Consider the membership function Warm shown below.

We would take a reading from a sensor and convert it with an ADC.

Our membership function input is then really an integer reading from the ADC so we could remap the function on a new axis.

Equation of the line is \( y = mx + b \)

\[
\begin{align*}
0 &= m(140) + b \\
1 &= m(130) + b \\
b &= -140m \\
l &= 130m - 140m
\end{align*}
\]

\[
\begin{align*}
m &= -0.1 \\
b &= 14 \\
y &= -0.1x + 14.0
\end{align*}
\]
So to find the degree of membership in Warm we must substitute into

\[ y = -0.1x + 14.0 \]

This would not be a difficult equation to implement in circuitry but it still would involve fractions that require fixed-point or floating point numbers.

What if we use integers for the \( y \)-axis as well?

That is, let’s scale the degree of membership to some other range besides 0 to 1, similar to what we would do for the Analog-to-Digital Conversion itself.

\[
\begin{align*}
&\text{A-to-D} \\
&\text{Analog Voltage} \\
&5.0 \text{V} \\
&0.0 \\
\end{align*}
\]

So, let’s try a maximum degree of membership of 255 (it’s really only the relative strength of degrees of membership that matter in the fuzzy control process.)

Look at membership function Warm again.
The equation of the line for an input of 133 is \( y = -25.5x + 3570 \).

The slope is still a fraction and takes us away from integers.

However, what if we pay closer attention to our numbers and change the maximum degree of membership to a “convenient” value?

Try this:

\[ O = 140m + b \]
\[ I = 130m + b \]
\[ m = -4 \]
\[ b = 560 \]

\[ y = -4x + 560 \]

Not only is the slope an integer, but even more importantly, it no longer even requires a multiply. It is just a shift operation.
We can now outline the structure of a Membership Function Circuit (MFC). Each input membership function will be represented by one MFC. Since we are doing this in hardware we can have the input variable go to many MFC’s working in parallel.

Assuming the previous triangular shaped membership function, the comparators will determine if the input variable is >120°, >130°, and >140°. This allows us to determine if the input variable intersects the membership function at all. If it does, we can also determine whether it intersects the left line (with positive slope) or the right line (with negative slope).

* We will compute the intersection of the input with both lines simultaneously with the math circuitry shown, and then use the multiplexer to output the appropriate degree of membership.
The MUX select lines can easily be created from a simple truth table.

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

To left of membership function

Up slope line

Down slope line

To right of membership function

\[
X_1 = 130 \cdot 140
\]

\[
X_2 = 120 \cdot 130
\]

For the end or "plateau" membership functions, it is even easier.

\[M\]
<table>
<thead>
<tr>
<th></th>
<th>&gt;130</th>
<th>&gt;140</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$X_3 = 140 \quad X_4 = 130$

Can use comparator outputs directly as MUX select lines

To help in the rule evaluation, each of these MFC circuits also output a "triggered" signal in addition to the proper degree of membership.

**Fuzzy Inference Engine**

Since the MFCs output both the degree of membership and a trigger signal we can directly map the rules (summarized as a Fuzzy Associative Matrix) into circuit matrix.
Defuzzification

Centroid Method is OK in software, but is not the most practical for hardware.

Two other techniques are probably more appropriate for hardware synthesis.

(1) Mean-Max Membership and (2) Weighted Average

Consider the following two output membership functions that have been triggered.
Mean-Max Membership: For all output membership functions triggered, average the crisp outputs that correspond to their maximum memberships (peaks).

\[ \text{OUT} = \frac{\sum_{i=1}^{n} C_i}{n} \]

\[ \text{OUT} = \frac{C_A + C_B}{2} = \frac{40 + 50}{2} = 45 \]

This method doesn't account for the degree of membership, though, limited output resolution.

Weighted Average: We've seen this weighted technique already.

\[ \text{OUT} = \frac{\sum_{i=1}^{n} \mu(Z_i) \times C_{Z_i}}{\sum_{i=1}^{n} \mu(Z_i)} \]

\[ \text{OUT} = \frac{0.6(40) + 0.4(50)}{0.6 + 0.4} = \frac{\mu(A) C_A + \mu(B) C_B}{\mu(A) + \mu(B)} \]
Mean-Max Membership: “Simple”, Just need to know which functions were triggered and their centers.

However many layers are needed.
Weighted Average:

Centers of Output Membership Functions

Degrees of Output Membership

Multiplier Block

Adder Block

Dividend

Crisp Output

Adder Block

Divisor

Divider