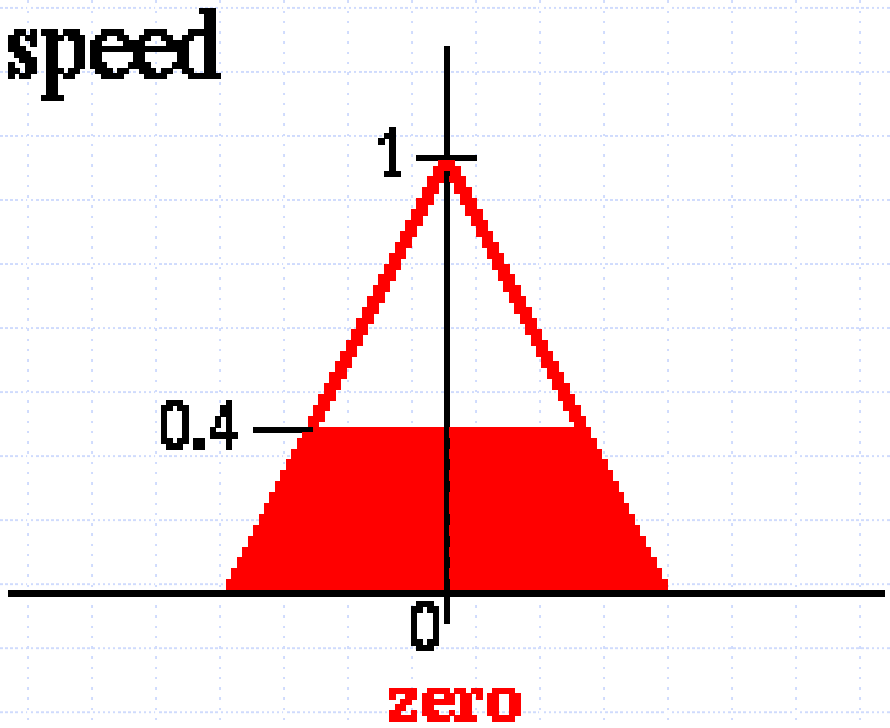
FLC: Péndulo Invertido (12)

- Sólo 4 reglas se activan, y las unimos en un único resultado.

IF (angle=zero and angular_velocity=zero) THEN speed=zero

- así:

,



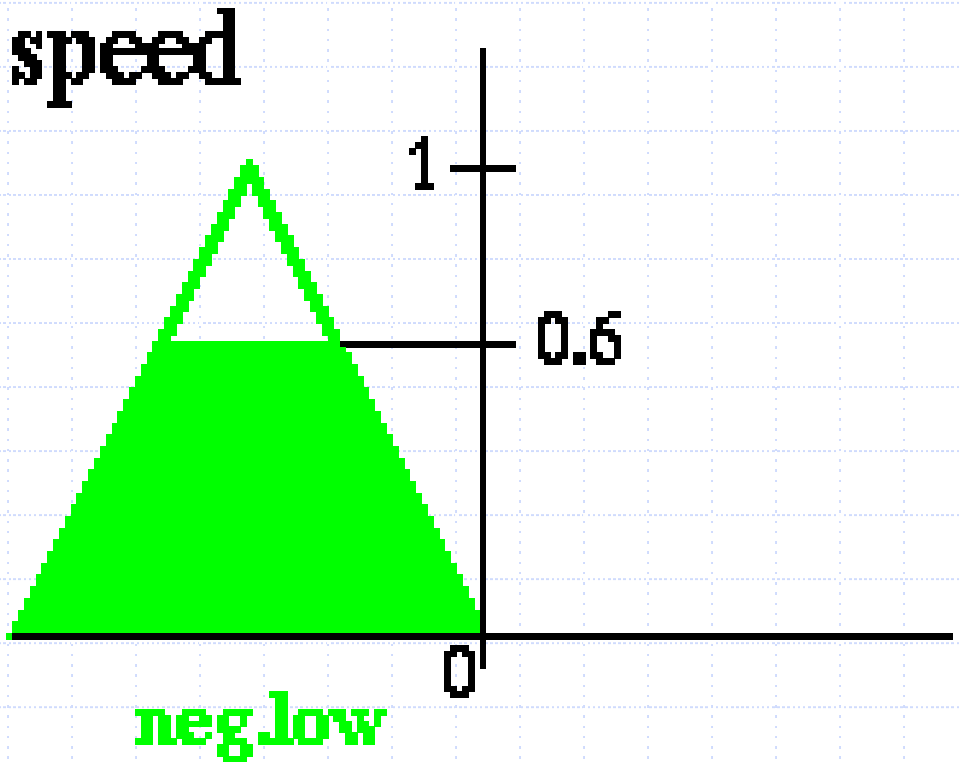
FLC: Péndulo Invertido (13)

- Sólo 4 reglas se activan, y las unimos en un único resultado.

  IF (angle=zero and angular_velocity=NL) THEN speed=NL

- así:

,



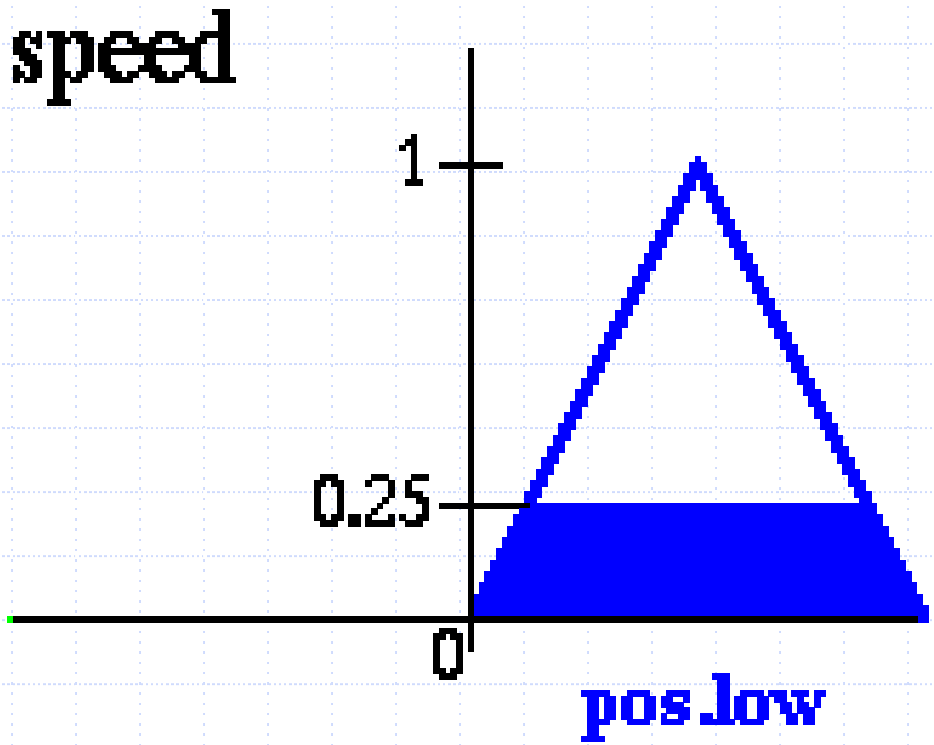
FLC: Péndulo Invertido (14)

- Sólo 4 reglas se activan, y las unimos en un único resultado.

IF (angle=PL and angular_velocity=zero) THEN speed=PL

así:

,



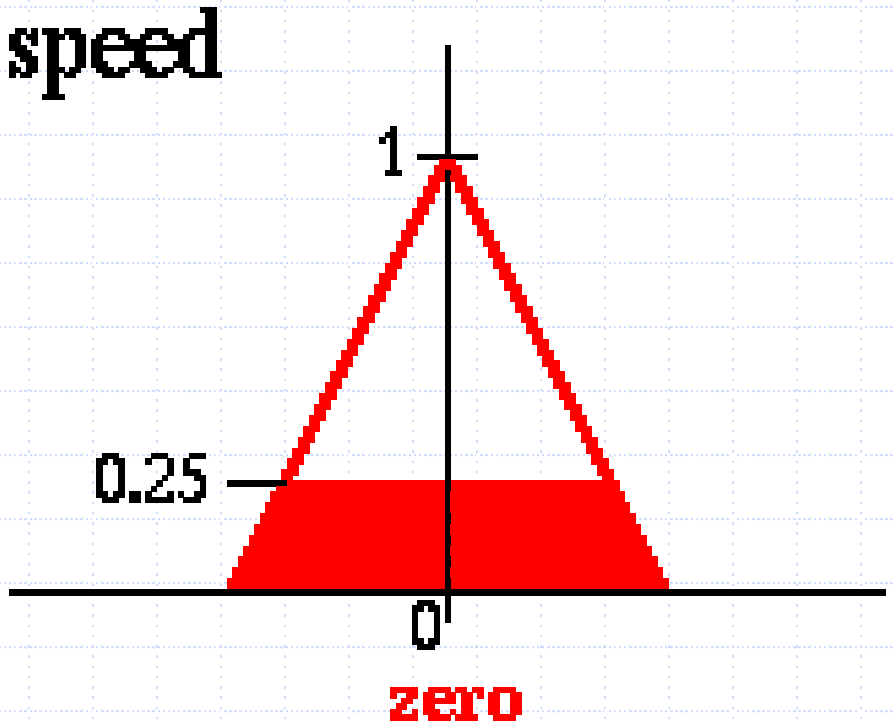
FLC: Péndulo Invertido (15)

- Sólo 4 reglas se activan, y las unimos en un único resultado.

  IF (angle=PL and angular_velocity=NL) THEN speed=zero

- así:

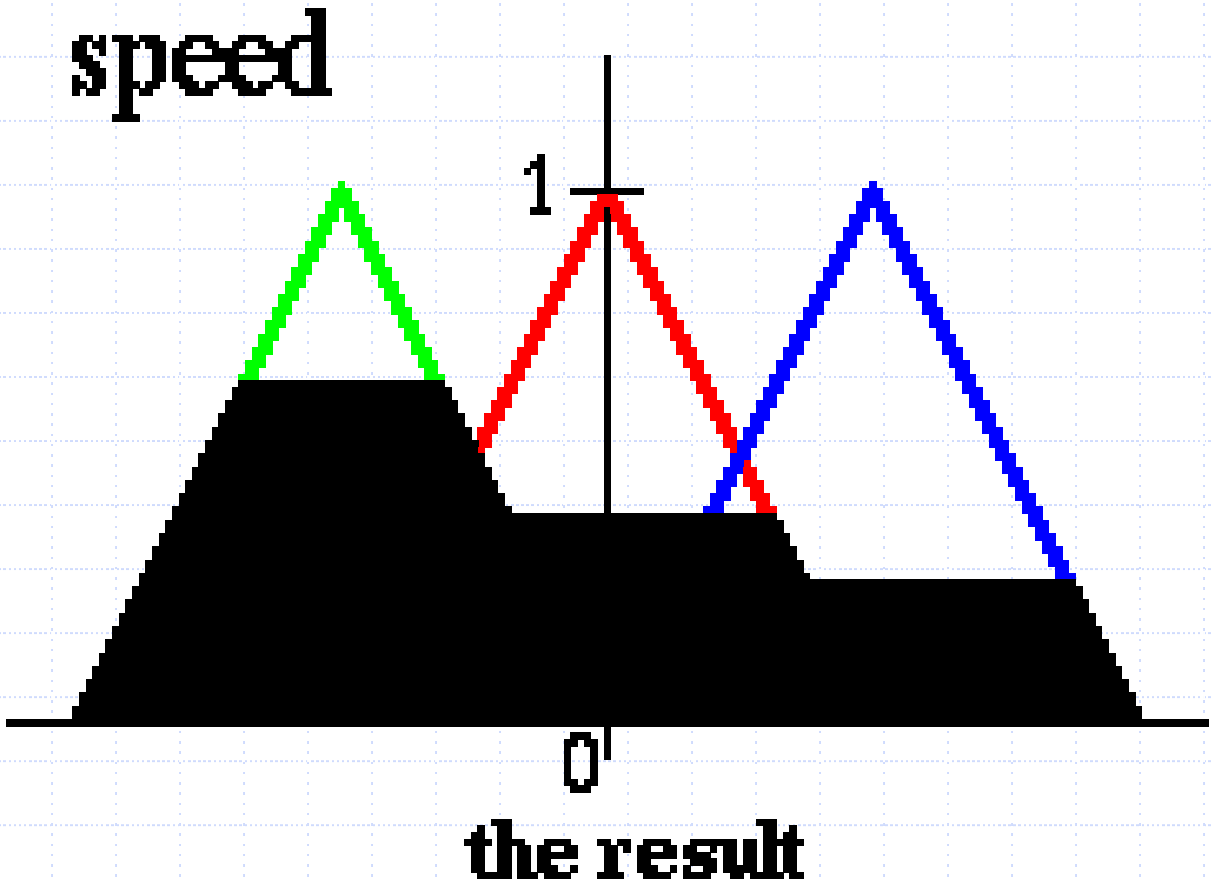
,



FLC: Péndulo Invertido (16)

- Las 4 reglas unidas:

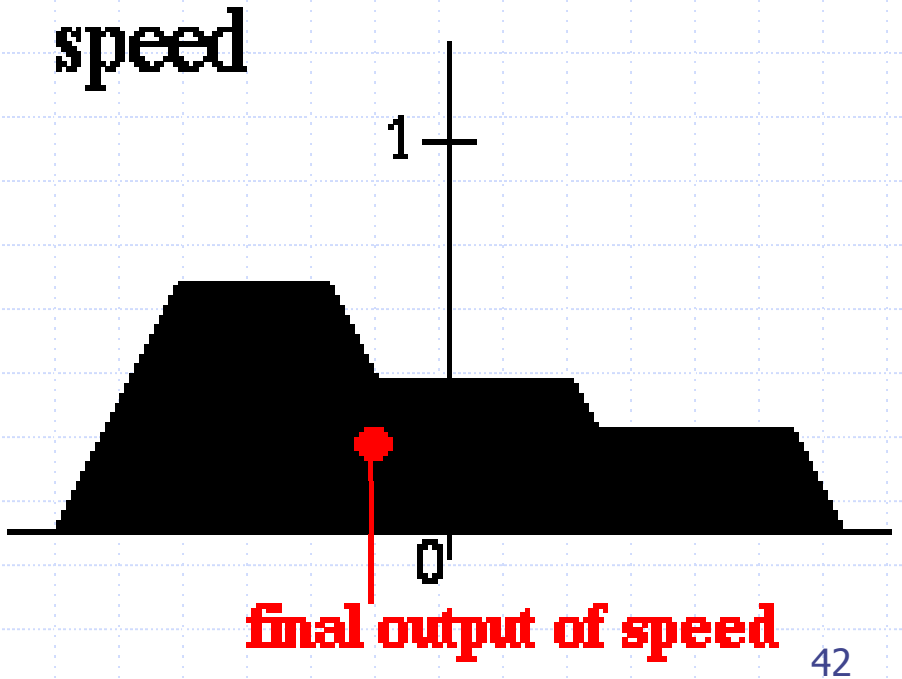
,



FLC: Péndulo Invertido (17)

- El resultado del FLC es un conjunto difuso de velocidad, así se debe elegir el valor representativo para la salida.
- Por el centro de gravedad es:

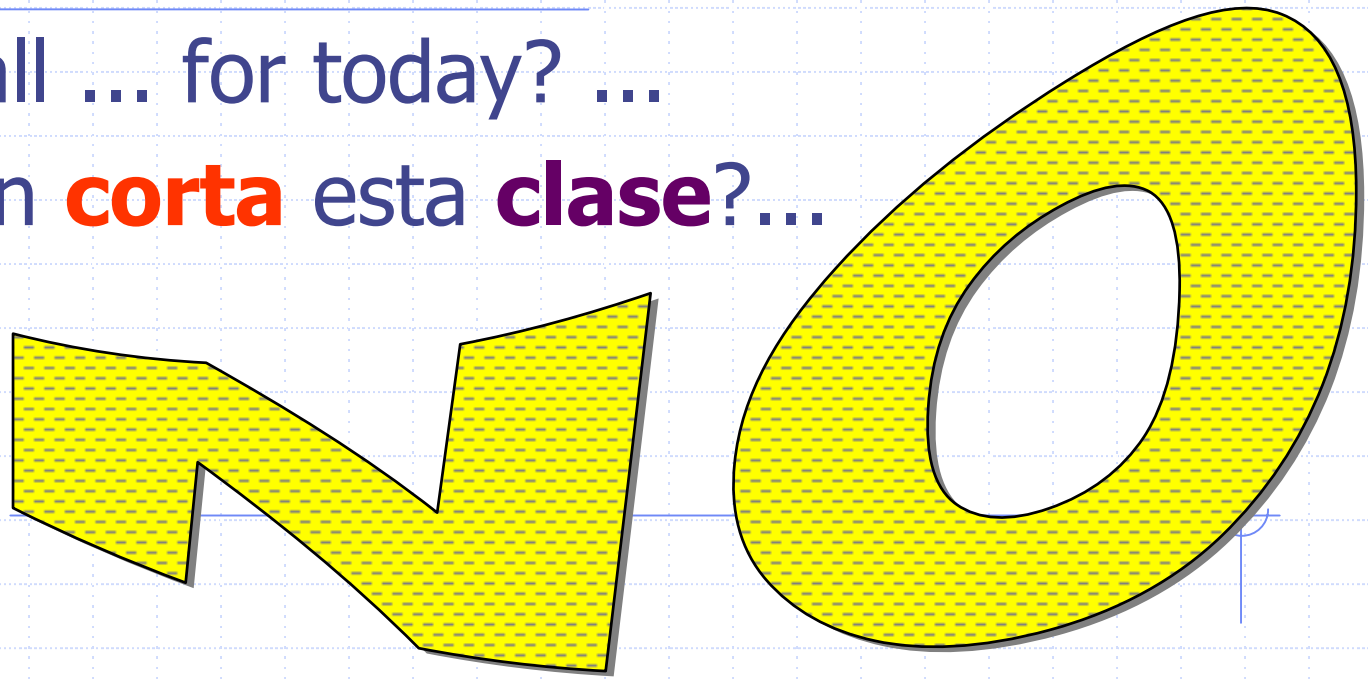
Procedimiento utilizado para controladores 'Mamdani'



The End

That's all ... for today? ...

Salió tan **corta** esta **clase**?...



Técnicas de FL aplicadas al control automático

Partes de un FLC

CHP: 2' ... El (RE)greso

Nelson Acosta

INCA – UNCPBA - ARGENTINA

INvestigación en Computación Aplicada

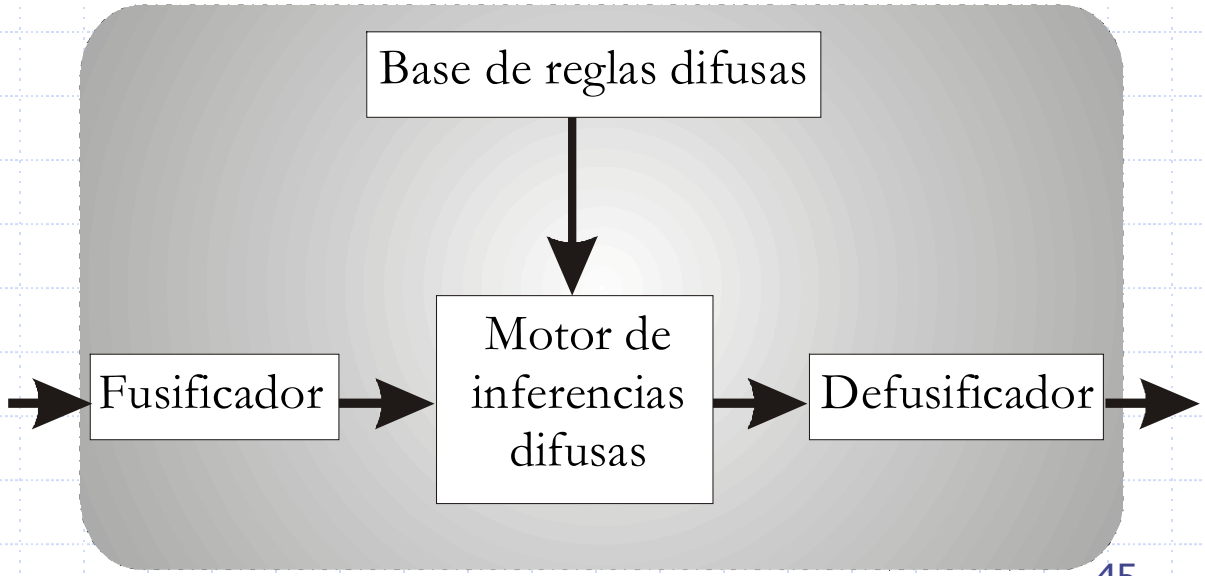
http://www.exa.unicen.edu.ar/inca/

Email: nacosta@exa.unicen.edu.ar

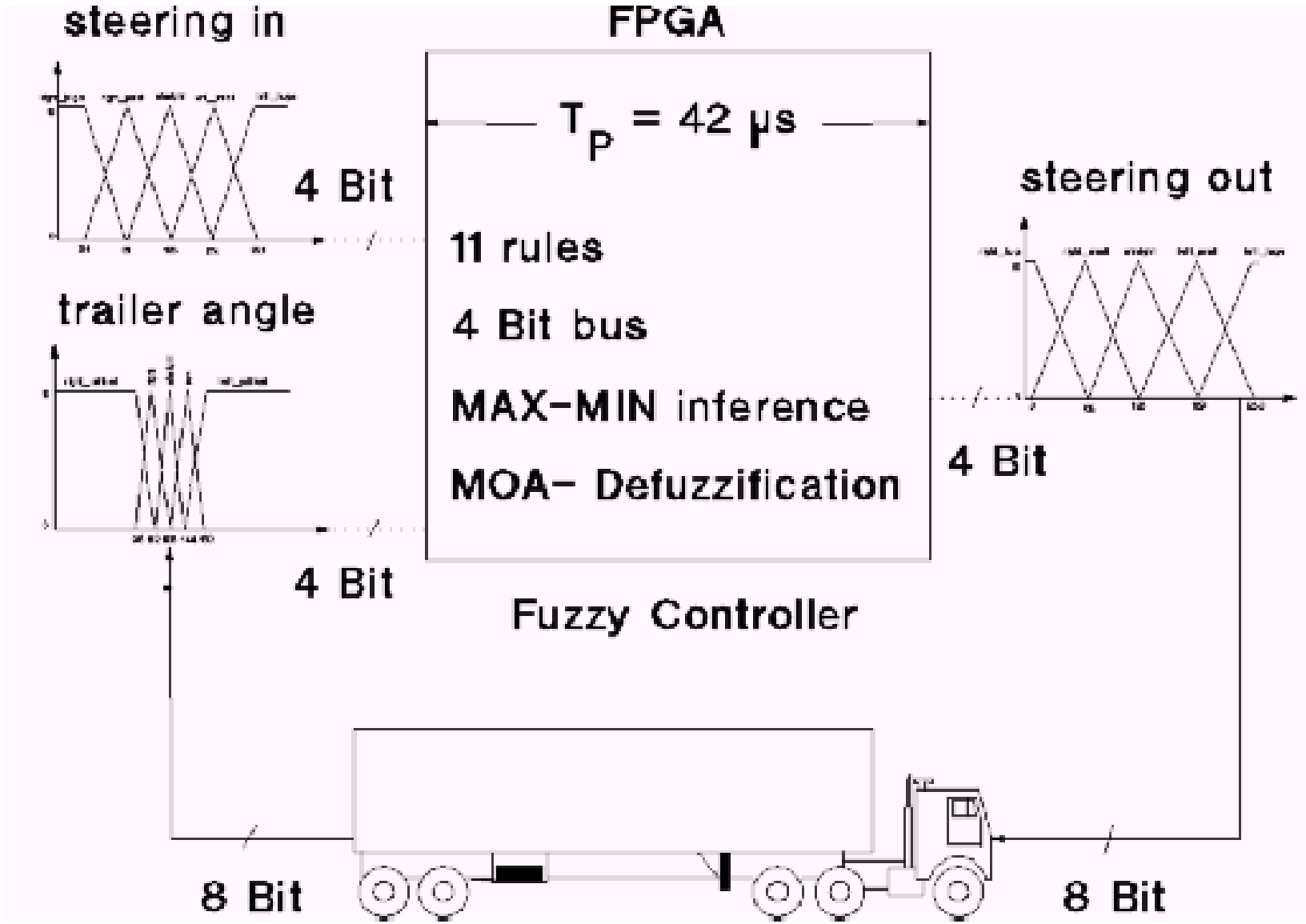
Lógica Difusa

- **REGLAS:** definen el comportamiento del controlador.
- **MOTOR DE INFERENCIAS:** realiza el cálculo o aplicación de las reglas.
- **FUSIFICADOR:** traduce a valores difusos los valores de las variables de entrada.
- **DEFUSIFICADOR:** traduce el valor difuso de salida a su valor *crisp* de salida.

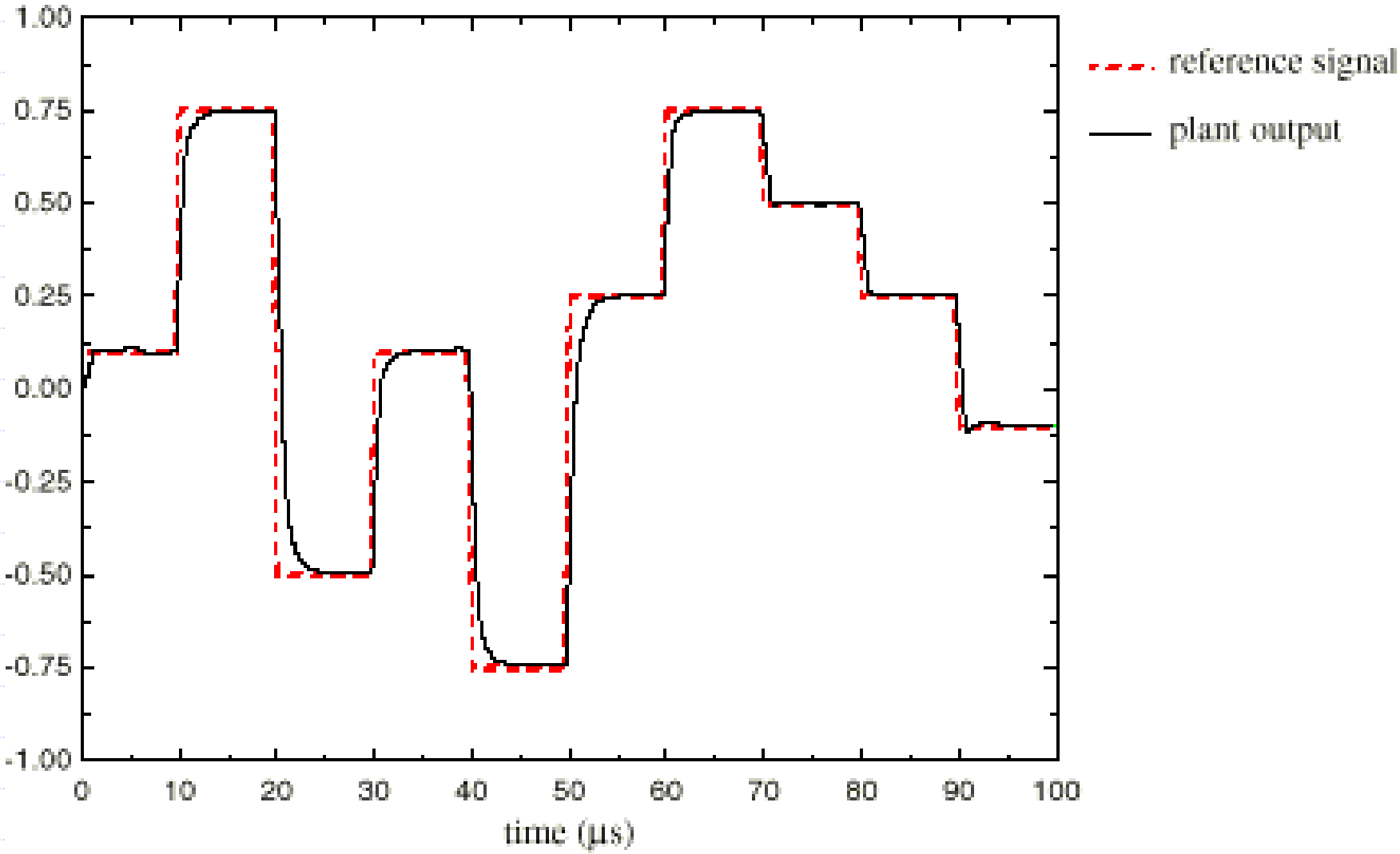
Modelo difuso de '**Mamdani**'.



Lógica Difusa (2)



Lógica Difusa (3)



Lógica Difusa (4)

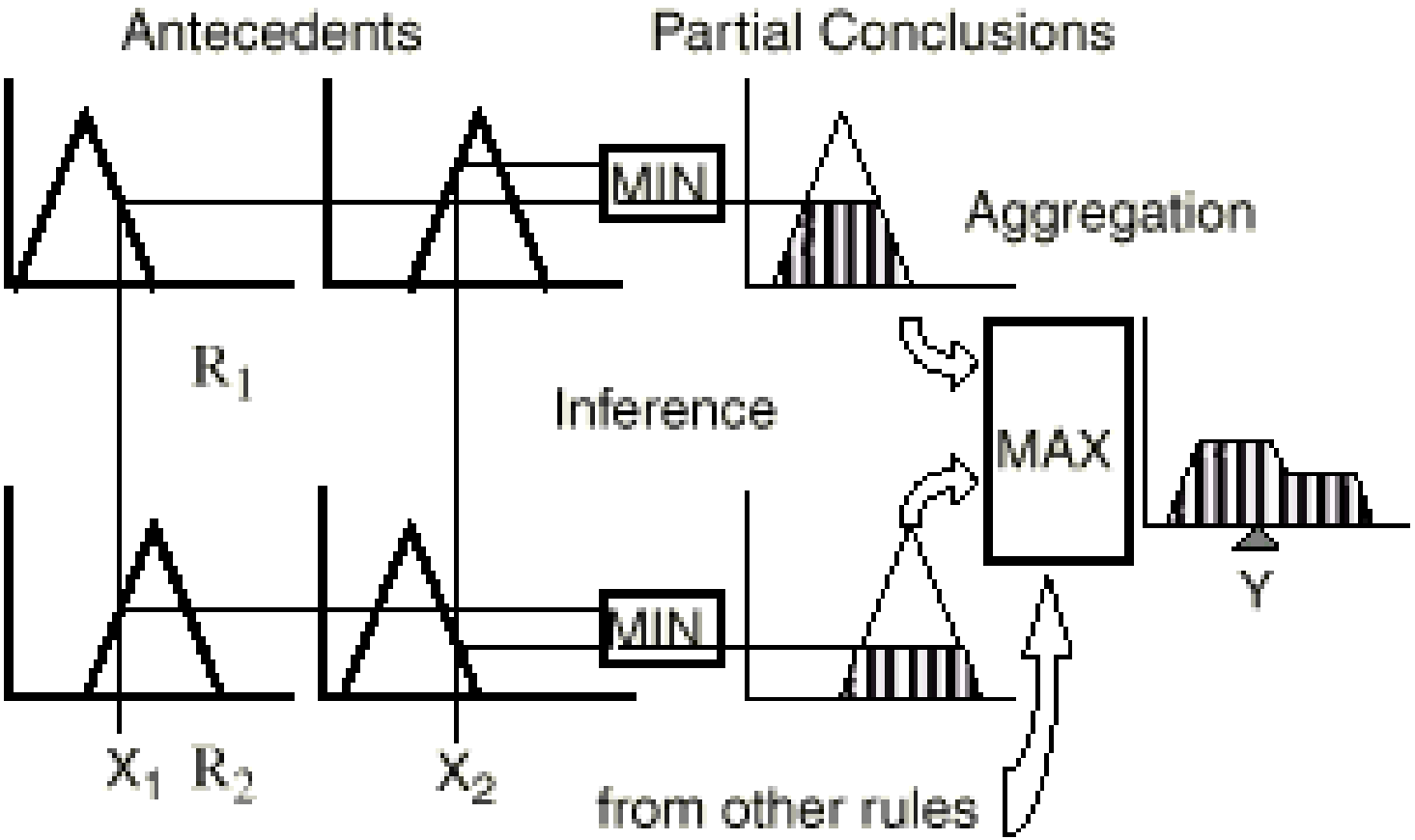


Figure 2: Symbolic representation of fuzzy inference.

Especificación de FLC

Características deseables:

- Fácil de implementar herramientas
- Describe variables de entrada
- Describe variables de salida
- Describe reglas
- Describe funciones de pertenencia
- Describe funciones de decodificación

VARIABLES

- Dominio de la variable

FUNCIONES DE PERTENENCIA

- Lista de puntos (o polígono de pertenencia)

Especificación de FLC (2)

Lenguaje FIL (de Aptronix):

- Gramática C-like.
- Orientado al software.
- Elección de técnicas a ser usadas (inferencia y defusificación).

Otros lenguajes:

- Redundancia (ancho de bits de las MF y por otro lado se define el dominio).
- Varios archivos para la definición de un FLC (variables, funciones de pertenencia y reglas)

=> posibles inconsistencias

Se define nFIL (del INCA):

- Gramática PROLOG-like.
- Un FLC por archivo de definición.
- Tamaño de datos deducido a partir de los dominios de las variables.

Especificación de FLC (3)

Lenguaje FIL (de Aptronix):

```
$ FILENAME: pdlm2.fil
fiu tvfi (min max);
$mamdani/tvfi (min max prod sum binter bunion) 1.0 4 5 6 7 8
>velocity " "    [  PL (@0,255, @32,255, @80,0),
                    PM (@32,0, @80, 255, @112,0),
                    PS (@80,0, @112,255, @144,0),
                    NS (@112,0, @144,255, @176,0),
                    NM (@144,0, @176,255, @222,0),
                    NL (@176,0, @222,255, @255,255) ];
>angle " "   [      NL (@0,255, @64,0),
                    NM (@0,0, @64, 255, @128,0),
                    ZR (@64,0, @128,255, @192,0),
                    PM (@128,0, @192,255, @255,0),
                    PL (@192,0, @255,255)          ];
. . . . . . . . . . . . . . . . . . . . . . . . . .
```

Especificación de FLC (4)

Lenguaje FIL (de Aptronix):

```
. . . . . .
$ rules start here:
IF velocity is NL and angle is NL THEN voltage is NL;
IF velocity is NL and angle is NM THEN voltage is NL;
IF velocity is NL and angle is ZR THEN voltage is NL;
IF velocity is NL and angle is PM THEN voltage is NM;
IF velocity is NL and angle is PL THEN voltage is NS;
IF velocity is NM and angle is NL THEN voltage is NL;
IF velocity is NM and angle is NM THEN voltage is NL;
IF velocity is NM and angle is ZR THEN voltage is NM;
IF velocity is NM and angle is PM THEN voltage is NS;
IF velocity is NM and angle is PL THEN voltage is PS;
. . . . . .
end.
```

Especificación de FLC (5)

nFIL:

invar("IV",NIV,MinX,MinY,MaxX,MaxY)

- Define la variable de entrada "IV", en el dominio definido por el rectángulo (MinX,MinY) a (MaxX,MaxY).
- También la asocia con el identificador de variables NIV.

Especificación de FLC (6)

nFIL:

outvar("OV",NOV,MinX,MinY,MaxX,MaxY)

- Define la variable de salida "OV", en el dominio definido por el rectángulo (MinX,MinY) a (MaxX,MaxY).
- También la asocia con el identificador de variables NOV.

Especificación de FLC (7)

nFIL:

f_p("FP",NFP,"IV", Puntos_Polígono)

- define función de pertenencia "**FP**" sobre la variable de entrada "**IV**".
- La función es descrita por el polígono que describe la secuencia de puntos **Puntos_Polígono**.

f_d("FD",NFD,"OV", Puntos_Polígono)

- define función de pertenencia de salida (o de decodificación) llamada "**FD**" sobre la variable de salida "**OV**".
- La función es descrita por el polígono que describe la secuencia de puntos **Puntos_Polígono**.

Especificación de FLC (8)

nFIL:

```
si(
 [is("IVa","FP3"), is("IVb","FP5"),...],
                  is("OVg","FD8")
)
```

Esta sentencia implementa la siguiente regla:

- IF (IV_a is FP_3)and (IV_b is FP_5)and ... THEN (OV_g is FD_8)

NOTE QUE:

- Todos los AND en 1 sola regla.
- Los OR se implementa con varias reglas (ya que SÓLO hay un consecuente por regla).

Especificación de FLC (9)

- **Ejemplo de Kosko en nFIL:**

```
invar("x",0,0,0,255,255)
f_p("LE",0,"x",0,255,31,255,97,0)
f_p("LC",1,"x",76,0,101,255,127,0)
f_p("CE",2,"x",64,0,128,255,128,0)

invar("phi",0,0,0,255,255)
f_p("RB",7,"phi",0,0,32,255,64,255,128,0)

outvar("teta",0,0,0,60,255)
f_d("NB",0,"teta",0,255,0,255,14,0)
f_d("PM",1,"teta",5,0,15,255,25,0)
f_d("PS",2,"teta",17,0,24,255,30,0)

si([is("x","LE"),is("phi","RB")],is("teta","PS"))
si([is("x","LC"),is("phi","RB")],is("teta","PM"))
si([is("x","CE"),is("phi","RB")],is("teta","PM"))
```

Partes de un FLC

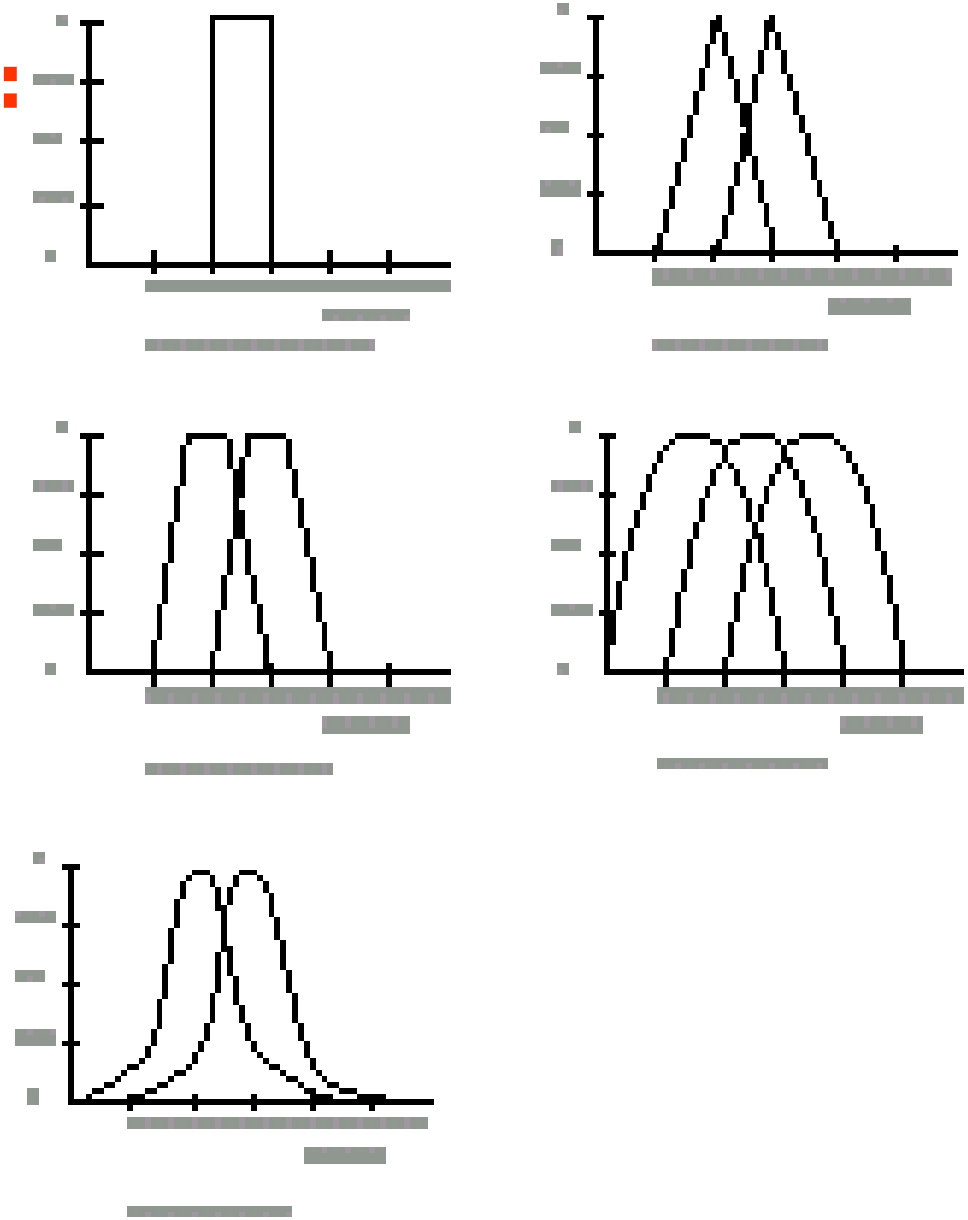
A continuación ...

- Funciones de pertenencia.
- Fusificador.
- Reglas.
- Reglas Activas.
- Defusificador.

Funciones de pertenencia

Tipos clásicos de MF:

- Rectangulares,
- Triangulares,
- Trapezoidales,
- Parabólicas,
- Gausianas,
- etc ...



Funciones de pertenencia (2)

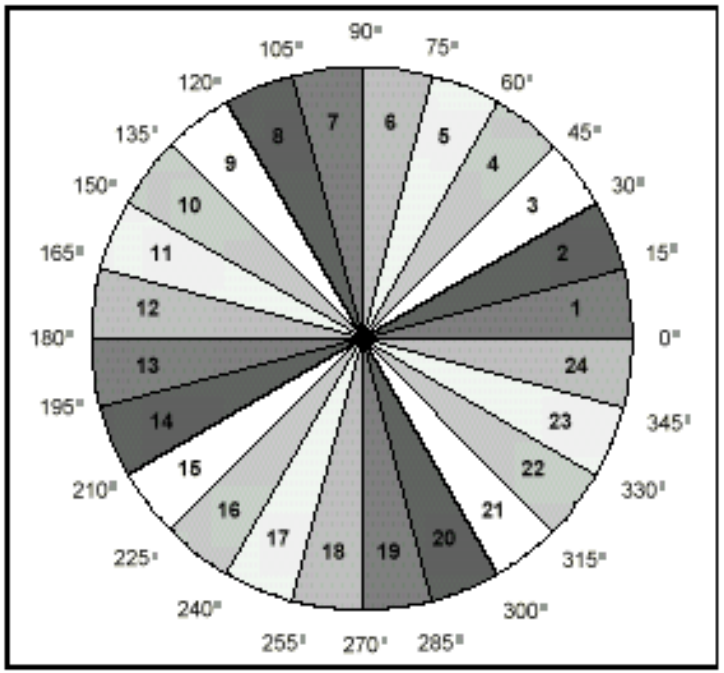


Fig. 11. Numerical notation of fuzzy rule tables.

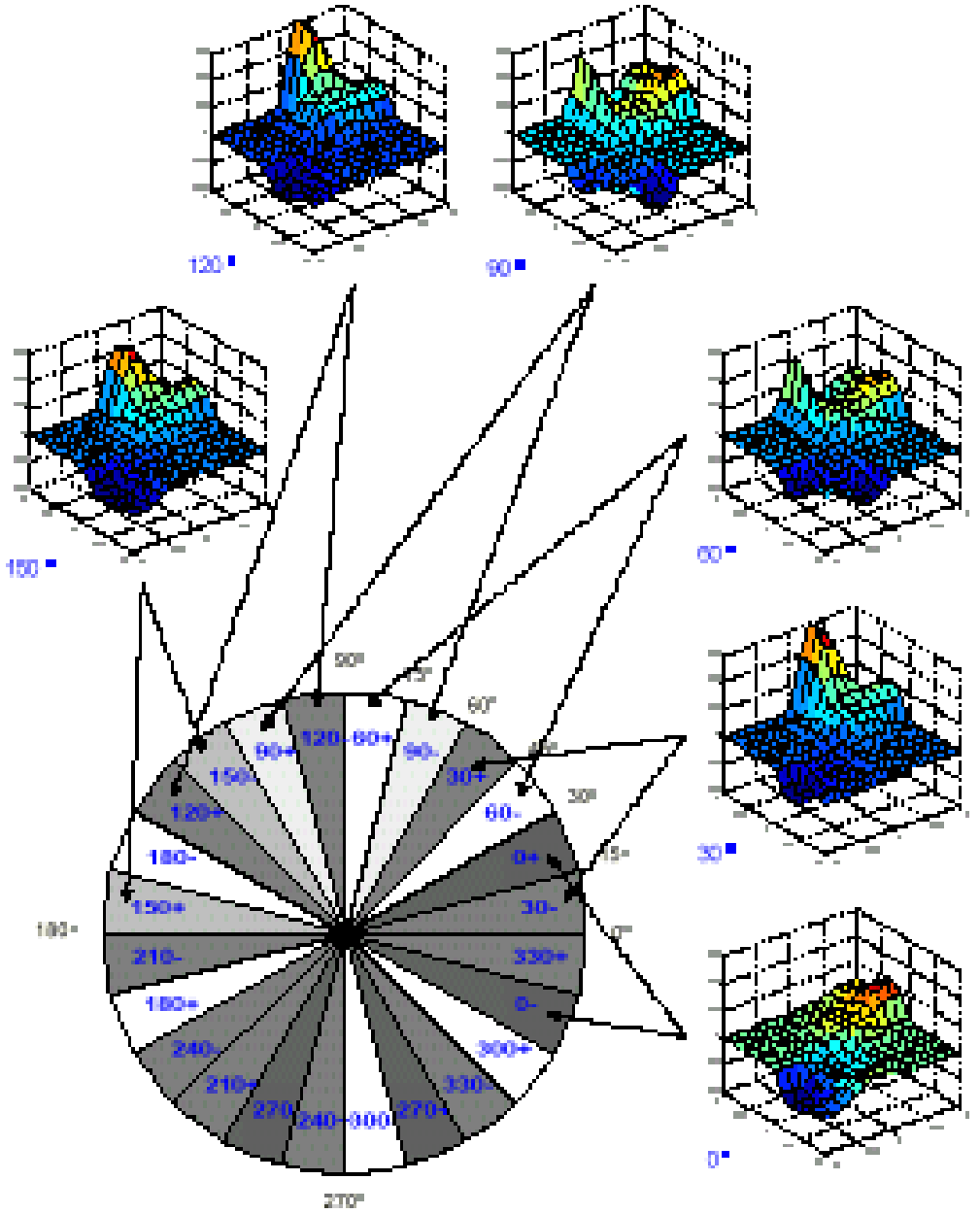


Fig. 4. Distribution of fuzzy rule tables (control surfaces) for the operating range 0° - 150°.

Funciones de pertenencia (3)

SU IMPLEMENTACIÓN:

- **Por Memoria:**
  - Cálculos fuera de línea.
  - Costo independiente de la complejidad de la función.
  - Gran tamaño y tiempo fijo.
- **Calculadas:**
  - Cálculos en línea.
  - Costo dependiente de la complejidad de la función.
  - Menor tamaño y tiempo variable.

Funciones de pertenencia (4)

Implementación por cada Variable:

- **Por Memoria**. Un banco para cada MF.
- **Por Memoria** con (***O***)verlapping bancos. Con O bancos de memoria (**+**gasto en memoria para almacenar el identificador de MF).
- **Calculadas individualmente**. Se calcula cada MF individualmente.
- **Calculadas por activación**. Sólo se calculan las MF que aportarán información (**+**costo por la selección de rangos y, además también, **+**gasto en memoria para almacenar el identificador de MF).

Funciones de pertenencia (5)

Funciones Triangulares (con 3 puntos)

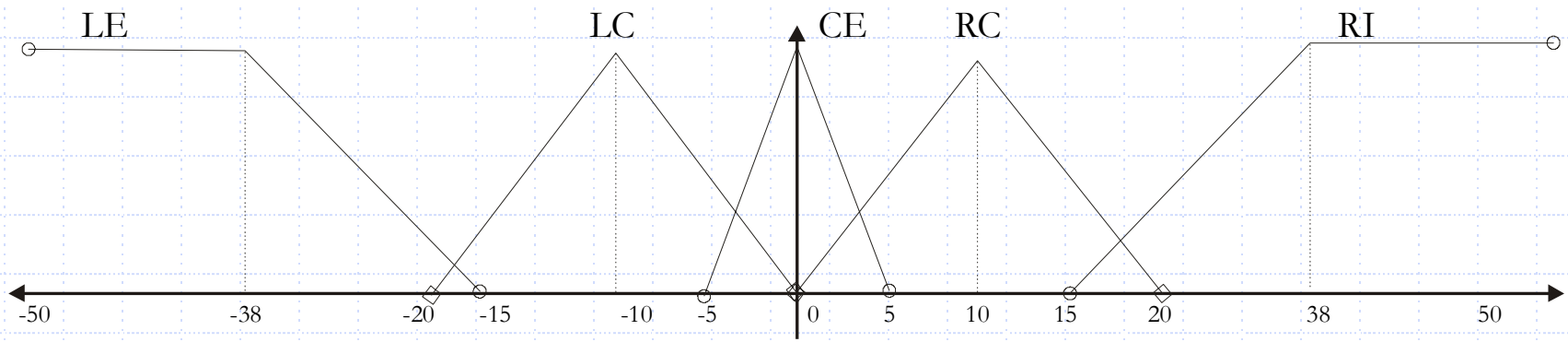
$$F_{LE} = (LE_{x0}, LE_{y0}), (LE_{x1}, LE_{y1}), (LE_{x2}, LE_{y2})$$

$$F_{LC} = (LC_{x0}, LC_{y0}), (LC_{x1}, LC_{y1}), (LC_{x2}, LC_{y2})$$

$$F_{CE} = (CE_{x0}, CE_{y0}), (CE_{x1}, CE_{y1}), (CE_{x2}, CE_{y2})$$

Tratamiento especial:

SI (x < -380)o(si x > 380) **ENTONCES** $x_{mf} := 2^n-1$;

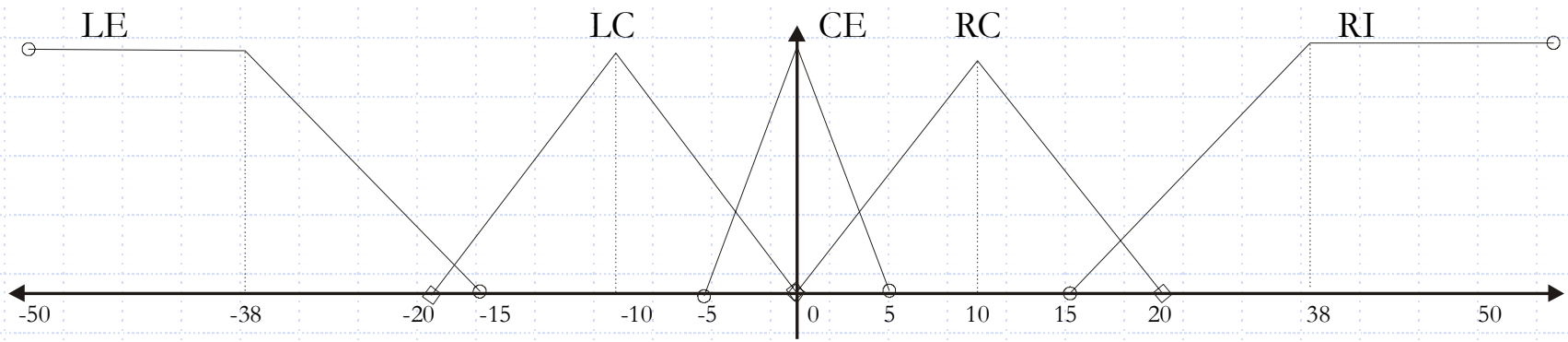


Funciones de pertenencia (6)

Polinomio Lineal Genérico (*n* puntos)

$$F_{LE} = LE_0 \rightarrow LE_1, \quad LE_1 \rightarrow LE_2$$
$$F_{LC} = LC_0 \rightarrow LC_1, \quad LC_1 \rightarrow LC_2$$
$$F_{CE} = CE_0 \rightarrow CE_1, \quad CE_1 \rightarrow CE_2$$

- Cambio de la estructura de datos y cálculo.
- De triangulo se pasa a segmento.



Funciones de pertenencia (7)

Polinomio lineal (SEGMENTOS+ID)

$$LE_0 \rightarrow LE_1 = (LE_{x0}, LE_{y0}) \rightarrow (LE_{x1}, LE_{y1}), LE_{MF}$$

$$LE_1 \rightarrow LE_2 = (LE_{x1}, LE_{y1}) \rightarrow (LE_{x2}, LE_{y2}) , LE_{MF}$$

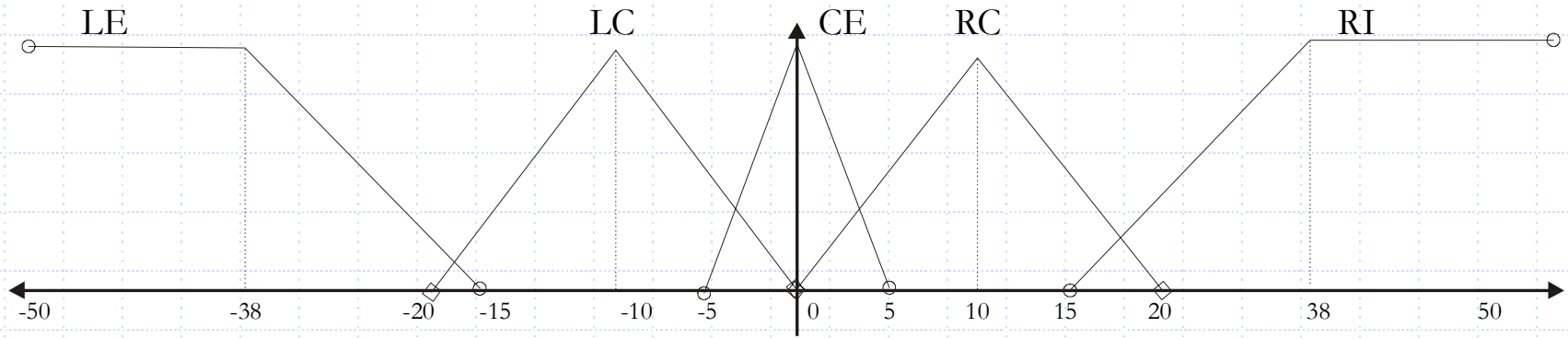
$$LC_0 \rightarrow LC_1 = (LC_{x0}, LC_{y0}) \rightarrow (LC_{x1}, LC_{y1}) , LC_{MF}$$

$$LC_1 \rightarrow LC_2 = (LC_{x1}, LC_{y1}) \rightarrow (LC_{x2}, LC_{y2}) , LC_{MF}$$

$$CE_0 \rightarrow CE_1 = (CE_{x0}, CE_{y0}) \rightarrow (CE_{x1}, CE_{y1}) , CE_{MF}$$

$$CE_1 \rightarrow CE_2 = (CE_{x1}, CE_{y1}) \rightarrow (CE_{x2}, CE_{y2}) , CE_{MF}$$

. . . . . .



Funciones de pertenencia (8)

Clases de MF que pueden implementarse usando polinomios lineales genéricos

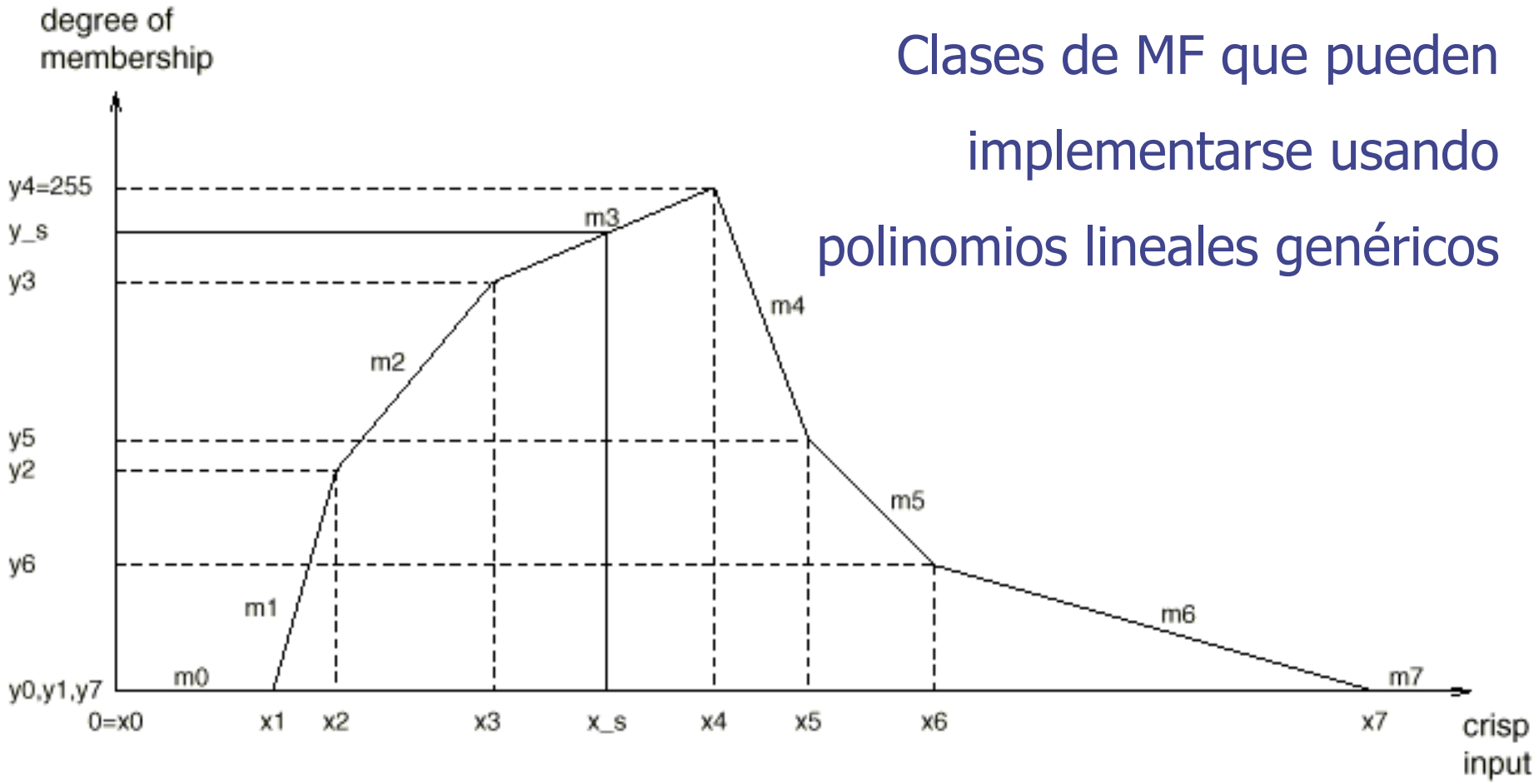
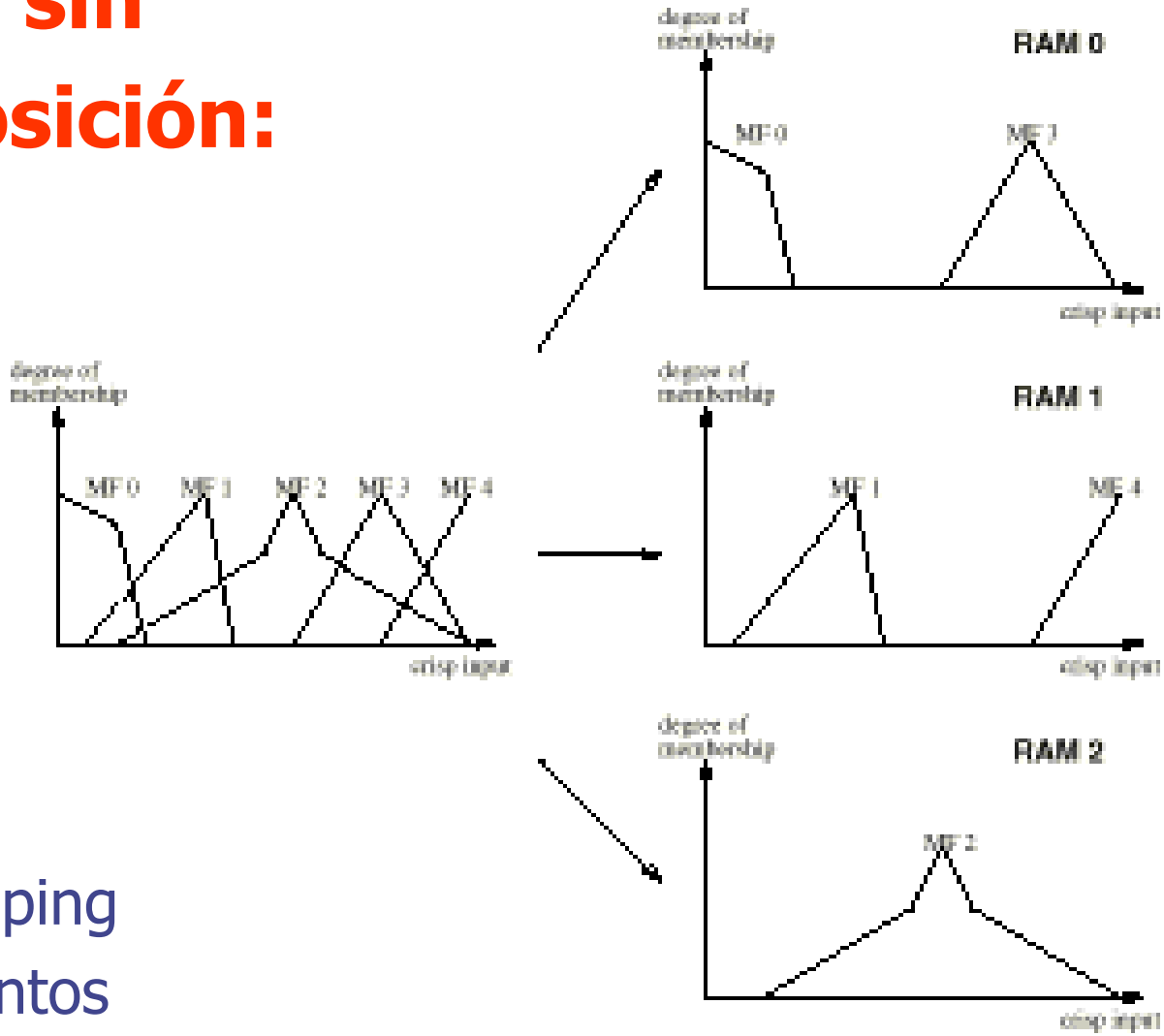


Fig. 17. Membership function approximation

Funciones de pertenencia (9)

En ROM sin superposición:



Ejemplo:
con Overlapping
de 3 segmentos

Fig. 16. Overlap-free Membership Function Storage

Funciones de pertenencia (10)

Impacto de Ensanche de las MF:

- Las **pendientes** de las MF tienen un impacto sobre el comportamiento del FLC
- Al ensancharse la MF se **suaviza** más el comportamiento.
- Puede que haya **mayor superposición de segmentos**.
- Puede **agregar complejidad** de cálculo y disparar un número mayor de reglas.
- **COMPARAR** gráficamente el comportamiento.

Reglas

IF (v_1 is MF_a)&(v_2 is MF_{aa})... THEN s_1 es MF_s

IF (v_1 is MF_b)&(v_2 is MF_{bb})... THEN s_1 es MF_z

IF (v_1 is MF_c)&(v_2 is MF_{cc})... THEN s_1 es MF_s

ENTONCES

se debe calcular la inferencia como:

$$s1_s = \max(\min(v_{1a}, v_{2aa}), \min(v_{1c}, v_{2cc}));$$

$$s1_z = \max(\min(v_{1b}, v_{2bb})) = \min(v_{1b}, v_{2bb});$$

NOTE QUE:

- Todos los AND en 1 sola regla.
- Los OR se implementa con varias reglas (ya que SÓLO hay un consecuente por regla).

Reglas (2)

Funciones para inferencia *max-min*:

- **MINIMO**. Opciones:
  - Operación de 2 operandos. Se debe hacer un arbol de cálculo para evaluar todo.
  - Operación de (*Nro-Maximo-de-Variables-de-Entrada*) operandos.
- **MAXIMO**. Opciones:
  - Operación de 2 operandos. Se debe hacer un arbol de cálculo para evaluar todo.
  - Operación de (*Nro-Maximo-de-Variables-de-Reglas-Con-Igual-Consecuente*) operandos.

Motores de Inferencia

Por la forma de evaluación:

- **NORMAL**. Requiere el cálculo de cada regla individualmente, acumulando su máximo.
- **REGLAS ACTIVAS**. Evalúa sólo las reglas que aportan información a la toma de decisión.

Por la forma de definición de reglas:

- **Con reglas internas**, metidas dentro del software.
- **Con motor de inferencia y reglas externas**. Más lento, pero más flexible (permite prototipar rápidamente).

Reglas (4)

Ideas para RA o reglas externas:

- Basado en **software**, **hardware** o **algoritmos**. Usan un algoritmo para determinar que reglas se deben evaluar.
- Basado en una **estructura de datos**. La base de reglas (o FAM: fuzzy associative memory) tiene los punteros a las MF y son accedidas de forma automática.
- Basado en una **hardware a medida**. La FAM y motor de inferencias son hechos a medida de la aplicación.

Reglas Activas

Características BÁSICAS inferencia

- No todas las reglas aportan información
- Sólo se ejecutan un máximo de 4 reglas
- Variables de entrada y funciones de pertenencia superpuestas

X selecciona 2 MF y ***PHI*** selecciona 2 MF.

ϕ \ X	LE	LC	CE	RC	RI
RB	**PS**	**PM**	**PM**	**PB**	**PB**
RU	**NS**	**PS**	**PM**	**PB**	**PB**
RV	**NM**	**NS**	**PS**	**PM**	**PB**
VE	**NM**	**NM**	**ZE**	**PM**	**PM**
LV	**NB**	**NM**	**NS**	**PS**	**PM**
LU	**NB**	**NB**	**NM**	**NS**	**PS**
LB	**NB**	**NB**	**NM**	**NM**	**NS**

Reglas Activas (2)

Objetivo

- Calcular SÓLO reglas que aporten información.

Procedimiento

- Las MF deben indicar cual es la función de pertenencia con valor > 0 de cada variable.

ϕ	X					
		LE	LC	CE	RC	RI
	RB	PS	PM	PM	PB	PB
	RU	NS	PS	PM	PB	PB
	RV	NM	NS	PS	PM	PB
	VE	NM	NM	ZE	PM	PM
	LV	NB	NM	NS	PS	PM
	LU	NB	NB	NM	NS	PS
	LB	NB	NB	NM	NM	NS

- Se accede a la tabla de reglas con dichos valores para determinar las reglas a ejecutar.

Reglas Activas (3)

Implementación

- **Basado en algoritmo**. Un proceso realiza la selección de reglas ...
- **Basado en estructuras de datos**. La fusificación genera también punteros que apuntan al consecuente en la FAM.

ϕ	X					
		LE	LC	CE	RC	RI
	RB	**PS**	**PM**	**PM**	**PB**	**PB**
	RU	**NS**	**PS**	**PM**	**PB**	**PB**
	RV	**NM**	**NS**	**PS**	**PM**	**PB**
	VE	**NM**	**NM**	**ZE**	**PM**	**PM**
	LV	**NB**	**NM**	**NS**	**PS**	**PM**
	LU	**NB**	**NB**	**NM**	**NS**	**PS**
	LB	**NB**	**NB**	**NM**	**NM**	**NS**

Defusificador

- Se debe calcular bien los tamaños de los **registros que acumulan** el numerador N y el denominador D.

$$\mathbf{wN} = \log_2(\mathbf{maxValMF}) + \log_2(\mathbf{maxValDF}) + \log_2(\mathbf{O})$$

$$\mathbf{wD} = \log_2(\mathbf{maxValMF}) + \log_2(\mathbf{NroDF})$$

- La **precisión** de la división define la precisión de la variable de salida del FLC.

Y Ahora? ...

Será cierto que termina la clase 2?

LISTO!