Capítulo 2

Control Borroso

En este capítulo se hará una introducción al Control Borroso, explicando el funcionamiento de la lógica borrosa y los puntos más importantes a tener en cuenta para el diseño de un sistema borroso. Es de destacar que el control borroso, principal aplicación de los sistemas borrosos, aparte de un tema de estudio académico, resulta muy importante desde el punto de vista industrial, campo en el que como se ha visto en el Capítulo de introducción de este proyecto existen desde hace tiempo mulitud de aplicaciones de estos sistemas en funcionamiento. También se hablará de la norma IEC 1131-7 (*International Electrotechnical Commission*), que define un lenguaje común para todos los autómatas programables y en concreto la Parte 7 que hace referencia a la programación de controladores borrosos.

Puesto que en este proyecto se utilizará el Control Borroso implementado en un autómata programable (*Programmable Logic Controller*, PLC) se explicará la Librería Borrosa (*Fuzzy Library*) del software Unity Pro de Schneider Electric. Dicha librería cumple con la norma IEC 1131-7 y nos permitirá aplicar técnicas de control borroso en un PLC.

Por último se realizará una introducción a la Fuzzy Logic Toolbox™ de MATLAB®, haciendo hincapié en la interfaz gráfica de dicha herramienta que proporcionará un modo sencillo e intuitivo para desarrollar los sistemas borrosos. Esta herramienta de MATLAB será muy útil para realizar las simulaciones de los controladores borrosos que se utilizarán a lo largo de este proyecto.

2.1. Control Borroso

Los actuales sistemas de procesamiento digitales tienen como base el binomio formado por la lógica *booleana* o clásica y la arquitectura von Neumann. Sin embargo, y pese a sus indiscutibles logros, este esquema presenta problemas a la hora de abordar tareas como las denominadas del mundo real, donde la información se presenta masiva, imprecisa y distorsionada.

Para abordar este tipo de tareas se han propuesto modelos alternativos, entre los que destacan los sistemas basados en lógica borrosa (*fuzzy logic*). Estos modelos

de procesamiento y control se engloban con los términos inteligencia computacional (por oposición a la inteligencia artificial clásica) o soft computing (por oposición a la hard computing convencional). Los sistemas borrosos se inspiran en las soluciones que la naturaleza ha encontrado a lo largo de millones de años de evolución para el tratamientode información de tipo masiva y distorsionada procedente del entorno natural.

Los sistemas borrosos (*fuzzy systems*) se introducen para manejar eficazmente conceptos vagos e imprecisos como los empleados en la vida cotidiana, y que nuestro cerebro está acostumbrado a tratar. Por ejemplo, en la realidad el agua no se presenta en tan sólo dos estados, caliente o fría, como diría la lógica *booleana*, sino más bien gélida, fría, templada, caliente o quemando. A partir de estos conceptos y como generalización de las reglas de la lógica *booleana*, base de nuestros sistemas digitales, los sistemas borrosos llevan a cabo un tipo de razonamiento aproximado semejante al desarrollado por el cerebro.

Estos sistemas ya se utilizan ampliamente en la resolución de tareas relacionadas con el procesamiento de señal, reconocimiento de patrones y control. El control es el mayor campo de aplicación de la lógica borrosa.

2.1.1. Lógica borrosa

La lógica borrosa tiene sus raíces en la teoría de conjuntos borrosos desarrollada por Lotfi A. Zadeh en la década de los 60. Esta teoría propone que un elemento siempre pertenece en un cierto grado a un conjunto y nunca pertenece del todo al mismo; esto permite establecer una manera eficiente para trabajar con incertezas, así como para acondicionar el conocimiento en forma de reglas hacia un plano cuantitativo, factible de ser procesado por computadores. De esta manera, los sistemas de control basados en lógica borrosa combinan unas variables de entrada (definidas en términos de conjuntos borrosos), por medio de grupos de reglas que producen uno o varios valores de salida.

Los sistemas borrosos permiten modelar cualquier proceso no lineal, aprender de los datos haciendo uso de determinados algoritmos de aprendizaje y por último permiten utilizar fácilmente el conocimiento de los expertos de un tema, bien directamente, bien como punto de partida para una optimización automática, al formalizar el conocimiento a veces ambiguo de un experto de una forma realizable. Además, gracias a la simplicidad de los cálculos necesarios (sumas y comparaciones, fundamentalmente), normalmente puden realizarse en sistemas baratos y rápidos.

Toda lógica consiste en formalizar el pensamiento humano, desde este punto de vista:

- Lógica Clásica: establece que cualquier enunciado o proposición puede tener un valor lógico verdadero o falso, en definitiva 1 y 0. De esta forma es posible desarrollar toda una lógica basada en leyes de este tipo.

- Lógica Borrosa: en vez de trabjar con el clásico concepto de inclusión o exclusión, introduce una función que expresa el grado de "pertenencia" de una variable hacia un atributo o "variable lingüística" tomando valores en el rango de 0 a 1.

La lógica borrosa tiene dos significados diferentes. En un sentido estricto, la lógica borrosa es un sistema lógico, que es una extensión de la lógica multi-evaluada. Sin embargo, en un sentido más amplio, la lógica borrosa (FL) es casi sinónimo de la teoría de conjuntos borrosos, una teoría que se refiere a las clases de objetos con fronteras poco definidas en las que la pertenencia es una cuestión de grado.

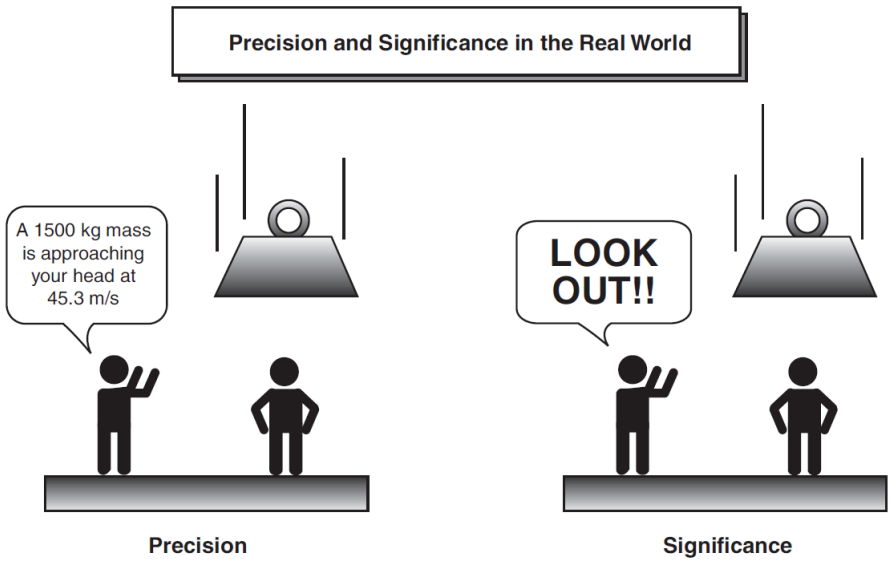

Figura 2.1.1: Precisión y significado en el mundo real.

2.1.2. ¿Por qué utilizar lógica borrosa?

A continuación se muestra una pequeña lista con algunas observaciones generales acerca de la lógica borrosa:

- La lógica borrosa es conceptualmente fácil de comprender. Los conceptos matemáticos detrás del razonamiento borroso son muy simples. La lógica borrosa es más un método intuitivo, sin la complejidad de una larga investigación.
- La lógica borrosa es flexible. Dado cualquier sistema, es fácil añadir más funcionalidades sin volver a empezar desde cero.
- La lógica borrosa es tolerante con los datos imprecisos. Todo es impreciso si se mira con atención. El razonamiento borroso se basa en esta comprensión del proceso en lugar de fijarse sólo en el final.
- La lógica borrosa puede modelar funciones no lineales de complejidad arbitraria. Se puede crear un sistema borroso para combinar con cualquier conjunto de

datos de entrada-salida. Este proceso se hace particularmente fácil por medio de las técnicas adaptativas como los Sistemas de Inferencia Neuro-Borrosos Adaptativos (*Adaptive Neuro-Fuzzy Inference Systems*, ANFIS).

- La lógica borrosa puede construirse con la experiencia de los expertos. En contraste directo con las redes neuronales, las cuales toman datos de entrenamiento y generan modelos opacos e impenetrables, la lógica borrosa permite confiar en la experiencia de las personas que ya entienden el funcionamiento del sistema.
- La lógica borrosa puede mezclarse con las técnicas de control convencionales. Los sistemas borrosos no tienen por qué reemplazar necesariamente los métodos de control convencionales. En muchos casos los sistemas borrosos aumentan y simplifican su implementación.
- La lógica borrosa se basa en el lenguaje natural. La base de la lógica borrosa es la base para la comunicación humana. Esta observación sustenta muchos de las otras declaraciones sobre la lógica borrosa. Debido a que la lógica borrosa se basa en las estructuras de descripción cualitativa utilizada en el lenguaje cotidiano, por ello la lógica borrosa es fácil de usar.

La última declaración es quizás la más importante y la que merece mas discusión. El lenguaje natural, el cual es utilizado por la gente ordinaria todos los días, se ha formado durante miles de años de historia humana hasta llegar a ser conveniente y eficiente. Las frases que se escriben en el lenguaje ordinario representan un triunfo en la comunicación eficaz.

Pero la lógica borrosa no es una panacea. ¿Cuándo no se debe utilizar la lógica borrosa? La lógica borrosa es un método práctico para asignar un espacio de entrada a un espacio de salida. Si se encuentra que no es conveniente, habría que probar otra cosa. Si ya existe una solución más simple, lo conveniente sería utilizarla. La lógica borrosa es la codificación del sentido común (usando el sentido común cuando se implemente se tomará probablemente la decisión correcta). Muchos controladores, por ejemplo, hacen un buen trabajo sin utilizar lógica borrosa. Sin embargo, si se toma el tiempo necesario para llegar a familiarizarse con la lógica borrosa, se verá que puede ser una herramienta muy poderosa para el tratamiento rápido y eficiente de la imprecisión y la no linealidad.

2.1.3. Conjuntos borrosos

La lógica borosa empieza con el concepto de un conjunto borroso. En los conjuntos clásicos algo está incluido completamente en él o no lo esta en absoluto. Esta situación puede describirse asignando un 1 a todos los elementos incluidos en el conjunto y un 0 a los no incluidos. A la función que asigna estos valores se le denomina funcion de inclusión o pertenencia (*membership fuction*). Los conjuntos borrosos permiten describir el grado de pertenencia o inclusión de un objeto (o valor de una variable) al concepto dado por la etiqueta que le da el nombre, asignando un número real entre 0 y 1 (así, por ejemplo, una persona podrá ser medio baja o muy alta).

Figura 2.1.2: (a) Conjunto clásico. (b) Conjunto borroso. El conjunto de los días de la semana, sin duda, incluye los lunes, jueves y sábados. Es así como, sin duda, excluye la mantequilla, la libertad y las aletas dorsales, y así sucesivamente. Todo cae en cualquiera de los grupos, no hay una cosa que sea un día de la semana y no sea un dia de la semana. Ahora, si consideramos el conjunto de días que comprende un fin de semana, la mayoría estaría de acuerdo en que los sábados y domingos pertenecen, pero ¿y el viernes? Se siente como una parte del fin de semana, pero de alguna manera parece que debería estar técnicamente excluido. Con los conjuntos clásicos no se tolera este tipo de clasificación.

Cualquier declaración puede ser borrosa. La principal ventaja que ofrece el razonamiento borroso es la capacidad de responder a una pregunta de sí o no con una respuesta no del todo de sí o no. Los seres humanos hacen este tipo de cosas todo el tiempo (muy pocas veces se obtiene una respuesta directa a una pregunta aparentemente simple) pero es un truco bastante nuevo para los ordenadores.

Uno de los ejemplos mas utilizados al explicar los conjuntos borrosos es el conjunto de personas altas. En este caso, el universo de discuro es todas las alturas posibles, desde 0.8 m a 2.7 m, y la palabra alto correspondería a una curva que define el grado en que cualquier persona es alta. Si se define el conjunto de personas altas como un conjunto clásico, se podría decir que todas las personas más altas de 1.8 m son oficialmente considerados altos. Sin embargo, esta distinción es claramente absurda. Tal vez tenga sentido considerar el conjunto de número reales mayores de 1.8 porque los números pertenencen a un plano abstracto, pero cuando queremos hablar acerca de gente real, no es razonable llamar a una persona baja de estatura y a otro alto cuando difieran en altura poco más del ancho de un cabello.

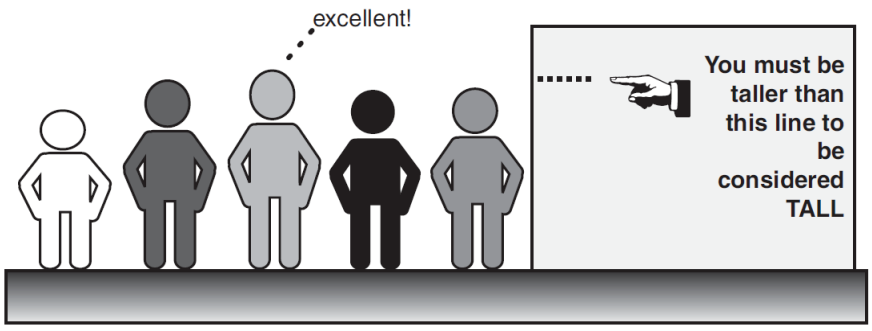

Figura 2.1.3: Ejemplo de un conjunto de personas donde se hace una distinción por altura.

La distinción que se ha mostrado anteriormente es impracticable, ¿cuál es la forma correcta de definir el conjunto de las personas altas entonces? La siguiente

Figura 2.1.4 muestra una curva que define la transición de una persona alta y a otra que no lo es.

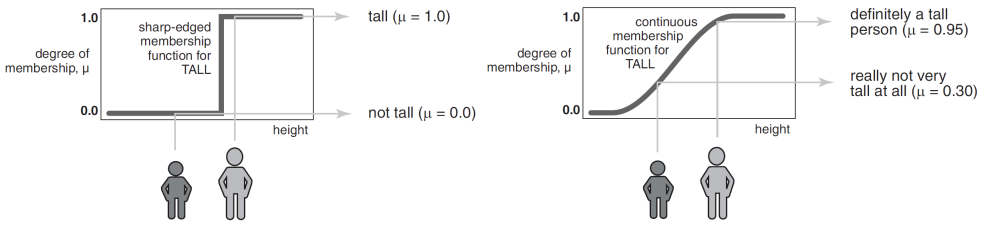

Figura 2.1.4: Funciones de pertenencia para definir si una sona es es alta o no.

Ambas personas son altas en cierta medida, pero una es significativamente menos alta que la otra. Las interpretaciones subjetivas y las unidades apropiadas ya se tienen en cuenta en los conjuntos borrosos. Si se dice "es alta", la función de pertenencia de altura debería tomar en cuenta si se refiere a un niño de seis años de edad o a una mujer adulta. Del mismo modo, las unidades se incluyen en la curva.

Una vez visto el ejemplo anterior podemos definir matemáticamente los conjuntos borrosos. Sea U un conjunto de objetos, por ejemplo, $U = \Re^n$, que se denominará universo de discurso. En términos matemáticos, un conjunto borroso F en U queda caracterizado por una función de inclusión o pertenecia μ_F que toma valores en el rango [0, 1], donde $\mu_F(u)$ representa el grado en el que $u \epsilon U$ pertenece al cojunto borroso F. Ello representa la generalización del concepto clásico de conjunto, en el que la función de pertenencia toma solamente los alores 0 o 1; por el contrario, para uno borroso, la función puede tomar también valores intermedios.

El conjunto soportado es el conjunto (clásico) de todos los valores de U para los que $\mu_F(u) > 0$. Se dice que un conjunto borroso es de tipo *singleton* si su conjunto soportado es de un sólo valor. Por otro lado, se dice que un conjunto borroso está normalizado si el máximo de su función de pertenencia es 1; obviamente, un conjunto borroso puede normalizarse multiplicando su función de pertenencia por un coeficiente fijo para que sea de tipo normalizado.

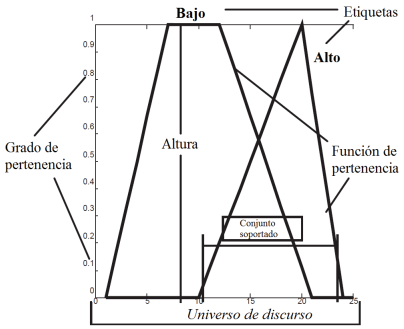

Figura 2.1.5: Términos principales de las funciones de pertenencia.

2.1.3.1. Funciones de pertenencia

Una función de pertenencia es una curva que define la forma en que cada punto en el espacio de entrada se le asigna un valor de pertenencia (o grado de pertenencia) entre 0 y 1 (esta es realmente la única condición que debe satisfacer la función de pertenencia), es decir, el valor de $\mu_F(u)$ indica el grado en que el valor de u de la variable U está incluida en el concepto representado por la etiqueta F. Para la definición de estas funciones de pertenencia se utilizan convencionalmente ciertas familias de formas estándar, por coincidir en el significado lingüístico de las etiquetas más utilizadas. Las mas frecuentes son la función de tipo trapezoidal, singleton, triangular, S, exponencial y de tipo gaussiana (Figura 2.1.6).

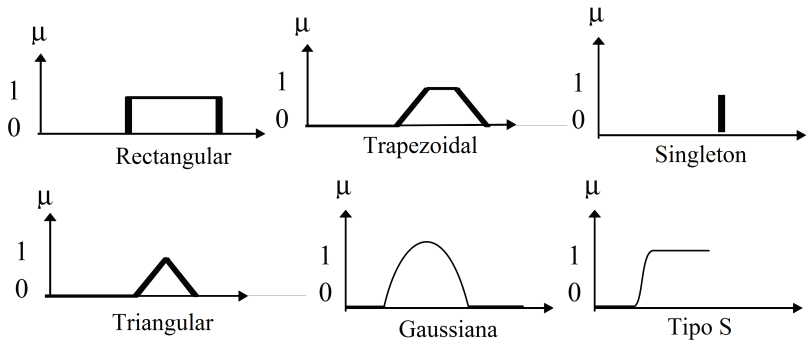

Figura 2.1.6: Formas comúnmente utilizadas de funciones de pertenencia.

Algunas propiedades de las funciones de pertenencia antes nombradas:

- La función de tipo trapezoidal se utiliza habitualmente en sistemas borrosos sencillos, pues permite definir un conjunto borroso con pocos datos, y calcular su valor de pertenencia con pocos cálculos. Se emplea especialmente en sistemas basados en microprocesador. Esta función resulta adecuada para modelar propiedades que comprenden un rango de valores (*adulto, normal, adecuada...*).
- La función de tipo singleton se utiliza habitualemnte en sistemas borrosos simples para definir los conjuntos borrosos de las particiones de las variables de salida, pues permite simplificar los cálculos y requiere menos memoria para almacenar la base de reglas.
- La función de tipo triangular es adecuada para modelar propiedades con un valor de inclusión distinto de cero para un rango de valores estrecho.
- La función de tipo S resulta adecuada para modelar propiedades como *grande, mucho, positivo...*
- La función de tipo gaussiana resulta adecuada para propiedades como *medio, normal, cero...*

2.1.3.2. Variable lingüística

Se denomina variable lingüística a aquella que puede tomar por valor términos del lenguaje natural, como *mucho, poco, positivo, negativo,* etc., que son las palabras

que desempeñan el papel de etiquetas en un conjunto borroso. Aunque el objetivo principal de este concepto es expresar de manera formal el hecho de que pueden asignarse como valor de una variable palabras tomadas del lenguaje natural, no obstante a una variable lingüística podrán asignarse también valores numéricos. Así, en una expresión como la t*emperatura es fría*, la variable temperatura debe ser entendida como una variable lingüística, pues se le asigna como valor el conjunto borroso *fría*, pero además esta variable puede también tomar valores numéricos como la *temperatura es* $4^{o}C$.

Otro concepto importante es el de partición. Se conoce por partición a un conjunto de los conjuntos (subconjunto) borrosos que se han definido para una variable A. Los nombres de los conjuntos borrosos que forman una partición se suelen expresar en forma abreviada por sus iniciales; así, una partición típica como {Negativo Grande, Negativo Pequeño, Cero, Positivo Pequeño, Positivo Grande} se representa como {NG, NP, CE, PP, PG}.

2.1.3.3. Operaciones borrosas

A los subconjuntos borrosos se les puede aplicar determinados operadores, o bien pueden realizarse operaciones entre ellos. Al aplicar un operador sobre un sólo conjunto borroso se obtiene otro conjunto borroso; de la misma manera al combinar dos o mas subconjuntos mediante alguna operación, se obtendrá otro conjunto.

Pueden definirse tres operaciones básicas: complemeto, unión e intersección. Estas operaciones básicas pueden expresarse de la siguiente manera en términos de las funciones de pertenencia de los conjuntos borrosos A y B:

Complemento	$\mu_{\overline{A}}(x) = 1 - \mu_A(x)$
Unión	$\mu_{A \bigcup B}(x) = max\left[\mu_A(x), \mu_B(x)\right]$
Intersección	$\mu_{A \bigcap B}(x) = min\left[\mu_A(x), \mu_B(x)\right]$

Cuadro 2.1: Operaciones entre conjuntos borrosos.

Es importante resaltar que el funcionamiento de estas operaciones básicas coincide con el de las correspondientes a las de la teoría clásica de conjuntos; de hecho, la teoría de conjuntos borrosos se reduce a la teoria clásica si reducimos la incertidumbre a 0, y admitimos sólo valores 0 y 1 para las funciones de pertenencia a un conjunto. Es decir, la teoría clásica de conjuntos puede considerarse un caso particular de la borrosa. En términos más generales, se está definiendo lo que se conoce como intersección borrosa (AND), unión borrosa (OR) y complemento borroso (NOT). Los operadores clásicos para estas funciones son: AND=*min*, OR=*max*, y NOT=complemento aditivo.

Además, pueden definirse también otras operaciones (igualdad, norma, concentración, dilatación...).

2.1.3.4. Inferencia borrosa

Como ocurre también en la lógica clásica, la lógica borrosa se ocupa del razonamiento formal con proposiciones, pero a diferencia de ésta, los valores de las proposiciones pueden tomar valores intermedios entre verdadero y falso. El objeto de la lógica borrosa es proporcionar un soporte formal al razonamiento basado en el lenguaje natural, que se caracteriza por tratarse de un razonamiento de tipo aproximado, que hace uso de unas proposiciones que a su vez expresan información de carácter impreciso.

Las reglas borrosas son básicamente de tipo IF-THEN (Si-Entonces) y expresan una relación o proposición borrosa. En lógica borrosa el razonamiento no es preciso, sino aproximado, lo cual quiere decir que se puede inferir de una regla una conclusión aunque el antecedente (premisa) no se cumpla plenamente. Estas declaraciones de reglas IF-THEN se utilizan para formular las instrucciones condicionales que comprenden la lógica borrosa.

Una sóla regla borrosa Si-Entonces tiene la forma: *Si x es A entonces y es B,*
donde A y B son valores lingüísticos definidos por conjuntos borrosos en los rangos (universos de discurso) X e Y, respectivamente. La parte "si" de la regla "x es A" se llama antecedente o premisa, mientras que la parte "entonces" de la regla "y es B" se llama el consecuente o conclusión. En general, una regla por sí sola no es eficaz.

Las reglas borrosas combinan uno o más conjuntos borrosos de entrada (antecedentes o premisas), y les asocian un conjunto borroso de salida (consecuente o consecuencia). Los conjuntos borrosos de la premisa se asocian mediante conjuntivas lógicas como *y, o* , etc. Una regla típica, de tipo IF-THEN, para un sistema de control sería "*Si* error *es* positivo_pequeño *y* derivada_de_error *es* negativo_pequeño *Entonces* acción *es* positiva_pequeña", que se suele expresar abreviadamete mediante expresiones del tipo "*Si* E *es* PP *y* dE *es* NP *Entonces* U *es* PP".

Las reglas borrosas permiten expresar el conocimiento que se dispone sobre la relación entre antecedentes y consecuentes. Para expresar este conocimiento de forma completa normalmente se precisa de varias reglas, que se agrupan formando lo que se conoce como una **base de reglas**, es decir, el conjunto de reglas que expresan las relaciones conocidas entre antecedentes y consecuentes. La base de reglas se puede representar bien como una tabla de las reglas que forman, o bien como una memoria asociativa borrosa o FAM (*Fuzzy Associative Memory*). Las FAM son matrices que representan la consecuencia de cada regla definida para cada combinación de dos entradas. Las FAM permiten realizar una representación gráfica clara de las relaciones entre dos variables lingüísticas de entrada y la variable ligüística de salida, pero requiere que se indique explícitamente todas las reglas que se pueden formar con estas dos variables de entrada.

Existen dos formatos de reglas:

- Borroso puro o de tipo Mamdani, que permite realizar un controlador borroso que estabiliza un sistema en torno a su punto de trabajo.

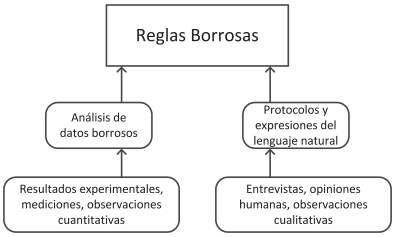

Figura 2.1.7: Las principales fuentes y formas de formulación de reglas.

- Tipo Sugeno, donde la función de salida es una combinacion lineal de las variables de entrada, o en un caso más general, una función genérica de las variables de entrada.

2.1.4. Sistemas de Control Borroso

El control borroso hace referencia al control y regulación de procesos técnicos, incluido el procesamiento de los valores medidos, se basa en el uso de reglas borrosas y de su tratamiento con ayuda de la lógica borrosa.

En general, la entrada a una regla IF-THEN es el valor actual de la variable de entrada y la salida es un conjunto borroso. Este conjunto después se desborrosifica (*defuzzified*) , asignando un valor a la salida. El concepto de desborrosificación (*defuzzification*) se describirá mas adelante.

Interpretar una regla IF-THEN implica distintas partes: en primer lugar evaluar el antecedente (que implica borrosificar la entrada (*fuzzifying*) y la aplicación de todos los operadores borrosos necesarios) y en segundo lugar la aplicación de este resultado con el consecuente.

Los sistemas expertos de control borroso basados en reglas, conocidos como controladores borrosos o FLC (*Fuzzy Logic Controllers*), o también, sistemas de inferencia borrosa o FIS (*Fuzzy Inference Systems*), son sin duda la aplicación más extendida de la lógica borrosa.

Un sistema de inferencia borrosa es el proceso de formulación de la asignación de un determinado insumo para una salida usando la lógica borrosa. El proceso de inferencia borrosa involucra todo lo visto anteriormente: funciones de pertenencia, operaciones borrosas, y reglas IF-THEN.

El proceso de infernecia borrosa consta de cinco partes: borrosificación de las variables de entrada, aplicación del operador borroso en el antecedente, implicación del antecedente al consecuente, la agregación de los consecuentes a través de las normas y la desborrosificación.

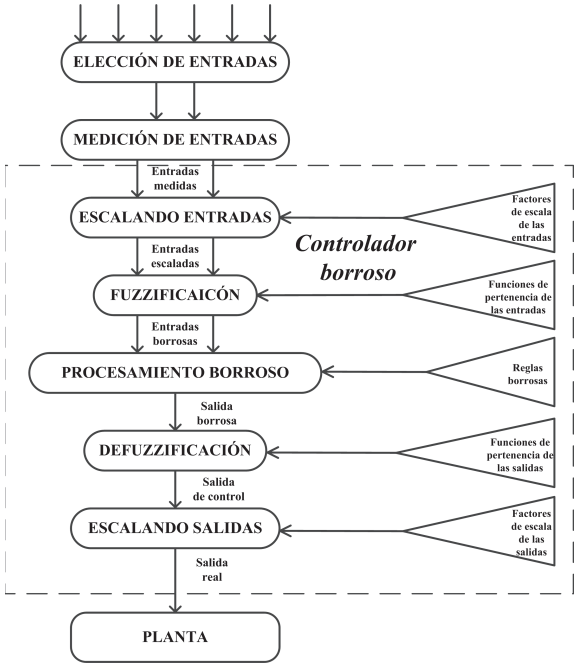

Figura 2.1.8: Funcionamiento del controlador Borroso.

2.1.4.1. Borrosificador (*Fuzzifier*)

El borrosificador establece una relación entre puntos de entrada no borrosos al sistema, y sus correspondientes conjuntos borrosos, es decir, determina una adecuada relación entre las variables de entrada y los términos lingüísticos. Con este fin, el grado real de pertenencia de las variables de entrada se determina para cada término lingüístico de la variable lingüística correspondiente. Se pueden utilizar diversas estrategias de borrosificación:

- Borrosificador singleton. Es el método de borrosificación más utilizado, principalmente en sistemas de control, y consiste en considerar los propios valores discretos con conjuntos borrosos.
- Borrosificador no singleton. En este método de borrosificacion se utiliza una función exponencial.

2.1.4.2. Desborrosificador (*Defuzzifier*)

El desborrosificador es la función que transforma un conjunto borroso en V (universo de discurso), normalmente salida de un dispositivo de inferencia borrosa, en un valor no borroso $y \epsilon V$. Para esta tarea se utilizan diversos métodos:

- Desborrosificador por máximo.
- Desborrosificador por media de centros.
- Desborrosificador por centro de área.

Estos métodos de desborrosificación son los empleados para obtener el valor de salida no borrosa de un dispositivo de inferencia borrosa que utiliza reglas de tipo Mamdani. Si las reglas utilizadas son de tipo Sugeno, el valor de salida no borrosa se obtiene como media ponderada de las salidas de cada regla de la base de reglas.

2.1.4.3. Diferentes controladores borrosos

De las docenas de maneras de hacer el trabajo de un sistema desconocido, resulta que la lógica borrosa es a menudo la mejor manera. Como Lotfi Zadeh dijo una vez: *"En casi todos los casos se puede contruir el mismo producto sin lógica borrosa, pero la lógica borrosa es mas rápido y más barato"*.

La arquitectura del controlador a utilizar depende de la aplicación concreta a llevar a cabo. No resulta fácil realizar una clasificación genérica de todas las arquitecturas posibles de controladores basados en lógica borrosa, pero podemos considerar los siguientes tres grandes grupos de controladores:

1. Controladores borrosos directos sin optimización. En esta estructura, un primer bloque realiza el preprocesado de las variables de entrada para proporcionar las entradas al controlador borroso. En el caso más simple este preprocesado puede consistir en un simple escalado de las magnitudes que se miden. Los controladores directos permiten realizar control de sistemas utilizando una descripción lingüística de las reglas de control. Estas reglas han de obtenerse del conocimiento que disponen los expertos sobre el control del sistema, o bien por procedimientos heurísticos, siendo esta solución en muchos casos suficiente para obtener un buen control del sistema.
2. Controladores borrosos directos con optimización. Parten de la estructura de los controladores borrosos directos sin optimización, añadiendo elementos que permiten ajustar sus parámetros internos con el fin de mejorar su eficiencia. Un primer elemento realiza la evaluación del funcionamiento del controlador para permitir al bloque siguiente decidir sobre las modificaciones a realizar.
3. Controladores borrosos híbridos. Se denominan así aquellos sistemas de control formados por dos controladores interconectados, de los cuales uno es convencional (como los PID) y el otro es borroso. El primero se encarga básicametne del control, garantizando un comportamiento estable, mientras que el controlador borroso actua en paralelo, introduciendo el componente heurístico en el proceso. Este segundo controlador borroso también puede emplearse para el ajuste de los parámetros del controlador convencional como en [11][11].

Una posible solución aceptable de control borroso debería satisfacer las especificaciones de diseño. No debe ser óptima en lo que respecta a algunos criterios, ya que es difícil probar que un sistema de control borroso es óptimo e incluso estable. Sin embargo, un controlador de lógica borrosa es capaz de proporcionar una solución estable y "buena".

De forma general los controladores borrosos no se deberían utilizar cuando:

- Con la teoría de control convencional se obtiene un resultado satisfactorio.
- Cuando ya existe un modelo matemático de fácilmente soluble y adecuado.
- El problema de control no tiene solución.

2.2. Norma IEC 1131-7

Esta parte del IEC 1131 define un lenguaje para la programación de aplicaciones de Control Borroso que usan los autómatas programables. La *International Electrotechnical Commission* (IEC) designa al Comité de Investigación 65A para la definición de una norma específica referente a los API (*Application Programming Interface*). El objetivo de esta norma es responder a la complejidad creciente de los sistemas de control y a la diversidad de autómatas incompatibles entre sí. La norma estandariza varios aspectos de la automatización, se compone de las siguientes partes:

- IEC 1131-1: Informaciones generales.
- IEC 1131-2: Especificaciones y ensayos de equipos.
- IEC 1131-3: Lenguajes de programación.
- IEC 1131-4: Recomendaciones al usuario.
- IEC 1131-5: Especificaciones de servicios de mensajería.
- IEC 1131-6: Comunicación por bus de campo.
- IEC 1131-7: Programación para Control Borroso.
- IEC 1131-8: Recomendaciones para la aplicación e implementación de lenguajes de programación.

Algunas de las principales ventajas de utilizar la norma IEC 1131-3 que define los lenguajes de programación para autómatas son las que se presentan a continuación:

1. Disminución de los costes de formación.
2. Homogeneidad de la documentación de las aplicaciones: estructura de programas idéntica, objetos de lenguaje predefinidos, etc.
3. Variedad de lenguajes estándar: cada función de una aplicación puede programarse en el lenguaje que mejor se adapte para asegurar la coherencia final.
4. Un paso hacia la portabilidad de los programas.

La norma define los lenguajes de programación, y para todos los lenguajes: la sintaxis y representación grafica de los objetos, la estructura de programas y la declaración de variables.

Los lenguajes normalizados:

- LADDER DIAGRAM (LD) o lenguaje (diagrama de contactos).
- FUNCTION BLOCK DIAGRAM (FBD) o esquema de bloques funcionales.
- INSTRUCTION LIST (IL) o lista de instrucciones.
- STRUCTURED TEXT (ST) o lenguaje textual estrcuturado.
- SEQUENTIAL FUNCTION CHART (SFC) o diagrama funcional de secuencias (basado en GRAFCET).

Los objetos de lenguaje IEC 1131 son:

- Declaración de variables
  - Los objetos no predefinidos deberán tener el nombre y el tipo declarado por el programador.
- Objetos predefinidos definidos en 3 zonas (como mínimo):
  - Zona memoria (%M).
  - Zona de entradas (%I).
  - Zona de salidas (%Q).

  Los objetos en cada zona pueden ser (definición mínima):
  - bits (X);
  - bytes (B) - 8 bits;
  - words (W) - 16 bits;
  - double words (D) - 32 bits;
- Ejemplos de objetos estándar:
  - word de la zona de entradas: %IWdir[1]
  - word de la zona memoria: %MWdir
  - bit de la zona memoria: %MXdir o %Mdir (la X se puede omitir)
  - doble word de la zona memoria: %MDdir
  - tabla de 8 words: %MWdir:8
  - bit extrado de word: %MWdir:X4
  - bit de la zona de salidas: %QXdir o %Qdir (la X se puede omitir)

Puesto que se va a trabajar con el software Unity de Schneider Electric que cumple con esta norma es conveniente realizar una pequeña introducción con las principales características de los distintos lenguajes de programación.

[1] dir hace referencia a una dirección de memoria o a una dirección física del autómata (rack, módulo, via, etc.).

Ladder Diagram

- Elementos gráficos organizados en redes conectadas por barras de alimentación.
- Forma gráfica de los elementos impuesta.
- Evaluación de la red por elementos interconectados.
- Elementos utilizados: contactos, bobinas, funciones, bloques funcionales, etc.
- Elementos de control de programa (salto, return,...).

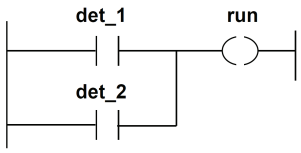

Figura 2.2.1: Ejemplo de programación en LD.

Function Block Diagram (FBD)

- Representación de funciones por bloques enlazados uno a otro.
- Ninguna conexión entre salidas de bloques de función.
- Evaluación de una red: de la salida de un bloque funcional a la entrada de otro bloque funcional.

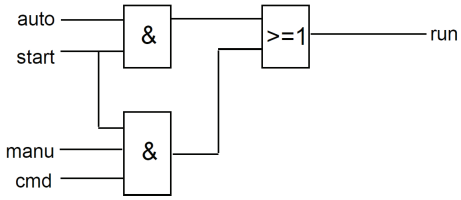

Figura 2.2.2: Ejemplo de programación en FBD.

Instruction List (IL)

Se encuentra formado por una serie de instrucciones: cada una debe empezar en una línea nueva. Una instrucción esta compuesta por un operador y uno o más operandos separados por comas. Otras características:

- Las etiquetas son opcionales y deben terminar en ":".
- Los comentarios son opcionales y deben ser el último elemento de una línea. El comienzo y el final de los comentarios está indicado mediante los símbolos (* *).
- Los bloques de función se emplean con la ayuda de un operador específico (CAL) o utilizando entradas del bloque funcional como operadores.

```
Etiqueta      Operador      Operando      Comentario

MARCHA :      LD            %IX1          (* pulsador *)
              ANDN          %MX5
              ST            %QX2          (* marcha *)
L10     :     LD            %IW12
              ADD           1
              ST            %MW41
              JMP           SET_OUT
```

Figura 2.2.3: Ejemplo de programación IL.

Structured text (ST)

- Sintaxis similar a la del PASCAL, permitiendo la descripción de estructuras algorítmicas complejas.
- Sucesión de enunciados para la asignación de variables, el control de funciones y bloques de función, usando operadores, repeticiones, ejecuciones condicionales, etc.
- Los enunciados deben terminar con ";".

```
J:=1 ;
WHILE J<=100 AND X1< >X2 DO ;
J:=J+2 ;
END_WHILE ;
```

Figura 2.2.4: Ejemplo de programación en ST.

Sequential Function Chart (SFC)

Es particularmente útil para describir funciones de control secuencial. Como punto de partida toma la norma GRAFCET IEC 848.

- Etapas representadas gráficamente por un bloque o literlamente mediante una instrucción común a los lenguajes IL y ST: STEP.....END_STEP
- Transiciones representadas gráficamente por una línea horizontal o literalmente mediante la instrución: TRANSITION.....END_TRANSITION
- Condición de transición programable en lenguaje LD, FBD, IL o ST.
- Acciones asociadas a las etapas: variables booleanas o un segmento de programa escrito en uno de los cinco lenguajes.
- Asociación entre acciones y etapas de forma gráfica o literal.
- Propiedades (calificaciones) de acción que permiten temporizar la acción, crear pulsos, memorizar...

Antes de concluir con esta pequeña descripción de la norma es necesario comentar dónde están los límites de la misma:

- Implementaciones no fijadas: nombre de tareas, tamaño del editor gráfico según elección del usuario, etc.

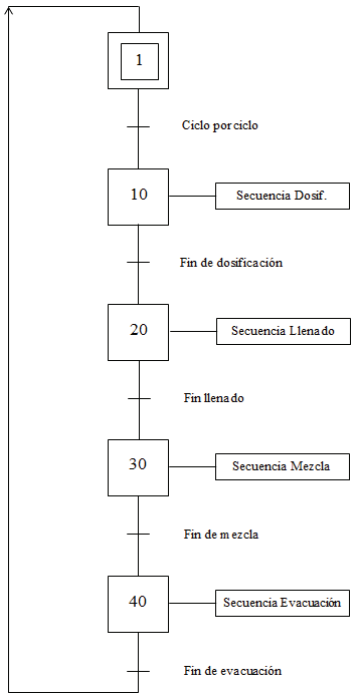

Figura 2.2.5: Ejemplo de programación SFC.

- No hay mínimo subconjunto a implementar, pero si se utiliza el nombre de un elemento de la norma (bloque función, ...), deberá tener las funciones asociadas.
- Servicios y herramientas de desarrollo y depuración de una aplicación (editores, lenguajes, documentación, etc.) no definidas.
- Ninguna regla precisa de operación, para los bolques de función por ejemplo.
- Reversibilidad entre lenguajes no descrita (LD a IL, FBD a ST, ...).
- Aún no existe una certificación IEC 1131-3. Todos los fabricantes que cumplan la norma requerida deberán adjuntar las tablas de conformidad en su documetnación, así como una lista de posibles extensiones.

2.2.1. IEC 1131 - Parte 7: Programación Control Borroso

Esta parte de la norma IEC 1131 define un lenguaje de programación para aplicaciones con control borroso que usen autómatas programables.

La teoría y los sistemas existentes realizados en el área de Control Borroso difieren ampliamente en cuanto a terminología (definiciones), características (funcionalidades) e implementación (herramientas).

El objetivo de esta norma es ofrecer a los fabricantes y los usuarios una comprensión bien definida de los medios para integrar aplicaciones de control borroso en los

lenguajes para autómatas programables de acuerdo con la Parte 3, así como la posibilidad de intercambiar programas de control borroso entre los diferentes sistemas de programación.

Para lograr esto, la norma da una breve introducción a la teoría de control borroso en la medida en que sea necesario para la comprensión de la norma. De esta forma aquellas personas que no estén familiarizadas con la teoría de control borroso podrán entender la norma.

La terminología estándar se define en el apartado 3, la integración de la aplicación para Control Borroso en los lenguajes estándar de autómatas programables en el apartado 4, y un conjunto de tres niveles de funciones con su funcionalidad y la representación textual en el apartado 5. Este apartado 5 define la sintáxis y la semántica de las características estandarizadas en forma de lenguaje de control borroso textual (FCL *Fuzzy Control Language*) con las definiciones de la Parte 3 de la norma (lenguajes de programación) al igual que los tipos de datos y módulos de función.

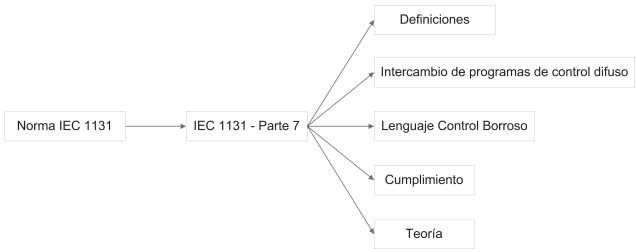

Figura 2.2.6: Esquema contenido de la norma IEC 1131-7.

El control borroso se utiliza desde en pequeñas y sencillas aplicaciones hasta en proyectos muy sofisticados y complejos. Para cubrir todo tipo de uso, en esta parte de la norma IEC 1131, se asignan las características de un sistema de control borroso compatible a las definiciones de clases de cumplimiento descritas en el apartado 6.

La Clase Base define un conjunto mínimo de caracterísitcas que tienen que lograrse por todos los sitemas compatibles. Esto facilita el intercambio de programas de control borroso.

Se definen características opcionales estándar en la Clase de Extensión. Programas de control borroso que aplican estas características sólo se pueden intercambiar entre sistemas que utilicen el mismo conjunto de características, si no únicamente es posible un cambio parcial. La norma no obliga a todos los sistemas compatibles a realizar todas las funciones de la Clase de Extensión, pero soporta la posibilidad de portabilidad (parcial) y la posibilidad de evitar el uso de características no estándar. Por lo tanto, un sistema compatible no debe ofrecer características no estándar que puedan ser significativamente realizadas mediante el uso de características estándar de la Clase Base y la Clase Extendida.

Con el fin de no excluir a los sistemas que utilizan sus propias características sofisticadas del cumplimiento de esta norma y no obstaculizar el progreso del desarrollo futuro, la norma permite también otras características no estándar que no están cubiertas por la Clase Básica y la Extendida. Sin embargo, estas características deben estar inscritas en el método estándar para asegurar que son fáciles de reconocer como características estándar.

La portabilidad de las aplicaciones de control borroso depende de los sistemas de programación diferentes y también de las características de los sitemas de control. Estas dependencias están cubiertas por la lista de verificación de datos entregada por el fabricante.

2.2.1.1. Definiciones

Algunas de las definiciones que aparecen en esta parte de la norma (que no aparecen en los elementos del lenguaje de la Parte 3):

- Acumulación (o resultado de la agregación): combinación de los resultados de las reglas lingüísticas en un resultado final.
- Agregación: cálculo del grado de cumplimiento de la condición de una regla.
- Activación[2]: el proceso por el cual el grado de cumplimiento de una condición actúa sobre un conjunto borroso de salida.
- Conclusión[3]: la salida de una regla lingüística, es decir, las acciones que se deben tomar (la parte THEN de un IF...THEN de la regla de control borroso).
- Condición: una composición de términos lingüísticos que forman la parte IF de una regla.
- Conjunto clásico: un conjunto clásico es un caso especial de un conjunto borroso, en el cual la función de pertenencia sólo toma dos valores, comúnmente definidas como 0 y 1.
- Defuzzificación: conversión de un conjunto borroso en un valor numérico.
- Grado de pertenencia: el valor de la función de pertenencia.
- Fuzzificación: conversión de un valor de entrada en los grados de pertenencia de las funciones de pertenencia definidas en la variable que toma este valor.
- Control Borroso: es un tipo de control en el que el algoritmo de contol se basa en lógica borrosa.
- Lógica Borrosa: colección de las teorías matemáticas basadas en la noción de un conjunto borroso. El Control Borroso es una rama del control multievaluado.
- Operador de control borroso: el operador utilizado en la teoría de la lógica borrosa.

[2]También conocido como composición.
[3]También conocido como consecuente.

- Conjunto borroso: un conjunto borroso A se define como el conjunto de pares ordenados (x, μA(x)), donde x es un elemento del universo de discurso U y μA(x) es la función de pertenencia, que atribuye a cada xϵU un número real ϵ[0, 1], que describe el grado en el que x pertenece al conjunto.
- Inferencia: aplicación de reglas lingüísticas en valores de entrada con el fin de generar valores de salida.
- Regla lingüística: la regla IF-THEN con la condición y la conclusión, uno o los dos al menos lingüística.
- Término lingüístico: en el contexto de términos lingüísticos de control borroso están definidos por conjuntos borrosos.
- Variable lingüística: variable que toma valores en el rango de términos lingüísticos.
- Función de pertenencia: una función que expresa en qué grado un elemento de un conjunto pertenece a un subconjunto borroso dado.
- Singleton: un singleton es un conjunto borroso cuya función de pertenencia es igual a uno en un momento dado e igual a cero en todos los demás.
- Base de reglas: colección de reglas lingüísticas para alcanzar determinados objetivos.
- Factor de ponderación: valor entre 0...1, que indica el grado de importancia, credibilidad, confianza de una regla lingüística.

Algunas de estas definiciones se han explicado con anterioridad en este capítulo.

2.2.1.2. Integración en autómatas programables

Las aplicaciones de control borroso programadas en FCL de acuerdo con el apartado 5 de esta parte de la norma deben ser encapsuladas en bloques de función (o programas) tal como se define en la IEC 1131 Parte 3, lenguajes de programación. El concepto de tipos de bloques de funciones e instancias de bloques de función en la Parte 3 se aplican a esta parte.

Los tipos de bloques de funciones definidos en FCL especificarán los parámetros de entrada y salida y las reglas específicas y declaraciones del control borroso. Las correspondientes instancias de los bloques de funciones contendrán los datos específicos de las aplicaciones de control borroso. Los bloques de funciones definidas en FCL pueden utilizarse en los programas y bloques de funciones escritas en cualquiera de los lenguajes de la Parte 3.

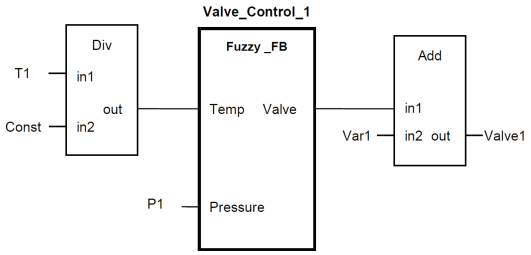

Figura 2.2.7: Ejemplo de un bloque de función de control borroso en representación FBD.

2.2.1.3. Lenguaje de control borroso FCL (*Fuzzy Control Language*)

La definición del lenguaje de control borroso (FCL) se basa en las definiciones de los lenguajes de programación de la Parte 3. El algoritmo de control borroso es externamente representado como un bloque de funciones de acuerdo con la Parte 3. Los elementos necesarios para la descripción de las partes internas lingüísticas del bloque de control de la función borrosa como funciones de pertenencia, reglas, operadores y métodos tienen que ser definidos de acuerdo con el apartado 5.

Los elementos del lenguaje de FCL estandarizan una respresentación común para el intercambio de datos entre herramientas de configuración de control borroso de los diferentes fabricantes.

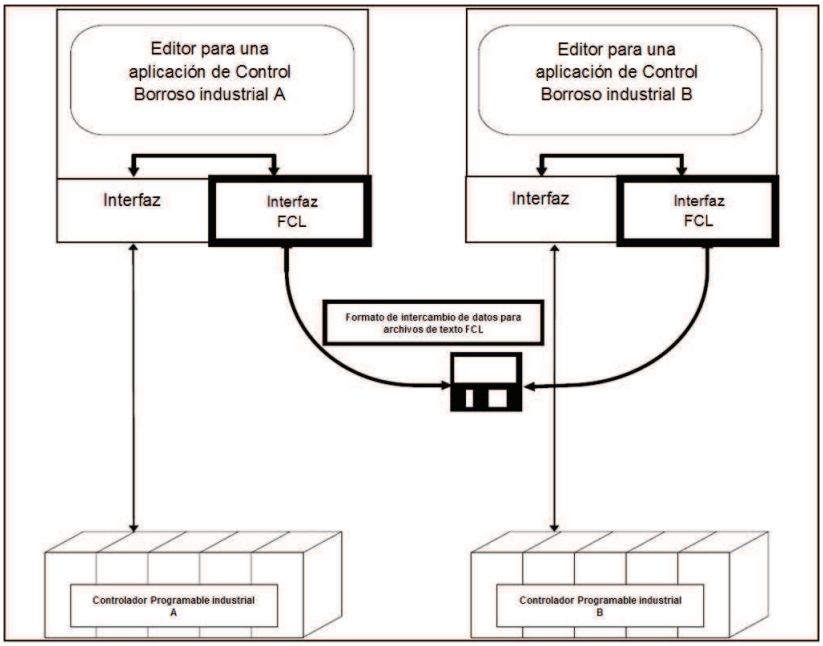

Figura 2.2.8: Intercambio de datos para de programas en lenguaje de Control Borroso (FCL).

Utilizando esta representación común de todos los fabricantes de autómatas programables pueden mantener su hardware, los editores de software y compiladores. El fabricante, solo tiene que implementar la interfaz de datos en su editor específico. El cliente podría intercambiar proyectos de control borroso entre los diferentes

fabricantes.

Algunos ejemplos de elementos del lenguaje de control borroso son los siguientes:

- Según el apartado 4 la vista externa de un bloque de función borroso requiere la utilización de los elementos del lenguaje estándar de la Parte 3.
- Los valores de las variables de entrada tienen que ser convertidos en grados de pertenencia de las funciones de pertenencias definidas en la variable. Esta conversión se describe entre las palabras clave: FUZZIFY y END_FUZZIFY.
- Una variable lingüística para una variable de salida tiene que ser convertido en un valor. Esta conversión se describe entre las palabras clave: DEFUZZIFY y END_DEFUZZIFY.
- La inferencia borrosa del algoritmo se define en uno o más bloques de reglas. Para la manipulación y para atender a la posibilidad de dividir la base de reglas en diferentes módulos, es posible el uso de varios bloques de reglas. Cada bloque de reglas tiene un único nombre. Las reglas se definen entre las palabras clave: RULERBLOCK y END_RULEBLOCK.
- Para la aplicación en diferentes sistemas de destino, puede ser necesario porporcionar información adicional al sistema a fin de permitir la mejor conversión posible de aplicaciones de control borroso. Esta información adicional puede ser necesaria en un elemento de lenguaje encerrado por OPTIONS y END_OPTIONS.

```
FUNCTION_BLOCK Fuzzy_FB
VAR_INPUT
        temp :      REAL;
        pressure :  REAL;
END_VAR
VAR_OUTPUT
        valve :     REAL;
END_VAR
FUZZIFY temp
        TERM cold := (3, 1)  (27, 0);
        TERM hot  := (3, 0  (27, 1);
END_FUZZIFY
FUZZIFY pressure
        TERM low := (55, 1) (95, 0);
        TERM high:= (55, 0) (95, 1);
END_FUZZIFY
DEFUZZIFY valve
        TERM drainage    := -100;
        TERM closed      := 0;
        TERM inlet       := 100;
        ACCU : MAX;
        METHOD : COGS;
        DEFAULT := 0;
END_DEFUZZIFY
RULEBLOCK No1
        AND : MIN;
        RULE 1 : IF temp IS cold AND pressure IS low THEN valve IS inlet
        RULE 2 : IF temp IS cold AND pressure IS high THEN valve IS closed WITH 0.8;
        RULE 3 : IF temp IS hot AND pressure IS low THEN valve IS closed;
        RULE 4 : IF temp IS hot AND pressure IS high THEN valve IS drainage;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

Figura 2.2.9: Ejemplo de bloque de función borroso.

2.2.1.4. Cumplimiento

Los niveles de conformidad para el sistema de control utilizando el lenguaje de control borroso (FCL) se muestra en la Figura 2.2.10.

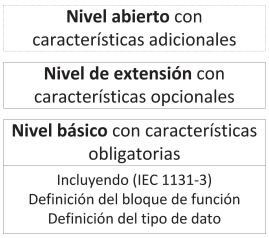

Figura 2.2.10: Niveles de conformidad.

Un sistema de control que use el lenguaje de control borroso (FCL) y quiera cumplir esta parte de la norma deberá lograr las siguientes reglas:

1. Será capaz de utilizar las características de los bloques de funciones de acuerdo con la Parte 3 de esta norma, a fin de realizar las funciones de control borroso. Por lo tanto, la definición de los bloques de función y los tipos de datos necesarios para los parámetros de entrada y de salida del bloque de funciones se hará de conformidad con la Parte 3.

2. Todas las características de funcionalidad de control borroso se llevarán a cabo de acuerdo con las definiciones de esta parte. Las características definen el conjunto de los elementos de Nivel Básico que todo los sistemas estándar tendrán en común.

3. Un subconjunto de los elementos del Nivel Extendido son elementos adicionales que pueden aplicarse opcionalmente. La aplicación será exactamente de acuerdo con las definiciones de esta parte. Estás características serán marcadas como extensiones estándar y una lista de las funciones realizadas formará parte de la documentación del sistema.

4. Otras características que excedan el Nivel Básico y el Nivel Extendido podrán realizarse siempre que estas características no tengan la misma o parecida funcionalidad o representación de las características estándar, lo que evita cualquier posible confusión. Estas características serán marcadas como características de Nivel Abierto y una lista formará parte de la documentación del sistema.

5. El intercambio de programas de aplicacion entre los diferentes sistemas de control borroso se hará en la forma textual del lenguaje de control borroso (FCL) de acuerdo con las definiciones contenidas en esta parte. Este formato se pondrá a disposición de los sistemas estándar como formatos de entrada y de salida.

6. Con el fin de lograr una interfaz de usuario más cómoda y adecuada y que no impida el progreso, las representaciones externas para el diseño, entrada, pruebas, etc. de programas de aplicación de control borroso pueden realizarse por cualquier medio grafico o textual.

Se entregará una lista de verificación de datos en la documentación técnica. En esta lista el fabricante de sistemas de automatización, herramientas de programación borrosa y software de aplicación describe las características de funcionamiento específicas de su sistema de control borroso.

2.3. Fuzzy Library

Para el uso del control borroso con Unity Pro, esta librería proporciona funciones elementales (*Elementary functions* EFs) y bloques de función (*function blocks* EFBs). Esta librería nos permitirá implementar técnicas de control borroso en autómatas programables de Schneider Electric como el PLC M340 con el que trabajamos en este proyecto. Utilizaremos la Fuzzy Control Library Version 1.2.

A continuación se describirán los diferentes tipos de bloques y sus aplicaciones que se utilizan en Unity Pro:

- Función elemental (EF).
- Bloque de función elemental (EFB).
- Bloque de función derivado (DFB).
- Procedimiento.

Función elemental

Las funciones elementales (EF) no tienen estados internos y una salida única. Si los valores de entrada son los mismos, el valor de salida es el mismo para todas las ejecuciones de la función. Una función elemental se representa en los lenguajes gráficos (FBD y LD) como un bloque con marco con entradas y una salida. Las entradas se representan siempre a la izquierda y las salidas siempre a la derecha del marco. El nombre de la función, es decir el tipo de función, se muestra en el centro del marco. El número de entradas se puede incrementar con algunas funciones elementales.

Bloque de función elemental

Los bloques de funciones elementales (EF) tienen estados internos. Si las entradas tienen los mismos valores, el valor de las salidas puede tener otro valor durante distintas ejecuciones. Un bloque de función elemental se representa en los lenguajes gráficos (FBD y LD) como un bloque de datos de entradas y salidas. Las entradas están siempre representadas en la izquierda y las salidas siempre a la derecha del marco. El nombre del bloque de función, es decir, el tipo de bloque de función, se muestra en el centro del marco. El nombre de la instancia aparece por encima del marco.

Bloque de función derivado

Los bloques de funciones derivados (DFB) presentan las mismas propiedades que los bloques de funciones elementales. Son creados por el usuario en los lenguajes de programación FBD, LD, IL y/o ST.

Procedimiento

Los procedimientos son funciones con varias salidas. No tienen ningún estado interno. La única diferenca con las funciones elementales es que los precedimientos pueden tener más de una salida y admiten variables del tipo de datos VAR_IN_OUT. Los procedimiento no devuelven un valor. Los procedimientos son una ampliación de la norma IEC 61131-3 y se deben habilitar de forma explícita. No hay diferencia visual entre los procedimientos y funciones elementales.

Esta biblioteca de funciones porporcionada por Unity Pro nos permite utilizar la lógica borroso en autómatas programables.

Operación	**Función**	**Familia**	**Descripción**
Borrosificación	FUZ_STERM	Borrosificador (*Fuzzify*)	Borrosificación de un término
Borrosificación	FUZ_ATERM	Borrosificador (*Fuzzify*)	Borrosificación de hasta 9 términos de una sola vez
Borrosificación	FUZ_ATERM_STI, FUZ_ATERM_STR	Esctructura de borrosificador (*Fuzzify_Struct*)	Borrosificación de hasta 9 términos de una sola vez. Almacena el reslutado en una estructura de datos.
Inferencia	FUZ_MAX	Operadores OR	Operador OR: Máximo
Inferencia	FUZ_MIN	Operadores AND	Operador AND: Mínimo
Inferencia	FUZ_SUM	Operadores OR	Operador OR: Suma
Inferencia	FUZ_PROD	Operadores AND	Operador AND: Producto
Desborrosificación	DEFUZ	Desborrosificador (Defuzzify)	Desborrosificación con singletones
Desborrosificación	DEFUZ_STI, DEFUZ_STR	Estructura de desborrosificador (Defuzzify_Struct)	Desborrosificación con singletones. Entradas obtenidas de estructuras de datos.

Cuadro 2.2: Funciones para aritmética INT y REAL en la biblioteca de control borroso.

La diferencia entre aritmética entera y real es:

- La solución dentro del cálculo.
  - La solución del grado de pertenencia es 0.01 %.
  - El rango de los grados de pertenencia 0...1 se escala a 0...10000.
- La posibilidad de trabajar con los valores físicos en aritmética real.
- El tiempo de ejecución, que es más largo que en la aritmética real.

Como se puede observar en la Tabla 2.2, en Unity Pro se tiene la posibilidad de seleccionar un borrosificador para cada término de una variable o un borrosificador de hasta 9 términos de una varible. El borrosificador de Unity Pro soporta funciones de pertenencia de hasta 4 puntos (funciones de tipo trapezoidal, rectangular, triangular, singleton, etc.).

El borrosificador de un término individual ofrece una mayor flexibilidad para los usuarios más experimentados en lógica borrosa, pero la forma más facil de borrosificar una variable es utilizar FUZ_ATERM o FUZ_ATERM_ST, ya que estos

bloques borrosifican todos los términos de una variable borrosa. La idea detrás de esta simplificación es que las diferentes funciones de pertenencia de todos los términos de una variable borrosa no son independientes unos de otros. En la mayoría de los casos, el total de los grados de pertenencia de dos funciones sucesivas es 1.

Las reglas se realizan a través de la concatenación de los grados de pertenencia de las variables borrosas. Las entradas de los bloques, que representan los operadores borrosos, se combinan con los grados de pertenencia. Las parejas FUZ_MIN/FUZ_MAX y FUZ_PROD/FUZ_SUM han demostrado ser buenas combinaciones para la concatenación AND/OR.

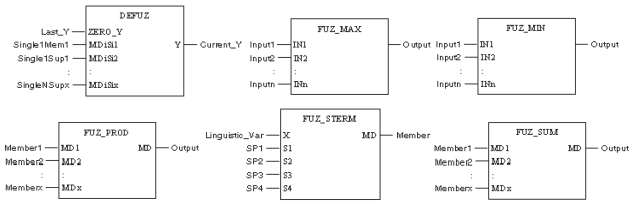

Figura 2.3.1: Ejemplos de bloques de funciones en representación FBD.

2.4. Fuzzy Logic Toolbox™

Un rápido crecimiento en la industria de la tecnología de la información ha desarrollado y lanzado un gran número de paquetes de diseño que se utilizan con éxito en diferentes aplicaciones para el diseño de controloadores borrosos. Entre ellos se encuentra la Fuzzy Logic Toolbox para Matlab® de MathWorks Inc. Esta *toolbox* la utilizaremos durante el desarrollo de este proyecto para probar y simular técnicas de control borroso y aunque se detallará en capítulos posteriores el manejo de la misma, en este capítulo se hará una introducción a la herramienta.

La Fuzzy Logic Toolbox™ proporciona funciones de Matlab, herramientas gráficas y un bloque de Simulink® para analizar, diseñar y simular sistemas basados en lógica borrosa.

La *toolbox* nos permite modelar comportamientos de sistemas complejos usando reglas de lógica simple e implementar estas reglas en un sistema de inferencia borrosa. Podemos usarlo como un motor de inferencia borrosa independiente. Alternativamente, podemos usar los bloques de inferencia borrosa en Simulink y simular los sistemas borrosos con un modelo integral del sistema dinámico completo.

Características principales:

- GUIs especializadas para la construcción de sistemas de inferencia borrosa y para ver y analizar los resultados.
- Funciones de pertenencia para la creación de sistemas de inferencia borrosa.

- Soporte de lógica AND, OR y NOT en las reglas definidas por el usuario.
- Sistemas de inferencia borrosa de tipo Mamdani y Sugeno estándar.
- Formación de la función de pertenencia automatizada a través de técnicas de aprendizaje neuroadaptativos y de agrupación borrosa.
- Capacidad para integar un sistema de inferencia borrosa en un modelo Simulink.
- Capacidad para generar código C embebido o ejecutables independientes de motores de inferencia borrosa.

Podemos crear y editar sistemas de inferencia borrosa con el software Fuzzy Logic Toolbox. Se pueden crear estos sistemas utilizando herramientas gráficas o funciones de comando en línea, o se pueden generar automáticamente usando técnicas de agrupación o neuro-borrosas adaptativas.

Si se tiene acceso al software Simulink, se puede probar facilmente los sistemas borrosos en un ambiente de simulación de diagramas de bloques. La Fuzzy Logic Toolbox puede utilizarse de manera independiente o junto al entorno de desarrollo Simulink.

La *toolbox* también nos permite ejecutar nuestros propios programas en C directamente. Esto es posible gracias a un Motor de Inferencia Borroso independiente que lee los sistemas borrosos guardados en una sesión MATLAB. Se puede personalizar el motor independiente para construir una inferencia borrosa en nuestro código. Todo el código proporcionado es compatible con ANSI®.

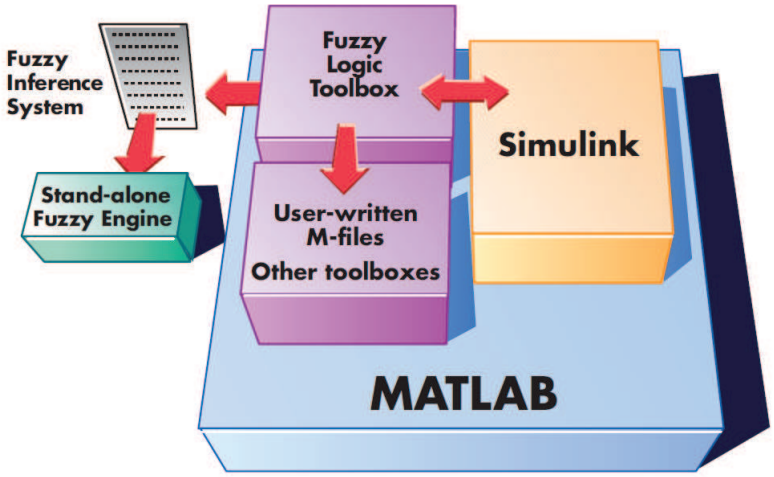

Figura 2.4.1: Diagrama del funcionamiento de la Fuzzy Logic Toolbox

Debido a la naturaleza integrada del entorno de MATLAB, podemos crear nuestrar propias herramientas para personalizar la *toolbox* o aprovecharla con otra *toolbox*, tales como el software Control System Toolbox™, Neural Network Toolbox™ o Optimization Toolbox™.

La Fuzzy Logic Toolbox es una ampliación del entorno de desarrollo MATLAB para el diseño de sistemas basados en lógica borrosa.

Ofrece un entorno gráfico muy sencillo para el desarrollo de este tipo de sistemas paso a paso. Esta *toolbox* facilita el diseño de las reglas y la selección de las funciones más utilizadas en lógica borrosa. Mediante las interfaces gráficas de usuario (GUIs) podemos construir, editar y ver un sistema de inferencia borroso (FIS):

- El Editor de Sistemas de Inferencia Borrosos permite manejar cuestiones de niveles altos para los sistemas. La *toolbox* no limita el número de entradas por ejemplo. Sin embargo, el número de entradas puede estar limitado por la memoria disponible del ordenador. Si el número de entradas es demasiado alto, o el número de funciones de pertenencia es demasiado grande, también puede dificultar el análisis del FIS usando otras herramientas gráficas.
- El Editor de Funciones de Pertenencia define la forma de todas las funciones de pertenencia asociadas a cada variable.
- El Editor de Reglas permite editar la lista de reglas que definen el comportamiento del sistema.
- El Visor de Reglas permite visualizar el diagrama de inferencia borroso. Se puede utilizar para realizar diagnósticos, por ejemplo, para ver qué reglas estan activas, o cómo las distintas formas de las funciones de pertenencia influyen en los resultados.
- El Visor de Superficie permite ver la dependencia de una de las salidas en una o dos de las entradas.

Estas GUIs están vinculadas dinámicamente, los cambios que se realicen en el FIS, afectan a lo que se ve en cualquiera de las otras GUIs abiertas. Por ejemplo, si cambia los nombres de las funciones de pertenencia en el Editor de Función de Pertenencia, los cambios se reflejan en las reglas que se muestran en el Editor de Reglas. Se puede utilizar la GUI para leer y escribir variables tanto para el espacio de trabajo (*Workspace*) de MATLAB como en un archivo . Se pueden tener todas o ninguna de las interfaces abiertas para cualquier sistema o tener multiles editores abiertos para cualquier número de sistemas FIS.

La Figura 2.4.3 muestra cómo trabajan los componentes principales de un FIS y los tres editores.
Mientras, los dos visores examinan el comportamiento del sistema entero.

Además de todo lo mostrado anteriormente, otra de las ventajas de utilizar MATLAB y sus herramientas, es que nos permite construir nuestras propias funciones de pertenencia, funciones de inferencia, etc. De este modo tenemos un control total de nuestro sistema borroso a desarrollar.

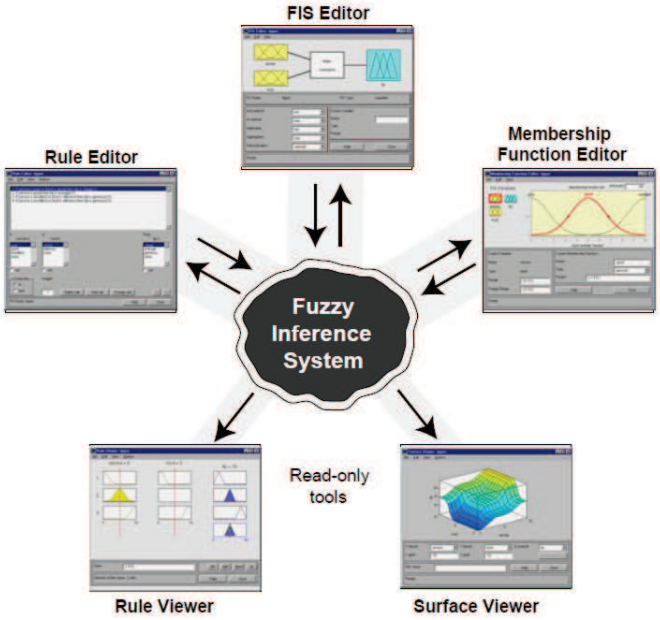

Figura 2.4.2: Herramientas de la interfaz gráfica de usuario de la Fuzzy Logic Toolbox.

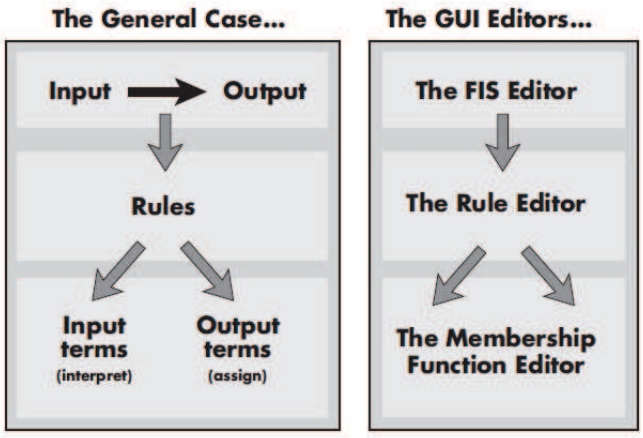

Figura 2.4.3: Funcionamiento de los tres editores.