INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC)

TECHNICAL COMMITTEE No. 65: INDUSTRIAL PROCESS MEASUREMENT AND CONTROL

SUB-COMMITTEE 65 B: DEVICES

IEC 1131 - PROGRAMMABLE CONTROLLERS

Part 7 - Fuzzy Control Programming

Committee Draft CD 1.0 (Rel. 19 Jan 97)

Contents

Fehler! Textmarke nicht definiert.

Foreword

A New Working Item Proposal (NWIP) has been submitted for standardisation of the programming methodology, environment and functional characteristics of Fuzzy Control in the area of Programmable Controllers; IEC 65B (U.S.A.) 60, July 1993.

The result of the international voting on the NWIP was the approval and the introduction in the IEC work program under the title preliminary “Fuzzy Control Standardisation“; IEC 65B (Secretariat) 194, February 1994.

Since this emerging technology is being included in today’s Programmable Controllers design this project was assigned by the IEC Sub-Committee SC65B (Devices) to its Working Group IEC 65B/WG7 (Programmable Controllers); IEC TC65B (Milan), meeting October 1994.

This Comittee Draft (CD 1.0) was elaborated by an international Task Force (TF8) of IEC/65B/WG7 in Frankfurt, Germany, December 6, 1996. It is planned to become Part 7 of the International Standard IEC 1131 “Programmable Controllers“.

Introduction

The theory of Fuzzy Logic in the application of control is named Fuzzy Control. The Fuzzy Control is emerging as a technology that can enhance the capabilities of industrial automation, and is suitable for control level tasks generally performed in Programmable Controllers (PC).

Fuzzy Control is based upon practical application knowledge represented by so called linguistic rule bases, rather than by analytical (either empirical or theoretical) models. Fuzzy Control can be used when there is an expertise that can be expressed in its formalism. That allows to take opportunity of available knowledge to improve processes and perform a variety of tasks, for instance

- control (closed or open loop, single or multi-variable, for linear or non linear systems)
- on-line or off-line setting of control systems’ parameters
- classification and pattern recognition
- real-time decision making (send this product to machine A or B ?)
- helping operators to make decisions or tune parameters
- detection and diagnosis of faults in systems

Its wide range of applications and natural approach based on human experience makes Fuzzy Control a basic tool that should be made available to Programmable Controller users as a standard.

Fuzzy Control can also in a straightforward way be combined with classical control methods.

The application of Fuzzy Control can be of advantage in such cases where there is no explicit process model available, or in which the analytical model is too difficult to evaluate (e.g. multiple input multiple output systems) or when the model is too complicated to evaluate it in real time.

Another advantageous feature of Fuzzy Control is, that human experience can be incorporated in a straightforward way. Also it is not necessary to model the whole controller with Fuzzy Control: sometimes Fuzzy Control just interpolates between a series of locally linear models, or dynamically adapts the parameters of a "linear controller", thereby rendering it non linear, or alternatively just "zoom in" onto a certain feature of an existing controller that needs to be improved.

Fuzzy Control is a multi-valued Control, no longer restricting the values of a Control proposition to "true" or "false". This makes Fuzzy Control particularly useful to model empirical expertise, stating, which control actions have to be taken under a given set of inputs.

The existing theory and systems already realised in the area of Fuzzy Control differ widely in terms of terminology (definitions), features (functionalities) and implementation (tools).

The goal of this Standard is to offer the manufactures and the users a well defined common understanding of the basic means to integrate Fuzzy Control applications in the Programmable Controller languages according to Part 3, as well as the possibility to exchange portable Fuzzy Control programs among different programming systems.

To achieve this, Annex A of the Standard gives a short introduction to the theory of Fuzzy Control and Fuzzy Control as far as it is necessary for the understanding of the Standard. It may be helpful for readers of this Standard which are not familiar with Fuzzy Control theory to read the Annex A first.

The standard terminology is defined in clause 3, the integration of Fuzzy Control application into the Standard Languages of Programmable Controllers in clause 4, and a three level set of features with their functionality and textual representation in clause 5. This clause 5 defines the syntax and the semantics of the standardised features in form of the textual Fuzzy Control Language (FCL) using the definitions from Part 3 (Programmable Controller Languages) like Data Types and Function Blocks.

Fuzzy Control is used from small and simple applications up to highly sophisticated and complex projects. To cover all kinds of usage by this Part the features of a compliant Fuzzy Control system are mapped into defined conformance classes described in clause 6.

The Basic Class defines a minimum set of features which has to be achieved by all compliant systems. This facilitates the exchange of Fuzzy Control programs.

Optional standard features are defined in the Extension Class. Fuzzy Control programs applying these features can only be fully ported among systems using the same set of features, otherwise a partial exchange may be possible only. The Standard does not force all compliant systems to realise all features in the Extension Class, but it supports the possibility of (partial) portability and the avoidance of the usage of non-standard features. Therefore a compliant system should not offer non-standard features which can be meaningfully realised by using standard features of the Basic Class and the Extension Class.

In order not to exclude systems using their own highly sophisticated features from complying with this Standard and not to hinder the progress of future development, the Standard permits also additional non-standard features which are not covered by the Basic Class and the Extension. However, these features need to be listed in a standard way to ensure that they are easily recognised as non-standard features.

The portability of Fuzzy Control applications depends on the different programming systems and also the charcteristics of the control systems. These dependencies are covered by the Data Check List to be delivered by the manufacturer.

1 Scope

This part of IEC 1131 defines a language for programming of Fuzzy Control applications which use programmable controllers.

2 Normative references

The following standard documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 1131. At the time of publication, the editions indicated below were valid. All such documents are subject to revision. Parties to agreements based on this part of IEC 1131 are therefore encouraged to apply the current editions of standards. Members of IEC and ISO maintain registers of current standard documents.

IEC 1-1581-FDIS 1996, International Electrotechnical Vocabulary, clause 351: Automatic Control
IEC 1131 Part 1: 1992, General information
IEC 1131 Part 2: 1992, Equipment requirements and tests
IEC 1131 Part 3: 1993, Programming languages
IEC 1131 Technical Report, Guidelines for users and implementers of IEC 1131-3

3 Definitions

Further definitions for language elements are given in Part 3.

3.1 **accumulation** [or **result aggregation**]: Combination of results of *linguistic rules* in a final result.

3.2 **aggregation**: Calculation of the degree of accomplishment of the *condition* of a rule.

3.3 **activation**: The process by which the degree of fulfilment of a *condition* acts on an output fuzzy set.

NOTE - Also known as composition.

3.4 **conclusion:** The output of a *linguistic rule*, i.e. the actions to be taken (the THEN part of an IF..THEN Fuzzy Control rule).

NOTE - Also known as consequent.

3.5 **condition**: A composition of *linguistic terms* forming the IF-part of a rule.

3.6 **crisp set**: A crisp set is a special case of a *Fuzzy set*, in which the *membership function* only takes two values, commonly defined as 0 and 1.

3.7 **defuzzification**: Conversion of a *Fuzzy set* into a numerical value.

3.8 **degree of membership**: membership function value.

3.9 **fuzzification**: Conversion of an input value into degrees of membership for the membership functions defined on the variable taking this value.

3.10 **Fuzzy Control**: A type of control in which the control algorithm is based on *Fuzzy Logic* (IEV 351-07-51 modified)

3.11 **Fuzzy Logic**: Collection of mathematical theories based on the notion of *Fuzzy set*. *Fuzzy Control* is a branch of multi-valued Control.

3.12 **Fuzzy Control operator**: Operator used in Fuzzy Logic theory

3.13 **Fuzzy set**: A *Fuzzy set* A is defined as the set of ordered pairs $(x,\mu A(x))$, where x is an element of the universe of discourse U and $\mu A(x)$ is the *membership function*, that attributes to each $x \in U$ a real number $\in [0,1]$, describing the degree to which x belongs to the set.

3.14 **inference**: Application of *linguistic rules* on input values in order to generate output values

3.15 **linguistic rule**: IF-THEN rule with *condition* and *conclusion*, one or both at least linguistic.

3.16 **linguistic term**: In the context of Fuzzy Control *linguistic terms* are defined by *Fuzzy sets*

3.17 **linguistic variable**: Variable that takes values in the range of *linguistic terms*.

3.18 **membership function**: A function which expresses in which degree an element of a set belongs to a given Fuzzy subset (see IEV 351-07-52).

NOTE - Also known as antecedent.

3.19 **singleton**: A singleton is a *Fuzzy set* whose *membership function* is equal to one at one point and equal to zero at all other points.

3.20 **rule base**: Collection of *linguistic rules* to attain certain objectives.

3.21 **weighting factor**: Value between 0..1, that states the degree of importance, credibility, confidence of a linguistic rule.

4 Integration into the Programmable Controller

The Fuzzy Control applications programmed in Fuzzy Control Language FCL according to clause 5 of this Part of the Standard shall be encapsulated in Function Blocks (or Programs) as defined in IEC1131 Part 3, Programming Languages. The concept of Function Block Types and Function Block Instances given in Part 3 apply to this part.

The Function Block Types defined in Fuzzy Control Language FCL shall specify the input and output parameters and the Fuzzy Control specific rules and declarations.

The corresponding Functions Block Instances shall contain the specific data of the Fuzzy Control applications.

Function Blocks defined in Fuzzy Control Language FCL can be used in Programs and Function Blocks written in any of the languages of Part 3, e.g. Ladder Diagram, Instruction List, etc. The data types of the input and output parameters of the Function Block or Program written in FCL shall match those of the corresponding “calling environment“ as illustrated in Figure 4-1.

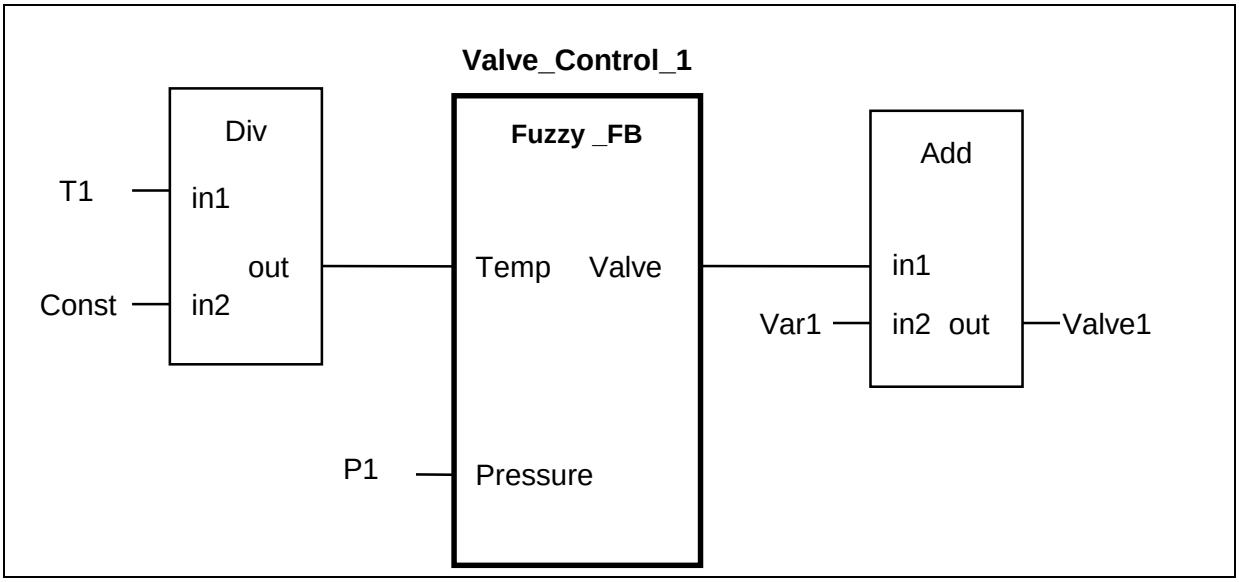


Figure 4-1: Example of a Fuzzy Control Function Block in FBD representation

In this example Valve_Control_1 is a user defined Function Block Instance of the Function Block Type Fuzzy_FB. The Function Block Type Fuzzy_FB may be programmed in Fuzzy Control Language FCL according to clause 5 of this Part. The Function Block Fuzzy_FB is used here in a program or a Function Block which is represented in the graphical language FBD (Function Block Diagram) of Part 3.

5 Fuzzy Control Language FCL

5.1 Exchange of Fuzzy Control programs

The definition of the Fuzzy Control Language FCL is based on the definitions of the programming languages in Part 3. The interaction of the Fuzzy Control algorithm with its program environment causes it to be “hidden“ from the program. The Fuzzy Control algorithm is therefore externally represented as a Function Block according to Part 3. The necessary elements for describing the internal linguistic parts of the Fuzzy Control Function Block like membership functions, rules, operators and methods have to be defined according to clause 5.

The language elements of FCL standardise a common representation for data exchange among Fuzzy Control configuration tools of different manufacturers shown in figure 5.1-1. Using this common representation every manufacturer of programmable controllers can keep his hardware, software editors and compilers. The manufactuer has only to implement the data interface into his specific editor. The customer would be able to exchange Fuzzy Control projects between different manufacturers.

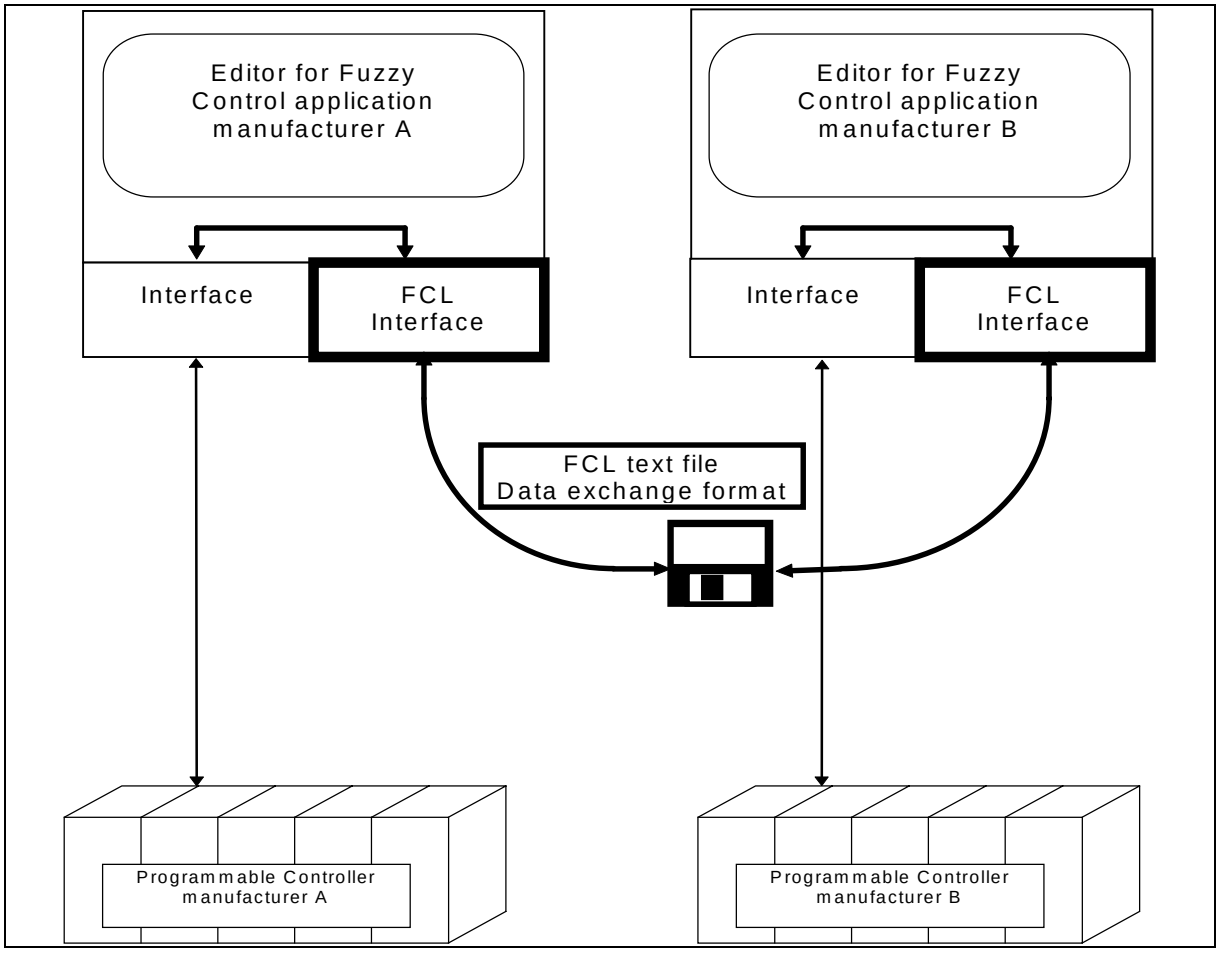


Figure 5.1-1: Data exchange of Programs in Fuzzy Control Language (FCL)

5.2 Fuzzy Control Language elements

Fuzzy control language elements in this clause are described using examples. The detailed production rule is given in clause 5.4.

5.2.1 Function Block interface

According to clause 4 the external view of the Fuzzy Function Block requires that the following standard language elements of Part 3 shall be used:

FUNCTION_BLOCK *function_block_name*	Function block
VAR_INPUT *variable_name*: *data_type*; .... END_VAR	Input parameter declaration
VAR_OUTPUT variable_name: *data_type*; .... END_VAR	Output parameter declaration
.... VAR *variable_name: data_type*; END_VAR END_FUNCTION_BLOCK	Local *variables*

With these language elements it is possible to describe a function block interface. The function block interface is defined with parameters which are passed into and out of the function block. The data types of these parameters shall be defined according to Part 3.

Figure 5.2.1-1 shows an example for a Function Block declaration in Structured Text (ST) and Function Block Diagram (FBD) languages.

<table>
<tr>
<td>FUNCTION_BLOCK Fuzzy-FB
VAR_INPUT
Temp: REAL;
Pressure: REAL;
END_VAR

VAR_OUTPUT
Valve: REAL;
END_VAR

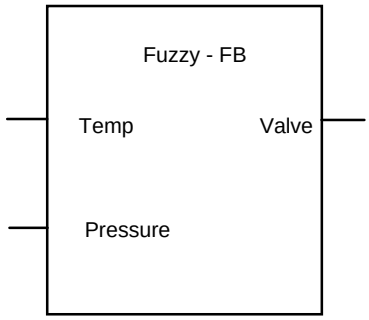
END_FUNCTION_BLOCK</td>
<td> 
</td>
</tr>
<tr>
<td>Structured Text (ST)</td>
<td>Function Block Diagram (FBD)</td>
</tr>
</table>

Figure 5.2.1-1: Example of a Function Block interface declaration in ST and FBD languages

5.2.2 Fuzzification

The values of the input variables have to be converted into *degrees of membership* for *the membership functions* defined on the variable. This conversion is described between the keywords FUZZIFY and END_FUZZIFY.

```
FUZZIFY variable_name
        TERM term_name := membership_function ;
        ....
END_FUZZIFY
```

After the keyword FUZZIFY the name of a variable which is used for the fuzzification shall be named.

This is the name of a previously defined variable in the VAR_INPUT section. This *linguistic variable* shall be described by one or more *linguistic terms*. The *linguistic terms* introduced by the keyword TERM described by *membership functions* in order to fuzzify the variable. A *membership function* is a piece-wise linear function. It is defined by a table of points.

membership_function ::= (*point i*), (*point* j), ...

Every point is a pair of the values of the variable and the membership degree of that value separated by a comma. The pairs are enclosed in parentheses and separated by commas.

point i ::= *value* of input i / *variable_name* of input i , *value* i of membership degree

With this definition all simple elements e.g. ramp and triangle can be defined. The points shall be given in ascending order of variable value. The membership function is linear between successive points. The degree of membership for each term is therefore calculated from the crisp input value by the linear interpolation between the two relevant adjacent membership function points.

The number of points can vary, but its maximum number is restricted according to the clause 6 conformance classes.

Example of *membership function* with 3 points for *linguistic term* "warm":

TERM warm := (17.5, 0.0), (20.0, 1.0), (22.5, 0.0);

If the value of a *linguistic variable* is less than the first base point in the look-up table all values below the first point in the lookup table shall have the same membership degree as defined at the first point.

If the value of a *linguistic variable* is greater than the last base point in the lookup table all values greater that the last point in the lookup table shall have the same membership degree as defined at the last point.

Definition of ramps:

```
FUZZIFY temp;
    TERM cold :=    (3, 1) (27, 0);
    TERM warm :=    (3, 0) (27, 1);
END_FUZZIFY
```

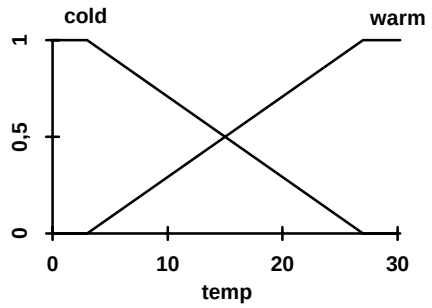


Figure 5.2.2-1: Example of ramp terms

In order to adapt the Fuzzy Control application on-line the base points in the membership functions can be modified. This can be done using variables which are input to the function block. These variables have to be declared in the VAR_INPUT section of the function block. An example for the use of variables for the definition of the points the membership functions is given in fig. 5.2.2-2.

NOTE - The values of membership functions points at runtime may be out of sequence.

```
VAR_INPUT
        temp: REAL;                         (* this input shall be fuzzified *)
        pressure: REAL;                     (* this input shall be fuzzified *)
        bp_warm1, bp_warm2 : REAL;          (* these inputs are for on-line adaptation *)
END_VAR
FUZZIFY temp
        TERM warm := (bp_warm1, 0.0), (21.0, 1.0),  (bp_warm2, 0.0);
..
END_FUZZIFY
```

Figure 5.2.2-2: Example of usage of variables for membership functions

5.2.3 Defuzzification

A *linguistic variable* for an output variable has to be converted into a value. This conversion is described between the keywords DEFUZZIFY and END_DEFUZZIFY.

After the keyword DEFUZZIFY the variable which is used for the *defuzzification* shall be named. This is the name of a previous defined variable in the VAR_OUTPUT section.

```
DEFUZZIFY variable_name
        TERM term_name := membership_function ;
        defuzzification_method ;
        default_value ;
        [range ;]
END_DEFUZZIFY
```

The definition of *linguistic terms* is given in clause 5.2.2 Fuzzification.

Singletons are special *membership functions* used for outputs in order to simplify the *defuzzification*. They are described only by a single value for the *linguistic term*. In figure 5.2.3-1 examples of terms are given.

Definition of *singletons:*

```
DEFUZZIFY valve
        TERM drainage := -100;
        TERM closed   := 0;
        TERM inlet    := 100;
END_DEFUZZIFY
```

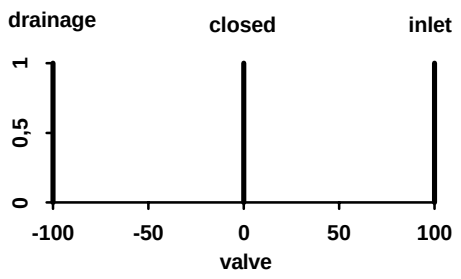


Figure 5.2.3-1: Example of singleton terms

The defuzzification method shall be defined by the language element Method.

```
METHOD :  defuzzification_method ;
```

The following defuzzification methods are possible.

Table 5.2.3-1: Defuzzification methods

Keyword	Explanation
COG	Centre of Gravity (Note 1)
COGS	Centre of Gravity for Singletons
COA	Centre of Area (Notes 2 and 3)
LM	Left Most Maximum
RM	Right Most Maximum

NOTE 1 - Centre of Gravity is equivalent to Centroid of Area

NOTE 2 - Centre of Area is equvalent to Bisector of Area

NOTE 3 - COA is not applicable if singletons are used.

Table 5.2.3-2: Formulae for Defuzzification methods

COG	$U = \frac{\int_{Min}^{Max} u\,\mu(u)\,du}{\int_{Min}^{Max} \mu(u)\,du}$
COGS	$U = \frac{\sum_{i=1}^{p} [u_i \mu_i]}{\sum_{i=1}^{p} [\mu_i]}$
COA	$U = u', \int_{Min}^{u'} \mu(u)du = \int_{u'}^{Max'} \mu(u)du$
RM	$U = \sup(u'), \mu(u') = \sup_{u \in [Min,Max]} \mu(u)$
LM	$U = \inf(u'), \mu(u') = \sup_{u \in [Min,Max]} \mu(u)$

where:

U :	result of defuzzification
u :	output variable
p :	number of singletons
μ :	membership function after accumulation
i :	index
Min :	lower limit for defuzzification
Max :	upper limit for defuzzification
sup :	largest value
inf:	smallest value

If the degree of membership is 0 for all *linguistic terms* of an output variable, that means: no rule for this variable is active. In that case the *defuzzification* is not able to generate a valid output. Therefore it is possible to define a default value for the output. This default value is the value for the output variable only in the case when no rule has fired.

DEFAULT := *value* | NC;

After the keyword DEFAULT the value shall be specified. Otherwise the key word NC (no change) shall be specified to indicate that the output shall remain unchanged if no rule has fired.

The range is a specification of a minimum value and a maximum value separated by two points.

RANGE := (minimum *value* .. maximum *value*);

The range is used for the specification of minimum and maximum values of an output variable. This is not applicable if singletons are used for output membership functions. In other cases, the RANGE is used for limiting each membership function to the range of each output variable and should be defined to avoid unpredictable output values.

If there is no range defined the default range shall be the range of the data type of the variable specified in Part 3.

5.2.4 Rule block

The inference of the fuzzy algorithm shall be defined in one or more rule blocks. For proper handling and to cater for the possibility of splitting the rule base into different modules, the use of several rule blocks is possible. Each rule block has a unique name.

Rules shall be defined between the keywords RULEBLOCK and END_RULEBLOCK.

```
RULEBLOCK ruleblock_name
        operator_definition ;
        [activation_method ;]
         accumulation_method ;
         rules ;
END_RULEBLOCK
```

The Fuzzy operators are used inside the rule block.

```
operator_definition ::= operator : algorithm
```

To fulfill de Morgan's Law, the algorithms for operators AND and OR shall be used pair-wise e.g. MAX shall be used for OR if MIN is used for AND.

Table 5.2.4-1: Paired algorithms

operator **OR**		operator **AND**	
keyword for **Algorithm**	**Algorithm**	keyword for **Algorithm**	**Algorithm**
MAX	Max ($\mu_1(x)$, $\mu_2(x)$)	MIN	Min($\mu_1(x)$, $\mu_2(x)$)
ASUM	$\mu_1(x)+\mu_2(x) - \mu_1(x)\ \mu_2(x)$	PROD	$\mu_1(x)\ \mu_2(x)$
BSUM	Min(1, $\mu_1(x) + \mu_2(x)$)	BDIF	Max (0, $\mu_1(x) + \mu_2(x) -1$)

An example of rule blocks:

```
RULEBLOCK first
        AND : MIN;
        ..
END_RULEBLOCK
RULEBLOCK second
        AND : PROD;
        ..
END_RULEBLOCK
```

The following language element defines the method of the activation:

```
ACT : activation_method;
```

The following activation methods are available:

Table 5.2.4-2: Activation methods

Name	Keyword	Algorithm
Product	**PROD**	$\mu_1(x)\ \mu_2(x)$
Minimum	**MIN**	$\mathbf{Min(\mu_1(x), \mu_2(x))}$

NOTE -The activation method is not relevant for singelton.

The following language element defines the method of the accumulation.

ACCU : *accumulation_method*;

The following *accumulation_methods* are possible:

Table 5.2.4-3: Accumulation methods

Name	**Keyword**	**Formula**
Maximum	MAX	$\mathrm{Max}(\mu_1(x), \mu_2(x))$
Bounded Sum	BSUM	$\mathrm{Min}(1, \mu_1(x) + \mu_2(x))$
Normalised Sum	NSUM	$\frac{\mu_1(x) + \mu_2(x)}{\mathrm{Max}(1, \mathrm{MAX}(\mu_1(x') + \mu_2(x')))}$

The inputs of a rule block are *linguistic variables* with a set of *linguistic terms*. Each term has a degree of membership assigned to it.

Inside the rule block the rules are defined. Each begins with the keyword RULE followed by a name for the rule and shall be concluded by a semicolon. Each rule has a unique number inside the rule block.

RULE *numbers* : IF condition THEN conclusion [WITH weighting factor];

The rule itself shall begin with the keyword IF followed by the *condition*. After the *condition* the *conclusion* follows beginning with the keyword THEN.

It is possible to combine several *subconditions* and input variables in one rule. The purpose of variables is to permit fuzzy degrees of membership to be imported into the Fuzzy Function Block. All of them shall be defined between the kewords IF and THEN and combined by the operators with the keywords AND, OR or NOT .

The priority of the operator is handled according to boolean algebra given in Table 5.2.4.1.

Table 5.2.4-4: Priority of operators

Priority	**operator**
1	( ) parenthesis
2	NOT
3	AND
4	OR

Simplified example for a rule:

RULE 1 : IF *subcondition1* AND *variable1* OR *variable2* THEN *conclusion*;

In the Basic Level of conformance the OR operation can be implemented by defining two rules:

RULE 3 : IF *subcondition 1* OR *subcondition 2* THEN *conclusion*;
replaced by:
RULE 3a : IF *condition 1* THEN conclusion;
RULE 3b : IF *condition 2* THEN conclusion;

The subcondition begins with the name of a *linguistic variable* followed by the keyword IS with an optional NOT and one *linguistic term* of the *linguistic variable* used in the *condition.*

Subcondition := *linguistic_variable* IS [NOT] *linguistic_term*

Note: The *linguistic terms* which are used in the *condition* shall match the *linguistic variable* in the same *condition*. The term used has to be previously defined with the keyword TERM.

Example of subconditions:

```
temp IS hot
temp IS NOT hot
```

It is also possible to use the keyword NOT in front of the subcondition. In this case parentheses shall be used.

```
IF NOT (temp IS hot) THEN ...
```

The *conclusion* can be split into several subconclusions and output variables.

The subconclusion begins with the name of a *linguistic variable* followed by the keyword IS and one *linguistic term* of the given *linguistic variable*.

Subconclusion := *linguisti_ variable* IS *linguisti_ term*

Example with several subconclusions in one or more lines:

```
    IF temp IS cold AND pressure IS low THEN var1, valve1 IS inlet , valve2 IS closed;
or In several lines:
    IF temp IS cold AND pressure IS low
    THEN  var1,
          valve1 IS inlet,
          valve2 IS closed;
```

Optionally it is possible to give each subconclusion a *weighting factor* which is a number of data type REAL with a value between 0.0 and 1.0. This shall be done by the keyword WITH followed by the *weighting factor*.

The *weighting factor* shall reduce the membership degree (membership function) of the subconclusion by multiplication of the result in the subconclusion with the *weighting factor*.

In order to manipulate the Fuzzy Control application parameters externally the *weighting factor* can be a variable. In this case the variable has to be declared in the VAR_INPUT section. This enables the possibility to change the *weighting factor* during runtime in order to adapt the Fuzzy Conrtol program to process needs.

If there is no WITH statement assigned to the subconclusion a default *weighting factor* of 1.0 shall be assumed.

IF *condition* THEN *subconclusion* [WITH *weighting_factor*] *subconclusion* ;

An example of a constant *weighting_factor*:

```
IF temp IS cold AND pressure IS low THEN    valve1 IS inlet WITH 0.5,
                                            valve2 IS closed;
```

An example of a variable *weighting factor*:

```
VAR_INPUT
        w_myrule1 : REAL := 0.8;
END_VAR
RULEBLOCK temp_rule
        RULE 1:     IF temp      IS cold AND pressure IS low
                    THEN valve   IS inlet WITH w_myrule1;
        ..
END_RULEBLOCK
```

5.2.5 Optional parameters

For implementation on different target systems it may be necessary to give additional information to the system in order to allow the best possible conversion of Fuzzy Control applications.

Such additional information may be required in a language element enclosed by OPTIONS and END_OPTIONS.

```
OPTIONS
        application_specific_parameters
END_OPTIONS
```

These language elements shall be used for features in the conformance class of the open level according clause 6.

5.3 FCL example

An example in Fuzzy Control Language is given in Figure 5.3-1.

```
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
```

Figure 5.3-1: Example for fuzzy function block

5.4 Production Rules and Keywords of the Fuzzy Control Language (FCL)

Annex A of Part 3 defines the specification method for textual languages for Programmable Controllers. This specification method is used here for FCL.

Annex B of Part 3 defines the formal specification of language elements for the textual programming languages of Part 3. For FCL a subset of the following language elements of Part 3 shall be used:

B.1.1 Letters, digits, identifers
B.1.2 Constants
B.1.3 Data types
B.1.4 Variables

5.4.1 Production Rules

Additionally to the above listed language elements of Part 3 following language elements shall be used:

function_block_declaration ::= ‘FUNCTION_BLOCK‘ function_block_name
{fb_io_var_declarations}
{other_var_declarations}
function_block_body
‘END_FUNCTION_BLOCK’

fb_io_var_declarations ::= input_declarations | output_declarations

other_var_declarations ::= var_declarations

function_block_body ::= {fuzzify_block}
{defuzzify_block}
{rule_block}
{option_block}

fuzzify_block ::= ‘FUZZIFY’ variable_name
{linguistic_term}
‘END_FUZZIFY

defuzzify_block ::= ‘DEFUZZIFY’ f_variable_name
{linguistic_term}
defuzzification_method
default_value
[range]
‘END_FUZZIFY’

rule_block ::= ‘RULEBLOCK’ rule_block_name
operator_definition
[activation_method]
accumulation_method
{rule}
‘END_RULEBLOCK’

option_block ::= ‘OPTION’
any manufacturere specific parameter
‘END_OPTION’

linguistic_term ::= ‘TERM’ term_name ‘:=‘ membership_function ‘;’

membership_function ::= singleton | points

singleton ::= numeric_literal | variable_name

points ::= {‘(‘ numeric_literal | variable_name ‘,’
numeric_literal ‘)’}

defuzzification_method ::= ‘METHOD’ ‘:’ ‘COG’ | ‘COGS’ | ‘COA’ | ‘LM’ | ‘RM’ ‘;’

default_value ::= ‘DEFAULT’ ‘:=‘ numeric_literal | ‘NC’ ‘;’

range ::= ‘RANGE’ ‘:=‘ ‘(‘numeric_literal ‘..’ numeric_literal‘)’ ‘;’

operator_definition ::= (‘OR’ ‘:’ ‘MAX’ | ‘ASUM’ | ‘BSUM’) |
(‘AND’ ‘:’ ‘MIN’ | ‘PROD’ | ‘BDIF’) ‘;’

activation_method ::= ‘ACT’ ‘:’ ‘PROD’ | ‘MIN’ ‘;’

accumulation_method ::= ‘ACCU’ ‘:’ ‘MAX’ | ‘BSUM’ | ‘NSUM’ ‘;’

rule ::= ‘RULE’ integer_literal ‘:’
‘IF’ condition ‘THEN’ conclusion [WITH weighting_factor] ‘;’

condition ::= (subcondition | variable_name) {AND’ | ‘OR’
(subcondition | variable_name)}

subcondition ::= (‘NOT’ ‘(‘ variable_name ‘IS’ [‘NOT’]) term_name ‘)’) |
(variable_name ‘IS’ [‘NOT’] term_name)

conclusion ::= { (variable_name | (variable_name ‘IS’ term_name)) ‘,’}
(variable_name | variable_name ‘IS’ term_name)

weighting_factor ::= variable | numeric_literal

function_block_name ::= identifier

rule_block_name ::= identifier

term_name ::= identifier

f_variable_name ::= identifier

variable_name ::= identifier

numeric_literal ::= integer_literal | real_literal

input_declarations ::= see IEC 1131-3 Annex B

output_declarations ::= see IEC 1131-3 Annex B

var_declarations ::= see IEC 1131-3 Annex B

identifier ::= see IEC 1131-3 Annex B

5.4.2 Keywords

Table 5.4.2-1: Reserved keywords

Keyword	Meaning	Clause
( )	Parentheses in condition, term, range	5.2.4
ACCU	Accumulation method	5.2.4
ACT	Actuation method	5.2.4
AND	AND operator	5.2.4
ASUM	OR operator, Algebraic sum	5.2.4
BDIF	AND operator, Bounded difference	5.2.4
BSUM	Accumulation method, Bounded sum	5.2.4
COA	Center of area defuzzification method	5.2.3
COG	Center of gravity defuzzification method	5.2.3
COGS	Center of gravity defuzzification of singletons	5.2.4
DEFAULT	Default output value in case no rule has fired	5.2.3
DEFUZZIFY	Defuzzification of output variable	5.2.3
END_DEFUZZIFY	End of defuzzification specifications	5.2.3
END_FUNCTION_BLOCK	End of function block specifications	5.2.1
END_FUZZIFY	End of fuzzification specifications	5.2.3
END_OPTIONS	End of options specifications	5.2.5
END_RULEBLOCK	End of rule block specifications	5.2.4
END_VAR	End of input/output variable definitions	5.2.1
FUNCTION_BLOCK	End of function block specifications	5.2.1
FUZZIFY	Fuzzification of input variable	5.2.2
IF	Begin of rule which is followed by the condition	5.2.4
IS	Follows linguistic variable in condition and conclusion	5.2.4
LM	Left Most Maximun defuzzification method	5.2.3
MAX	Maximum accumulation method	5.2.3
METHOD	Method of defuzzification	5.2.3
MIN	Minimum as AND operator	5.2.4
NC	No Change of output variable in case no rule has fired	5.2.3
NOT	NOT operator	5.2.4
NSUM	Normalised sum accumulation method	5.2.4

OPTIONS	Definition of optional parameters	5.2.5
OR	OR operator	5.2.4
PROD	Product as AND operator	5.2.4
RANGE	Range of variable for scaling of membership function	5.2.3
RM	Right Most Maximum defuzzification method	5.2.3
RULE	Begin of specification of fuzzy rule	5.2.4
RULEBLOCK	Begin of specification of rule block	5.2.4
TERM	Definition of a linguistic term (membership function) for a linguistic variable	5.2.2
THEN	Separates condition from conclusion	5.2.4
VAR	Definition of local variable (s)	5.2.1
VAR_INPUT	Definition of input variable(s)	5.2.1
VAR_OUTPUT	Definition of output variable(s)	5.2.1
WITH	Definition of weighting factor	5.2.4

6 Compliance

6.1 Conformance Classes of Fuzzy Control Language FCL

The levels of conformance for Control System using the Fuzzy Control Language (FCL) are shown in Figure 6.1-1. The hierarchy consists of the following three levels:

Basic Level including the definitions of the function block and the data types of Part 3.

Extension level with optional features.

Open Level encompassing additional features.

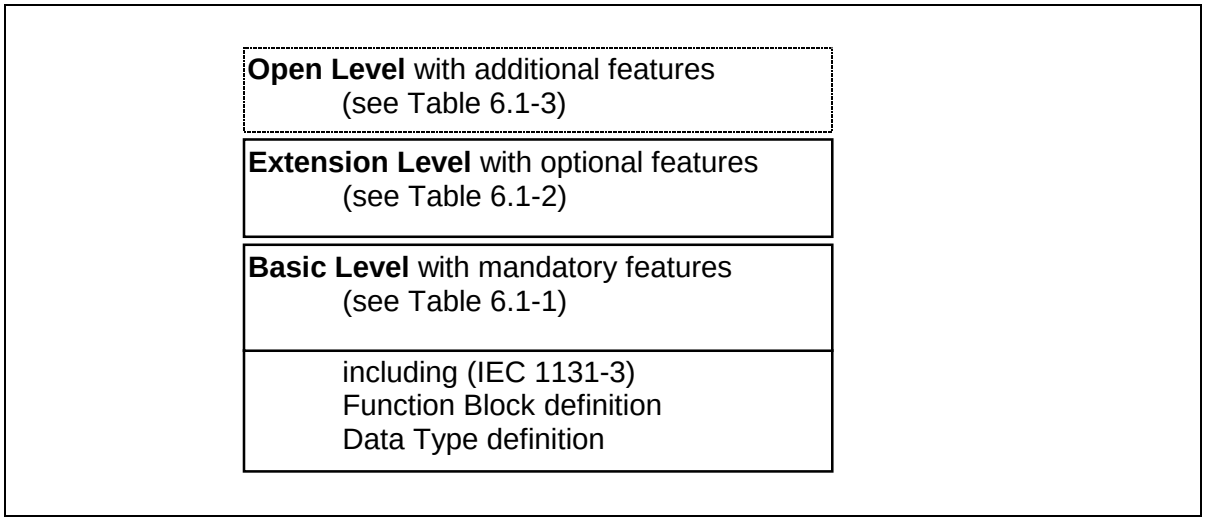


Figure 6.1-1: Levels of conformance

A Control System using the Fuzzy Control Language FCL claiming to conform with this part of the Standard shall achieve the following rules:

1. It shall be able to use the Function Block features according to Part 3 of this Standard in order to realise the Fuzzy Control functionality. Therefore the definition of the Function Blocks and the Data Types required for the input and output parameters of the function block shall be in accordance with Part 3.

2. All features of Fuzzy Control functionality defined in Table 6.1-1 shall be implemented according to the definitions of this Part. This table defines the set of Basic Level elements which all standard systems shall have in common.

3. A subset of the Extension Level elements defined in table 6.1-2 are additional elements which can be implemented optionally. The implementation shall be exactly according to the definitions of this Part. These features shall be marked as Standard Extensions and a list of realised features in form of Table 6.1-2 shall be part of the system documentation.

4. Further features exceeding the Basic Level and the Extended Level may be realised provided these features do not have the same or similar functionality or representation of the standard features thereby avoiding any possible confusion. These features shall be marked as Open Level features and a list in form of Table 6.1-3 shall be part of the system documentation.

5. The exchange of application programs among different Fuzzy Control systems shall be done in the textual form of the Fuzzy Control Language FCL according to the definitions in this part. This

format shall be made available by the standard comformant systems as input and output formats.

6. In order to achieve the most comfortable and suitable user interface and to not hinder future progress, the external representations for the design, input, testing etc. of Fuzzy Control application programs can be realised by any graphical or textual means.

The elements in the Table 6.1-1 are the basic set of features, which shall be realised in all standard Fuzzy Conrol systems.

Table 6.1-1: FCL Basic Level language elements (mandatory)

Language Element	**Keyword**	**Details**
function block declaration	VAR_INPUT, VAR_OUTPUT	contains input and output variables
membership function	input variable: TERM	maximum of three points (degree of membership co-ordinate = 0 or 1)
	output variable: TERM	only singletons
conditional aggregation	operator: AND	algorithm: MIN
activation	-	Not relevant because singletons are used only
accumulation (result aggregation)	operator: ACCU	algorithm: MAX
defuzzification	METHOD	algorithm: COGS
default		Note - Provisions shall be made to handle this feature in the Basic Class
condition	IF ... IS ...	n subconditions
conclusion	THEN	only one subconclusion
weighting factor	WITH	value only

The elements in the Table 6.1-2 are the extended set of features, which can be optionally realised in a standard Fuzzy Control system (e.g. for the AND operator the algorithm PROD or BDIF or both might be chosen).

Table 6.1-2: FCL Extension Level language elements (optional)

Language Element	Keyword	Details
function block declaration	VAR	contains local variables
membership function:	input variable:TERM	maximum of four points (degree of membership co-ordinate = 0 or 1)
	output variable:TERM	maximum of four points (degree of membership co-ordinate = 0 or 1)
conditional aggregation	operator: AND	algorithm: PROD , BDIF
	operator: OR	algorithm: ASUM , BSUM
	operator: NOT	1 - {argument}
	parentheses	( )
activation	operator: ACT	algorithm: MIN, PROD
accumulation	operator: ACCU	algorithm: BSUM , NSUM
defuzzification method	operator: METHOD	algorithm: COG , COA , LM , RM
default value	DEFAULT	NC, value
condition	IF	n subconditions, n input variables
conclusion	THEN	n subconclusions, n output variables
weighting factor	WITH	value assigned to variable in the declaration part VAR_INPUT.....END_VAR

The Table 6.1-3 defines an example of a list of language elements in the Open Level. This list shall be a part of the system documentation.

Table 6.1-3: Examples of List with Open Level language elements

free input/output membership functions (e.g. Gaussian, exponential)
more than four membership function points
degree of membership co-ordinate values from 0 to 1
please dream on .....

6.2 Data check list

This data check list shall be delivered within the technical documentation. In this list a manufacturer of programmable controllers, Fuzzy programming tools and application software shall describe specific performance features of their Fuzzy Control system. In order to facilitate the transfer of Fuzzy Control applications among different manufacturer's systems the following Data check list is the means of verifying a possible program transfer.

Table 6.2-1: Data Check List

Technical data	Manufacturer statement (*examples*)
data types of function block inputs and outputs	*REAL, INT*
line comments in the FCL program	*YES, NO*
length of identifiers (e.g. name of variables, ruleblocks, terms)	6, 8
max. number of input variables for fuzzification	6, 8
max. number of membership function terms per input variable	5, 7
max. total number of membership function terms for all input variables	30, 56
max. number of points for the membership function associated with each input variable term	3, 4, 10
max total number of points for membership functions associated with all input variable terms	90, 224
max. number of output variables for defuzzification	6, 8
max. number of membership function terms per output variable	5, 7
max. total number of membership function terms for all output variables	30, 56
max. number of points for the membership function associated with each output variable term	1, 4, 10
max total number of points for membership functions associated with the all output variable terms	90, 224
max. number of rule blocks	1, 10
max. number of rules per block	10
max. number of subconditions per rule	4, 10
max. number of all rules	15
max. number of subconclusions per rule	4
Nesting depth of ( )	1, 3

Annex A Theory (informative)

The following clause "Theory" is understood as an explanation of the Definitions given in clause 3.

A.1 Fuzzy Logic

In *Fuzzy Logic,* linguistic values and expressions are used to describe physical variables, instead of the names, numbers (usually real numbers) used in conventional open and closed-loop control systems. The terms "low" or "wide-open" are designated as *linguistic terms* of the physical values "temperature" or "heating valve opening". If an input variable is described by *linguistic terms*, it is referred to as a *linguistic value*.

Each *linguistic term* is described by a *Fuzzy Set* M. It is thus unequivocally defined mathematically by the two statements basic set G and *membership function* μ. The *membership function* states the membership of every element of the universe of discourse G (e.g. numerical values of a time scale [age in years]) in the set M (e.g. "young") in the form of a numerical value between zero and one. If the *membership function* for a specific value is one, then the linguistic statement corresponding to the *linguistic term* applies in all respects (e.g. "young" for an age of 20 years). If, in contrast, it is zero, then there is absolutely no agreement (e.g. "very old" for an age of 20 years).

The following notations are used to describe *Fuzzy Sets*:

for finite sets: as unordered, paired sets in incremental form:

$$M = \{(x_1,\mu_M(x_1)), (x_2,\mu_M(x_2)),..,(x_n,\mu_M(x_n))\}, \quad x_i \in G,\ i=1,2,..n \qquad (A.1)$$

the $\mu_M(x_i)$ are listed as numerical values.

for infinite sets:

$$M = \{x, \mu_M(x)\}, \qquad x \in G \qquad (A.2)$$

In order to more clearly illustrate the differences between *crisp* and *fuzzy* terms, the *linguistic terms* of the *linguistic variable* "Age" are represented in Figure A.1-1. While "full legal age" is unequivocally stipulated by law, and thus displays a discrete transition in relation to the *membership function*, a crisp age limit cannot be given for "adult".

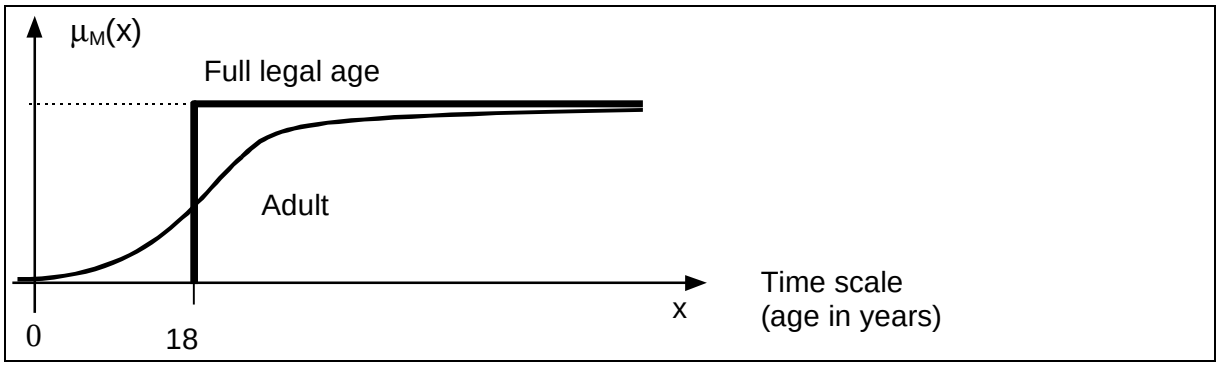


Figure A.1-1: Membership functions of the terms "full legal age" and "adult"

As an example, Figure A.1.-2 shows the description of the *linguistic variable* "Age" by *linguistic terms* and their hierarchy on the time scale "age in years" by means of *membership values*.

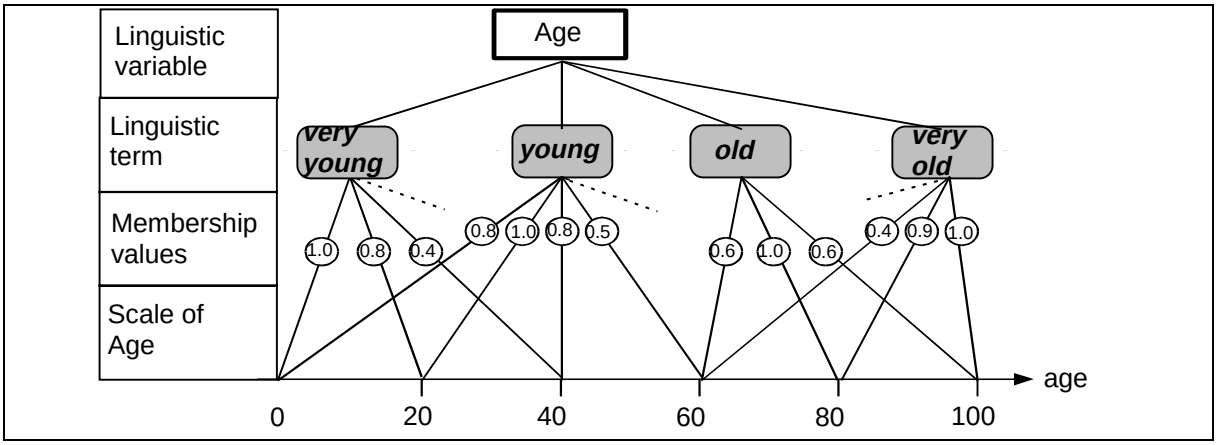


Figure A.1-2: Description of the linguistic variable "Age" by linguistic terms and their hierarchy on the time scale (age in years)

Typical forms of the *membership functions* are represented in Figure A.1-3. The following count as special forms:

- the definition via the rectangle (e.g. interval) in order to describe *value*s, as well as

- the *"singleton",* for the alternative representation of output variables.

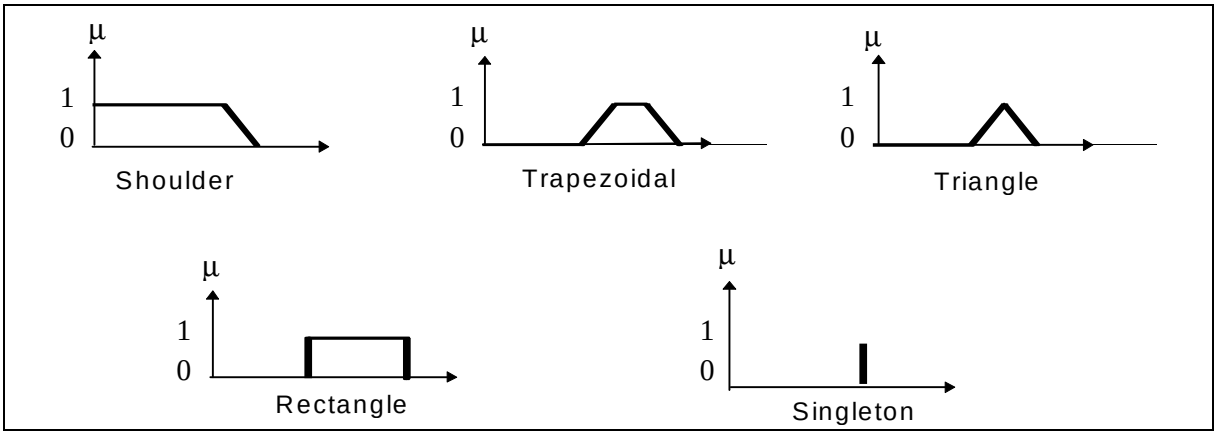


Figure A.1-3: Commonly used shapes of membership functions

An expression in which *linguistic variables* are related to *linguistic terms* represents a *linguistic statement* in *Fuzzy Logic*. Expressions such as "Temperature is high" or "Temperature is low", with the simple basic structure of

Linguistic variable - Symbol of comparison - Linguistic term (Temperature is low)	(A.3)

are referred to here as *linguistic statement.*

In contrast to *classical logic*, in which statements only assume one of the Boolean states "true" or "false", *linguistic statements* in *Fuzzy logic* possess a *degree of membership*.

Empirical knowledge can be defined in *rules*. Rule R_k has the following form:

R_k: IF *condition* P_k THEN *conclusion* C_k	(A.4)

In this context, the *condition* of each *rule* comprises a linguistic statement or a combination of statements via the input variables, while the *conclusion* determines the output variable in the sense of an instruction to act.

P_k = A AND B OR (NOT C) (A.5)

If *condition* and/or *conclusion* are determined by *linguistic statements*, the *rule* is then also referred to as a *linguistic rule*. A *rule base*, in turn, consists of several *rules* together. In general, several *rules* apply ("fire") at the same time in contrast to classical rule-based systems. Therefore, the results of the rules must be combined with one another via corresponding mathematical *operators*.

Relationships between *Fuzzy Sets* and *operations* with *Fuzzy Sets* are defined by the *membership functions*:

The following relationships apply between two *Fuzzy Sets*, A and B, whose elements x originate from a basic set G

equality A = B,

is true if $\mu_A(x) = \mu_B(x)$ for all $x \in G$ (A.6)

complete inclusion $A \subseteq B$,

is true if $\mu_A(x) \leq \mu_B(x)$, for all $x \in G$ (A.7)

partial inclusion $A \subset B$,

is true if $\mu_A(x) \leq \mu_B(x)$, for all $x \in G$

and $\mu_A(x) < \mu_B(x)$ for at least one $x \in G$. (A.8)

The following *operations* can be stipulated between two *Fuzzy Sets*, A and B, whose elements x originate from a basic set G:

the *intersection*

$A \cap B$ is defined by $\mu_{A \cap B}(x)=I(\mu_A(x), \mu_B(x))$, (A.9)

where I is called the intersection operator

the *union*

$A \cup B$ is defined by $\mu_{A \cup B}(x)=U(\mu_A(x), \mu_B(x))$, (A.10)

where U is called the union operator

the *complement*

is defined by $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$ (A.11)

As with classical (*crisp*) sets, the following interpretations apply to Fuzzy Sets:

the *intersection* is related to the logical operator AND,
the *union* is related to the logical operator OR,
the *complement* is related to the logical operator NOT

Elementary algorithms for mathematical implementation of intersection, union and complement are,

for the *intersection,* the minimum

$\mu_{A \cap B}(x) = \text{Min}\{\mu_A(x), \mu_B(x)\}$, $x \in G$ (A.12)

for the *union*, the maximum

$\mu_{A \cup B}(x) = \text{Max}\{\mu_A(x), \mu_B(x)\}$, $x \in G$ (A.13)

for the *complement*, subtraction from one

$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$ $x \in G$ (A.11)

In addition to the above-mentioned elementary fuzzy logic operators, there are a lot of choices for other fuzzy logic operators mainly for non control applications. It is worth noting here, that AND and OR operators cannot be chosen in an arbitrary way.

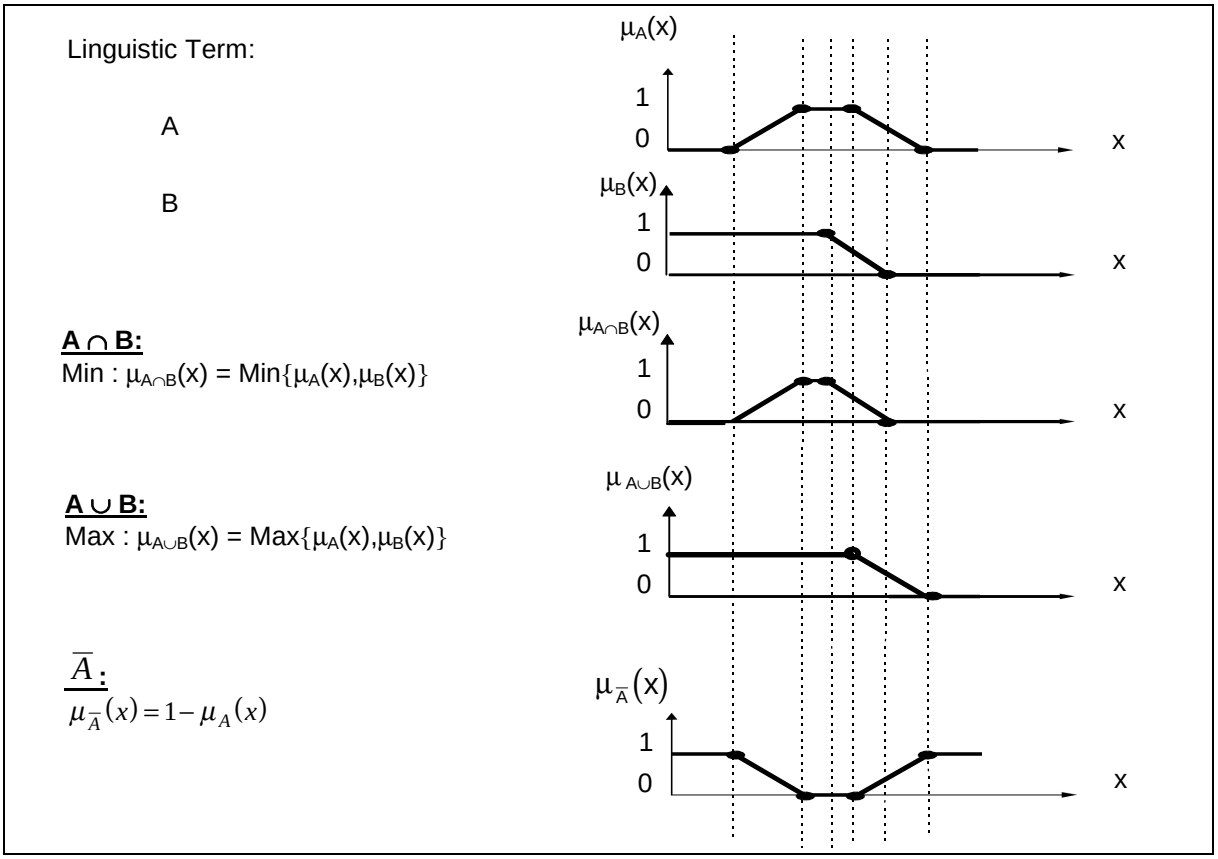


Figure A.1-4: Algorithms for implementing operations between two membership functions

A.2 Fuzzy Control

Fuzzy Control means the open and closed-loop control of technical processes, including the processing of measured values, which is based on the use of fuzzy rules and their processing with the help of *Fuzzy Logic*.

The input information comprises real variables in the form of measurable process variables, derived variables, as well as set points. The output variables are real variables in the form of correcting variables. Transformations must be performed between the input and output variables of the process and the *Fuzzy* world (*fuzzification, defuzzification*). The core component of *Fuzzy Control* consists of the linguistic rules of the *rule base* and the *inference*.

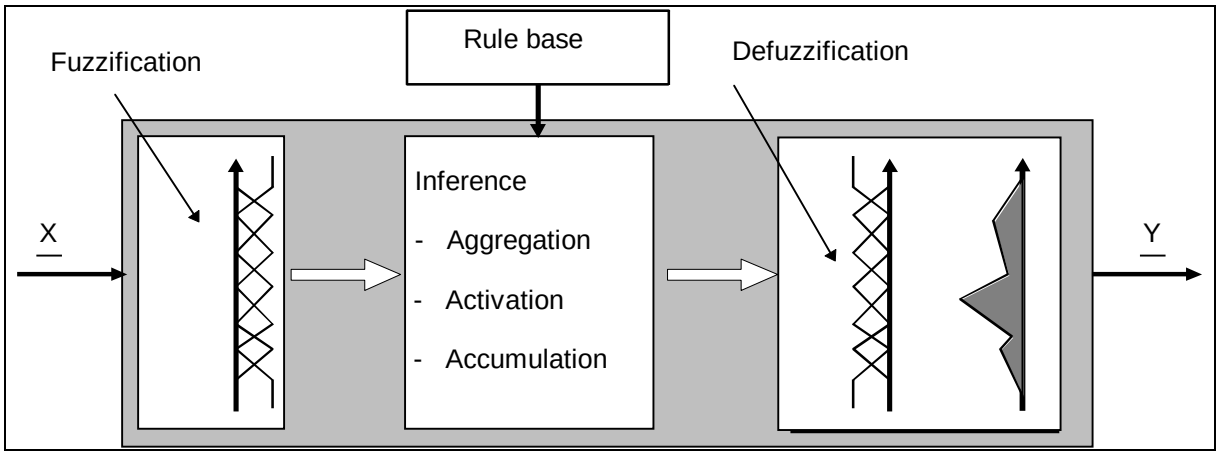


Figure A.2-1: Structure and functional elements of Fuzzy Control

The functional elements of *Fuzzy Control*, mentioned above and represented in Figure A.2-1, are explained below.

A.2.1 Fuzzification

The determination of the matching of input variables with the *linguistic terms* is referred to as *fuzzification*. To this end, the actual degree of membership for input variables is determined for each *linguistic term* of the corresponding *linguistic variable*.

Figure A.2.1-1 shows an example of *fuzzification.*

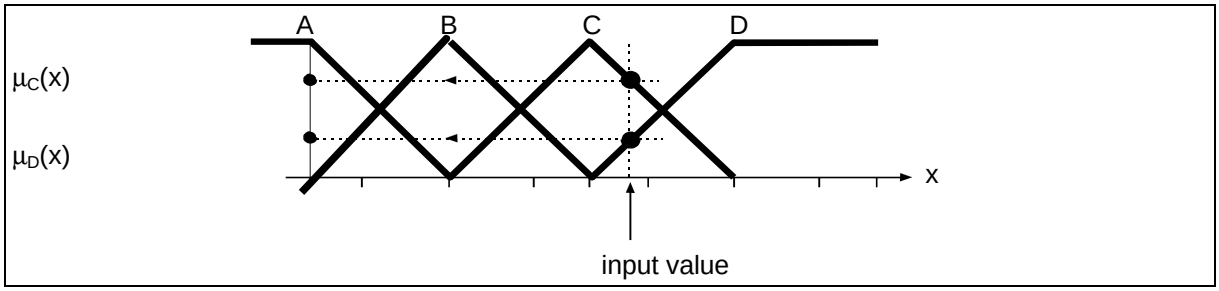


Figure A.2.1-1: The principle of fuzzification (by way of an example)

A.2.2 Rule base

The *rule base* contains empirical knowledge concerning the operation of a particular process under consideration. *Linguistic rules* are used to represent the knowledge. It can be easily shown that a fuzzy rule R_j based on an OR combination of m statements can be represented by m rules, whose statements are only combined by AND. Figure A.2.2-1 and A.2.2-2 are examples of different rule representations.

Rule Base

- Definition of the rule base in textual form

R1:	IF	Condition P1	THEN	Conclusion C1
R2:	IF	Condition P2	THEN	Conclusion C2
R3:	IF	Condition P3	THEN	Conclusion C3
….				
R_n:	IF	Condition P_n	THEN	Conclusion C_n

where condition can consist of a combination of linguistic statements, e.g. for rule k

$P_k = P_{k1}$ AND P_{k2} OR (NOT P_{k3})

and for conclusion there may be more than one, e.g.

$C_k = C_{k1}$, C_{k2}

Figure A.2.2-1: Representation of the knowledge base in the form of linguistic rules and in matrix form

If two input variables and one output variable are available, and if these two input variables are combined only by AND, the *rule base* can be given in the form of a matrix, where the values of the input variables are assigned to the columns and/or lines, and the fields of the matrix contain the values of the output variable.

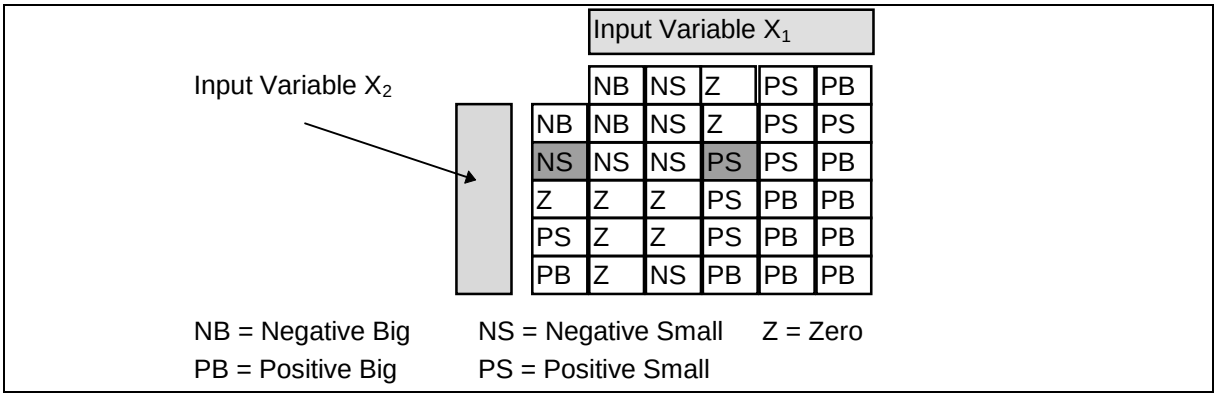


Figure A.2.2-2 Matrix representation of two variables.

A.2.3 Inference

More general and mathematically proven definitions of fuzzy rule bases are based on generalised modus ponens and fuzzy implication principles. The principles and definitions used here are valid for a simplified case of fuzzy rule base, basically inspired from the widely used Mamdani inferencing scheme. More complicated inferencing schemes are not in the realm of this standard.

The Inference consists of the three subfunctions *aggregation*, *activation* and *accumulation* shown in Figure A.2.3-1.

Execution of the rule base broken down into the following steps:

- Aggregation
  Determing the degree of accomplishement of the condition
  from the degree of membership of the subconditions

  P_{k1} AND P_{k2} OR (NOT P_{k3}) = P_k

  for all n conditions

- Activation
  Activation of the IF-THEN conclusion

  If condition P_k THEN conclusion C_k

  for all n rules

  - Consideration of the weighting factor of each rule

- Accumulation
  Combination of the weighted results
  of the rules into an overall result

Figure A.2.3-1: Elements of inference

- *Aggregation:*

If the condition consists of a single subcondition, then the validity of the *condition* is identical to that of the condition, i.e. the degree of accomplishment of the *condition* corresponds to the condition. However, if the *condition* consists of a combination of several subconditions, the degree of accomplishment must be determined by *aggregation* of the individual values. If an AND combination of subconditions exists, the degree of accomplishment is calculated by means of the AND *fuzzy logic operator.*

- *Activation*:

In the *conclusion*, *subconclusions* relate to the output variables. The *degree of membership* of the *conclusion* is then determined on the basis of the *degree of accomplishment* of the *condition* determined in aggregation (*Conclusion* IF A THEN B). In general, the MIN or PROD is used for activation.

If the rule base contains rules with weighting factors w_k, where $w_k \in [0, 1]$, this can be implemented by means of multiplication.

$$a_k^* = w_k \times a_k \qquad \text{(A.25)}$$

- *Accumulation:*

The results of the *rules* are combined to obtain an overall result. The *maximum algorithm is* usually used for *accumulation*. Table A.2.3-1 shows the *operators* commonly used for the individual *inference* steps.

Depending on the combination of *operators* in the individual steps, different *inference* strategies are obtained. The best-known are the so-called *MaxMin Inference* and *MaxProd Inference*, which use the *maximum* for *accumulation* and the *minimum* or the *algebraic product* for *activation*. In the case of the *MaxMin Inference*, the *membership functions* of the *Fuzzy Sets* of the *conclusions* are limited to the *degree of accomplishment* of the *condition* and then, in turn, combined to create a *Fuzzy Set* by forming a maximum. In *MaxProd Inference*, in contrast, the *membership functions* of the *Fuzzy Sets* of the *conclusions* are weighted, i.e. multiplied, with the *degree of accomplishment* of the *condition* and then combined.

Table A.2.3-1: Inference steps and commonly used algorithms

Inference step	*Operators*	Algorithms
Aggregation		
for AND	*Minimum*	$a_k = \text{Min}\{a_{K1}(x), a_{K2}(x)\}$
for OR	*Maximum*	$a_k = \text{Max}\{a_{K1}(x), a_{K2}(x)\}$
activation		
conversion of the IF-THEN-conclusion		
	Minimum	$c_k' = \text{Min}\{a_K, \mu_\infty(u)\}$
weighting factor of each rule		
	Multiplication	$c_k = \text{Mult}\{\omega_K, c_K'\} = \omega_K \times c_K'$
Accumulation	*Maximum*	$\mu_{accu}(u) = \text{MAX}\{c_i(u)\}$

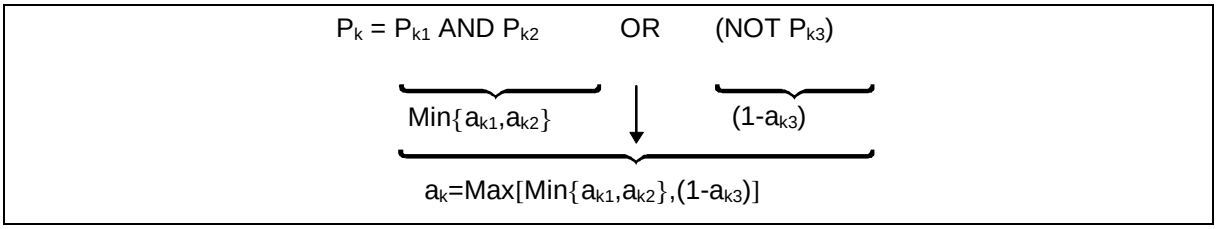


Figure A.2.3-2a: The principles of aggregation (by way of an example)

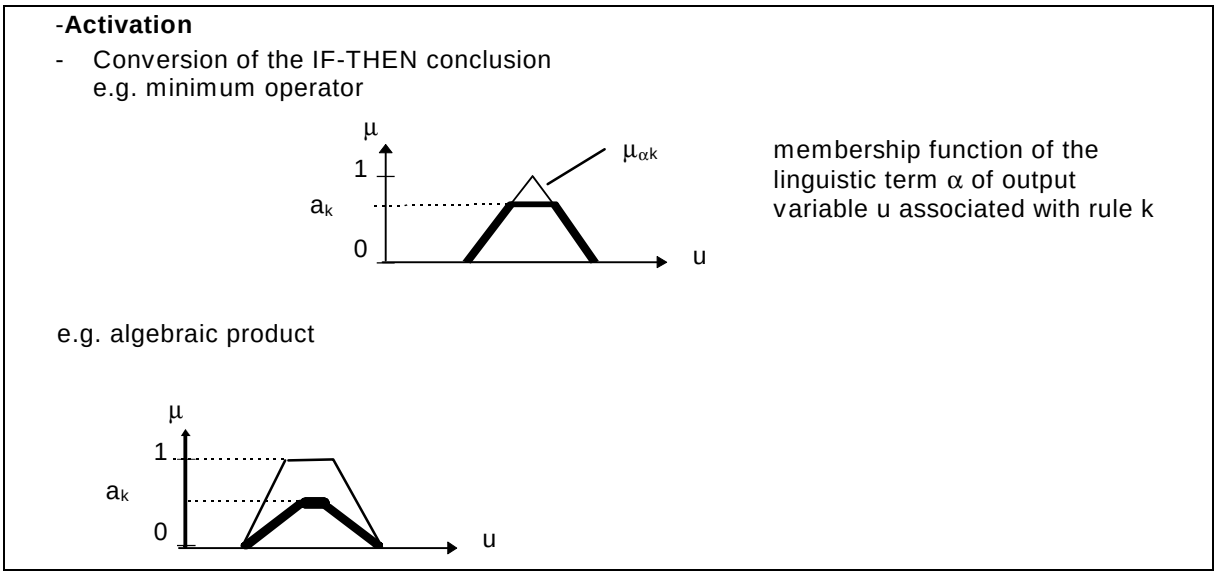


Figure A.2.3-2b: The principles of activation (by way of an example)

- **Accumulation**

e.g. Maximum operator

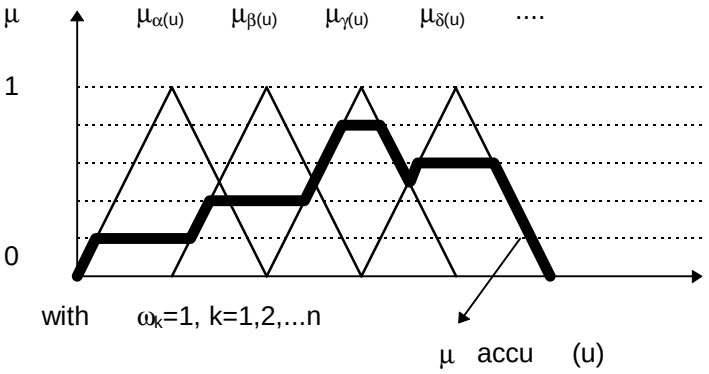


Figure A.2.3-2 c: The principles of accumulation (by way of an example)

A.2.4 Defuzzification

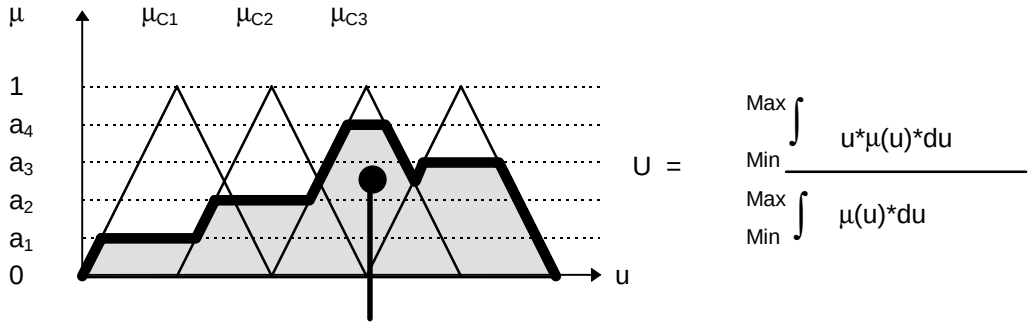
Inference supplies a *Fuzzy Set* or its *membership function* as a result. A control element cannot directly process this fuzzy information therefore the result of the *inference* process has to be converted into *crisp* numerical values. In this context, the *crisp* number to be determined (generally a real number) should provide a good representation of the information contained in the *Fuzzy Set*.

Defuzzification

Conversion of the fuzzy result of inference into a crisp output variable U

Center of Gravity (CoG) Method
The crisp output variable is determined as an abscissa value U of center of gravity under the membership function:

- example for center of gravity method



U : Center of Gravity (COG)

Figure A.2.4-1a: Methods of defuzzification

In the case of singletons mean of maximum and centre of gravity yield to the same result. [Is a formula for mean of maxima or a definition going to be supplied to account for the case when singletons are not used ? It should also be noted that Centroid of Area is equivalent to CoG calculation]

Further common methods are as follows:

Left Most Maximum LM
The value of the output variable is determined for which the *membership function* of the output reaches its leftmost maximum.

Right Most Maximum RM
The value of the output variable is determined for which the *membership function* of the output reaches its rightmost maximum.

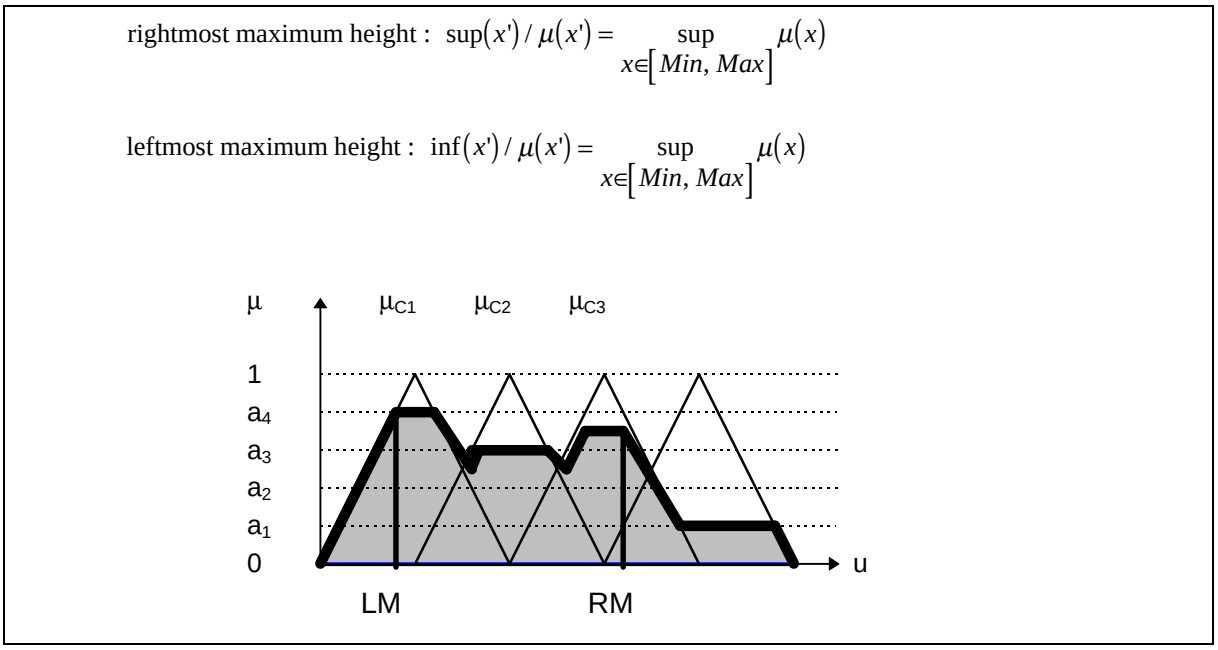


Figure A.2.4-1b: Difference between Left Most Maximum and Right Most Maximum

Centre of Area Method
Here, the output value is determined as the abscissa value of the centre of which divides the area under the *membership function* into 2 areas of equal size.

NOTE 1 - Centre of Gravity is equivalent to Centroid of Area
NOTE 2 - Centre of Area is equvalent to Bisector of Area
NOTE 3 - COA is not applicable if singletons are used.

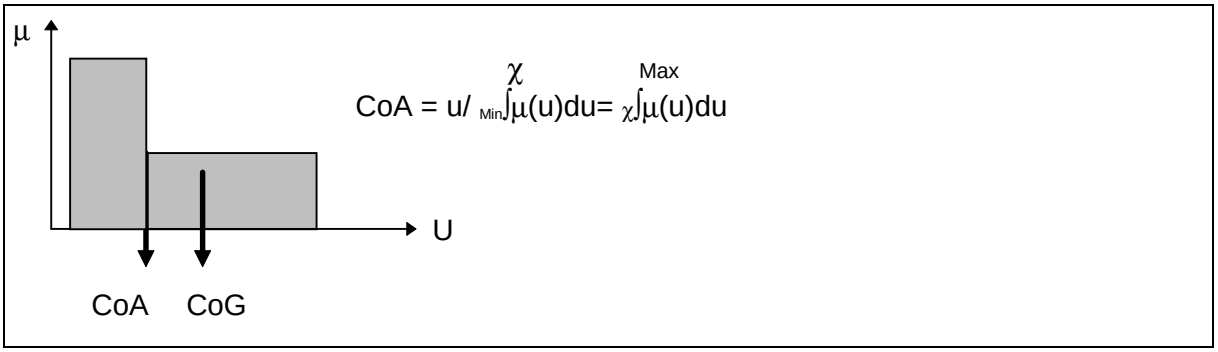


Figure A.2.4-1c: Difference between Center of Area and Center of Gravity

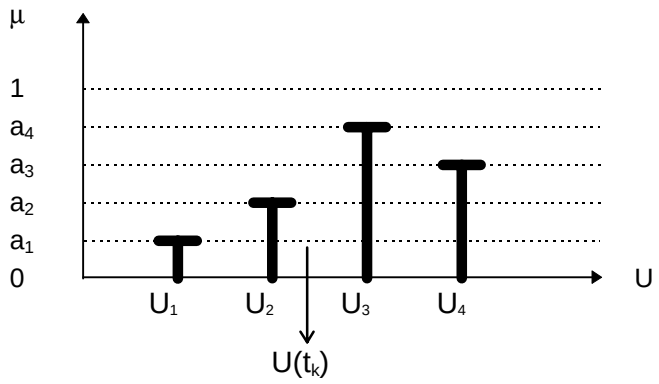
Defuzzification

If the membership functions of the output values are singletons, the calculation is given by:

- Center of Gravity Method for Singleton (COGS)

$$U(t_k) = \frac{\sum_{i=1}^{p} [U_i * a_i(t_k)]}{\sum_{i=1}^{p} [a_i(t_k)]}$$

where a_i are the singleton values of the individual rules



In the case of singletons the center of area is not applicable.
Example for the 4 singletons above :

$$U(t_k) = \frac{[U_1 * a_1(t_k) + U_2 * a_2(t_k) + U_3 * a_3(t_k) + U_4 * a_4(t_k)]}{[a_1(t_k) + a_2(t_k) + a_3(t_k) + a_4(t_k)]}$$

Figure A.2.4-1d: Methods of defuzzification

A. 3 Performance of Fuzzy Control

From a point of view of information technology, Fuzzy Control is a rule-based expert system. From the point of view of control systems technology, it is a generally non-linear characteristic field controller. Figure A.3-1 shows examples of Fuzzy Control characteristic curves. The current values of its output variables depend exclusively on the current values of the input variables, and not on previous values, except in the case where no rule is active and no default value has been defined. If the controller should be realised with dynamic behaviour, the dynamic functions must be provided external to the fuzzy function block.

These are usually differentiating and integrating elements of the first order. The output variables of these functions are additional input variables for Fuzzy Control. This also applies to the control deviation, which likewise has to be formed outside Fuzzy Control. Conversely, the output variables of Fuzzy Control can be passed to operators for processing of the correcting variables, e.g. an integration element for speed algorithms, or distributed among different control elements.

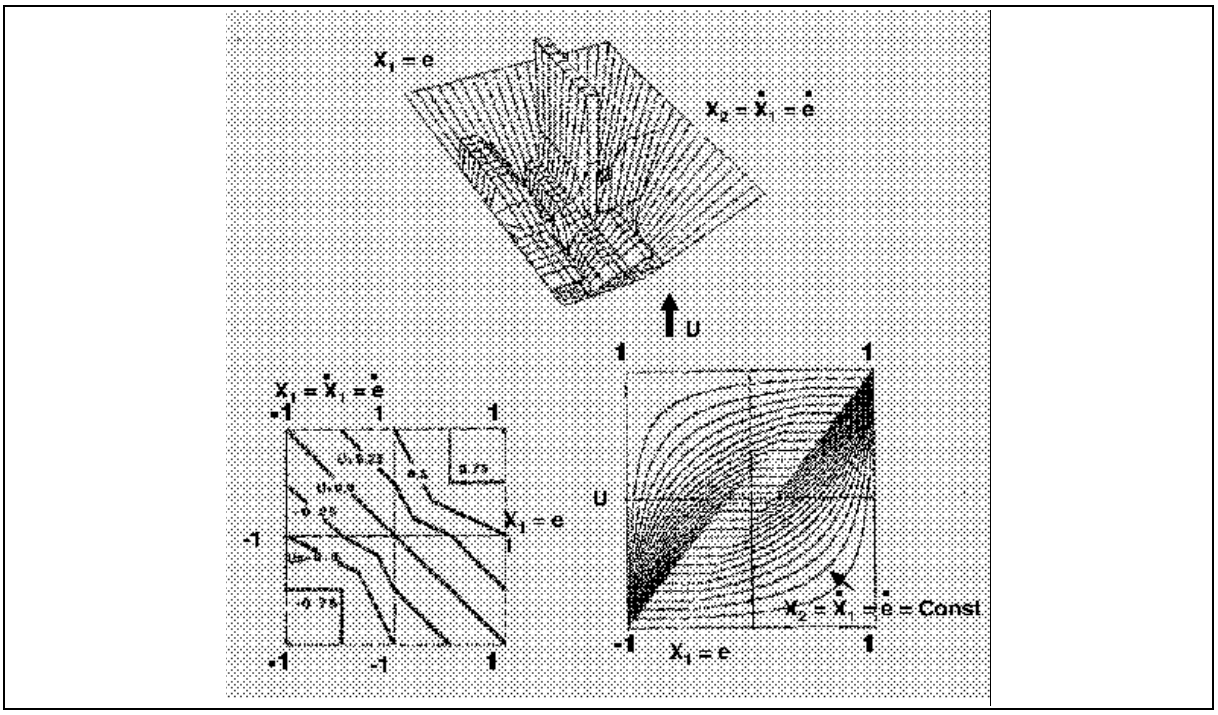


Figure A.3-1: Examples of Fuzzy Control characteristic curves

The fundamental structure of a fuzzy-based controller is shown in Figure A.3-2a, while Figure A.3-2b gives an example: the control difference is formed from the difference between the command variable and the controlled variable. This difference, as well as its time derivation and its integral over time, is transmitted to Fuzzy Control as three input variables which are independent from its perspective. The correcting variable is derived from the output variable by means of integration over time. If the rule base is designed in such a way that a velocity algorithm is described, the fuzzy-based controller thus displays dynamic behaviour similar to PID.

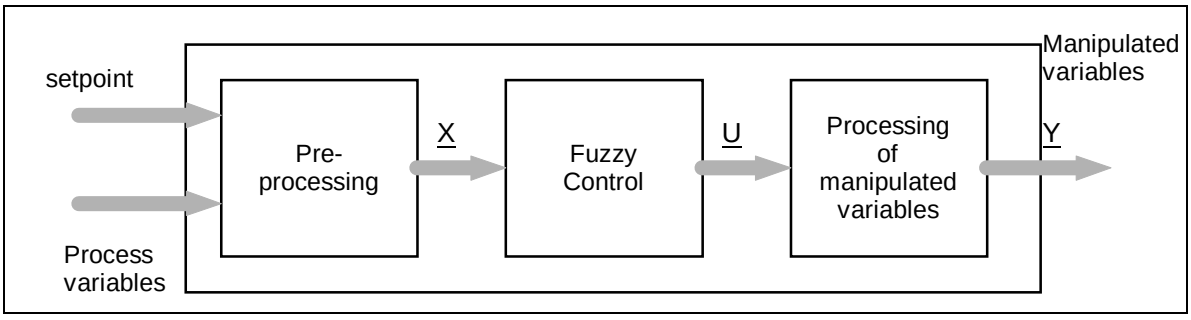


Figure A.3-2a: Fuzzy-based controller: Fundamental structure

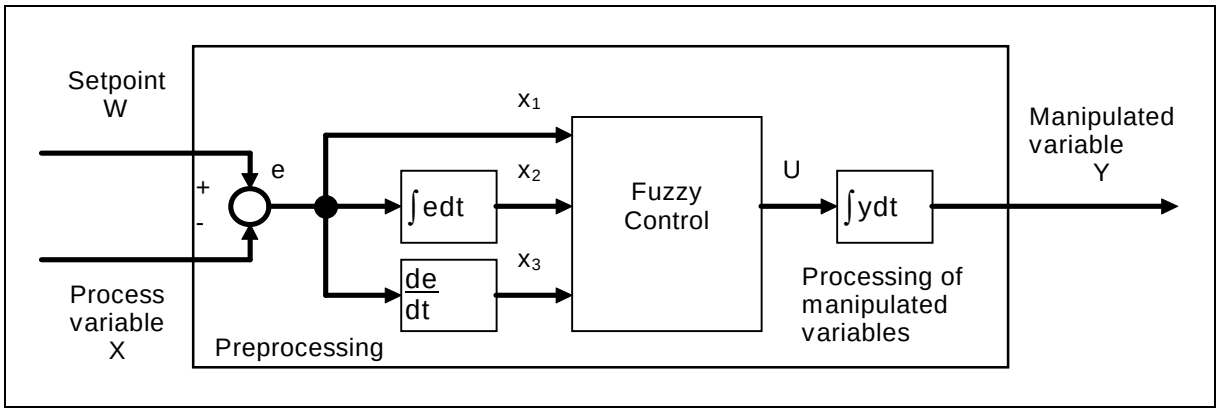


Figure A.3-2b: Example of a Fuzzy-based controller

Annex B Examples (informative)

One of the main application areas in the world of Programmable Controllers is in combination with conventional PID-controllers in order to improve the control quality of the PID controller. The following examples show the principal possibilities in general to give an idea where fuzzy control can be used.

B.1 Pre-control

The fuzzy controller supplements the conventional closed loop controller by a corrective signal for the manipulated value

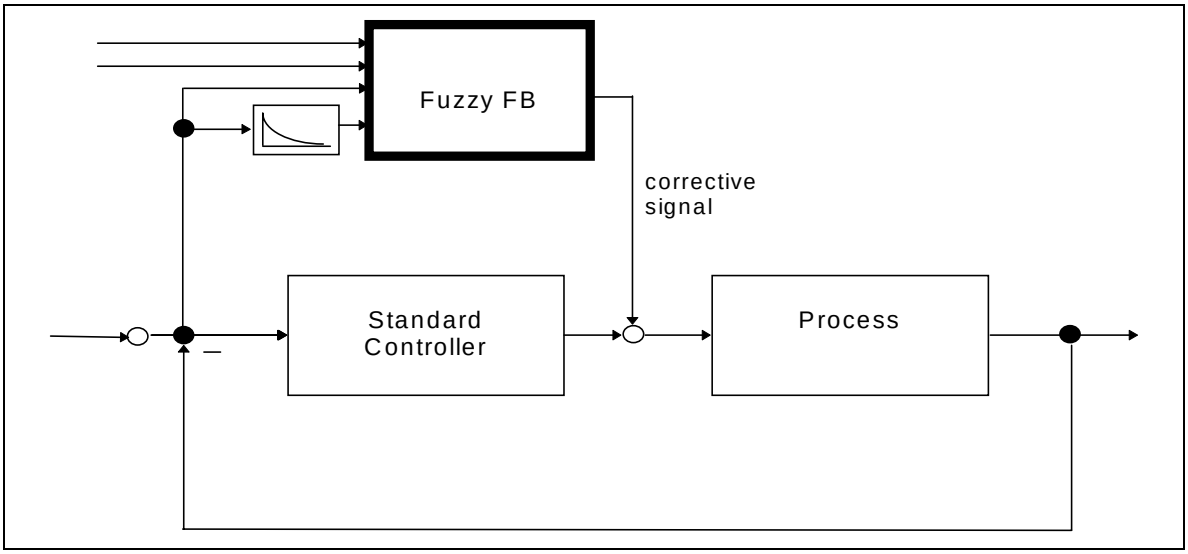


Figure B.1-1: Example of a pre-control

B.2 Parameter Adaptation of conventional PID controllers

The fuzzy controller is used to adapt the control parameters of a PID controller.

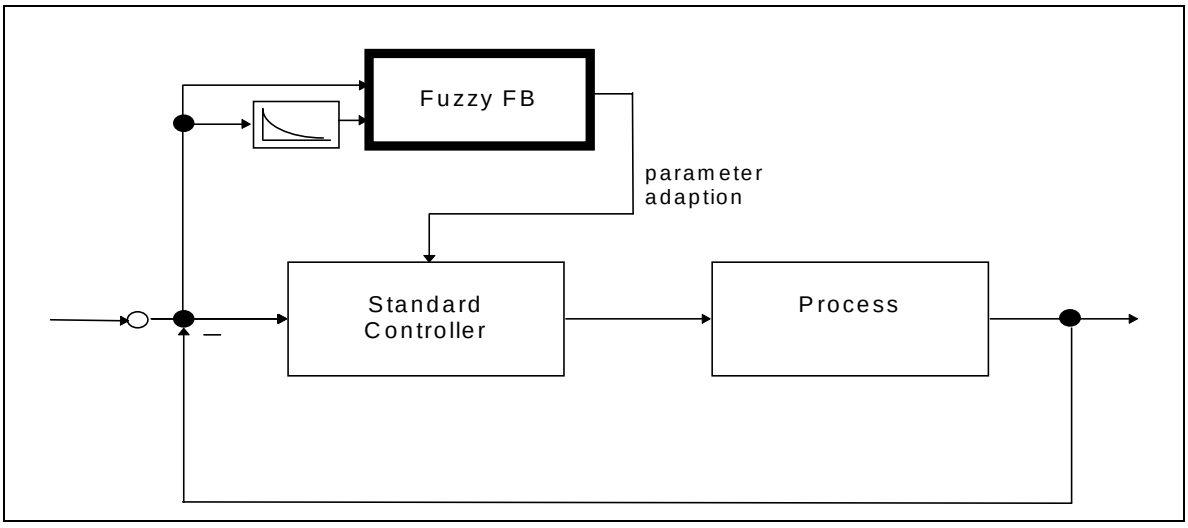


Figure B.2-1: Example of a parameter adaptation

B.3 Direct Fuzzy Control of a process

Another area of application is to include empirical process knowledge and linguistic control strategies directly into industrial automation. This applies to many processes where operator intervention would be necessary.

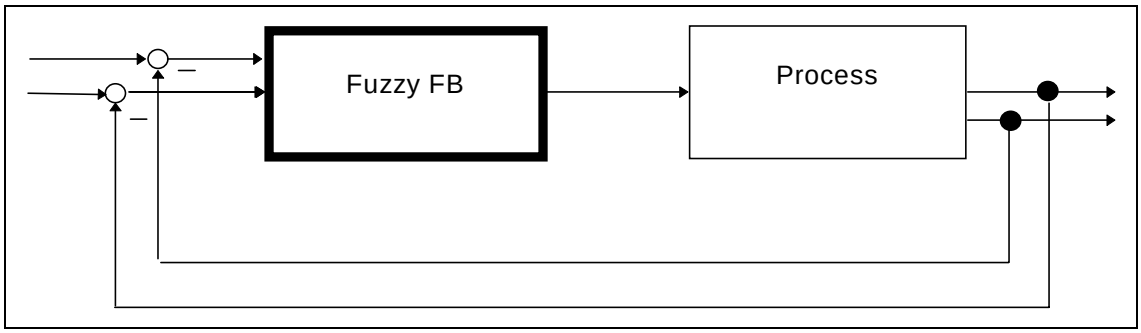


Figure B.3-1: Example of a direct fuzzy control

Annex C Industrial Example Container Crane

Container cranes are used to load and unload containers to and from ships in most harbors. They pick up single containers with flexible cables that are mounted at the crane head. The crane head moves on a horizontal track. When a container is picked up and the crane head starts to move, the container begins to sway as shown in figure C-1. While sway is no problem during transportation, a swaying container cannot be released.

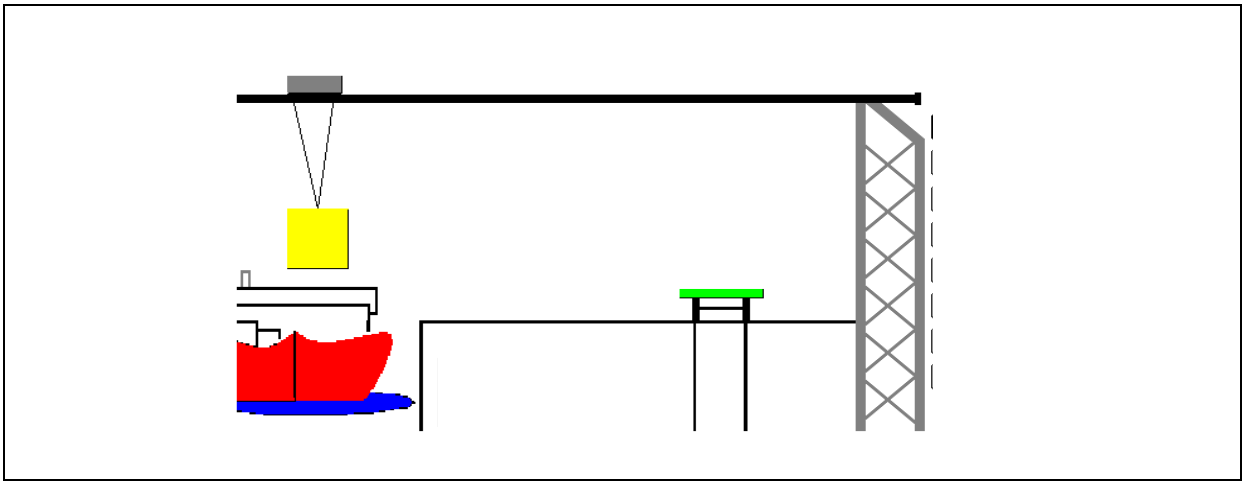


Figure C-1: Industrial example container crane

The analysis of the operator's actions reveals that the operator uses some "rules of thumb" to describe his control strategy.

1. Start with medium power.
2. If you got started and you are still far away from target, adjust the motor power so that the container gets a little behind the crane head.
3. If you are closer to the target, reduce speed so the container gets a little ahead of the crane head.
4. When the container is very close to target position, power up the motor.
5. When the container is over the target and the sway is zero, stop the motor.

To automate the control of this crane, sensors for the crane head position ("Distance") and the angle of the container sway ("Angle") are employed. The output is the motor power. First, linguistic variables have to be defined for all variables. The linguistic variables distance, angle and motor power are divided into 5 linguistic terms. The used forms of the membership function are ramps, triangles and singletons. The following figures show the linguistic variables and the linguistic terms.

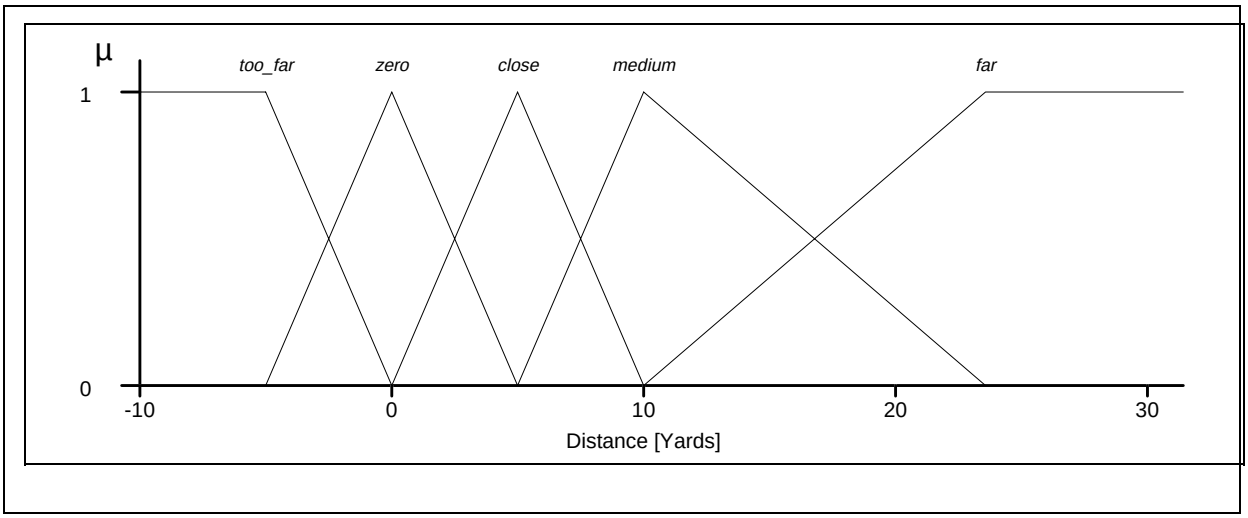


Figure C-2: Linguistic variable "Distance" between crane head and target position

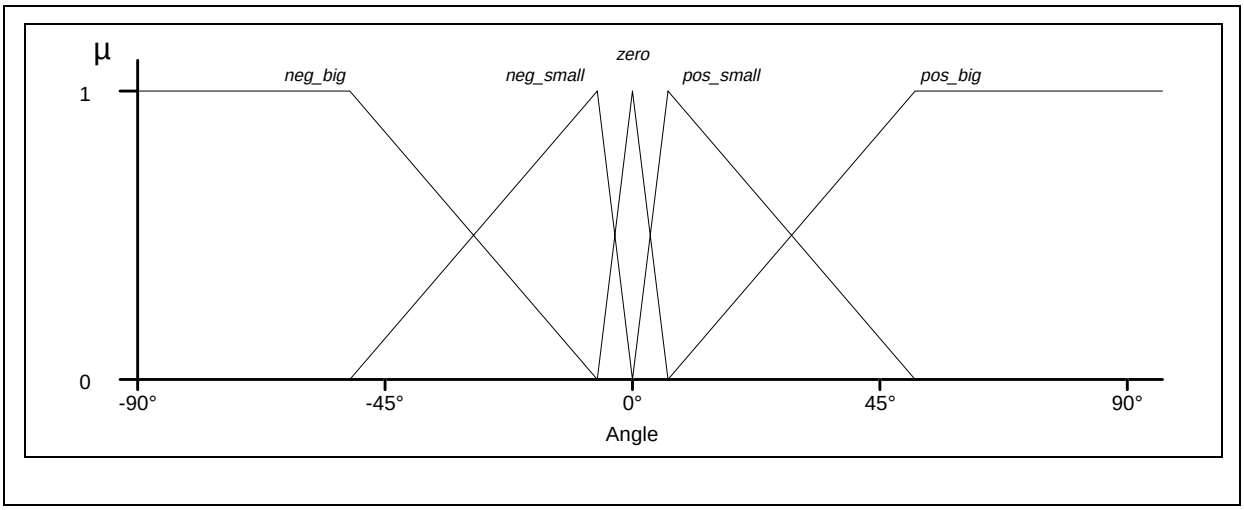


Figure C-3: Linguistic variable "Angle" of the container to the crane head

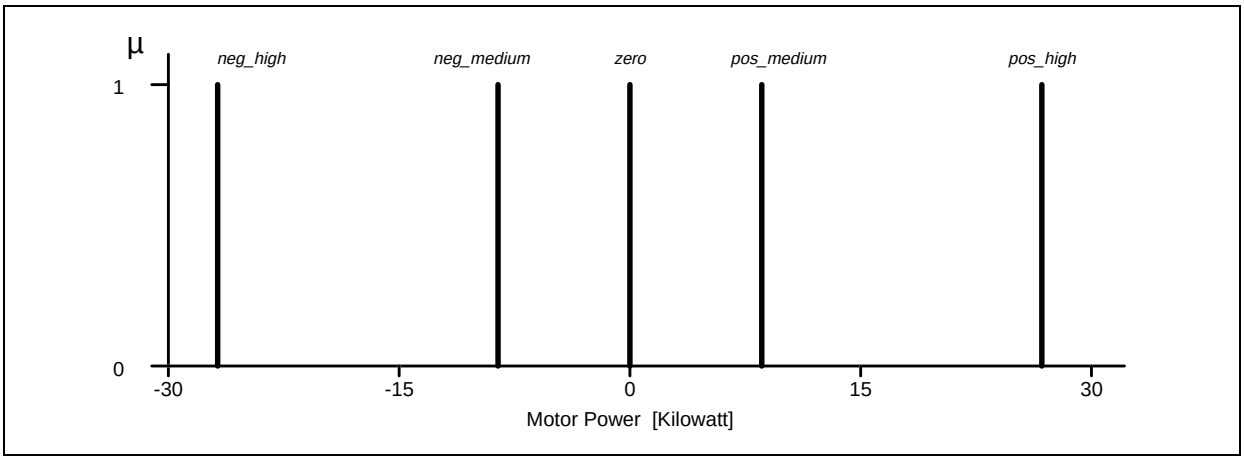


Figure C-4: Linguistic variable "Power"

Using these linguistic terms to describe the current condition of the crane, the five rules of thumb can be translated into a rule base. The next figure shows the definition of the rule base in FCL notation. Note, that rule 2 has been translated into two rules to fit the if-then format.

RULE 1: IF distance IS far	AND angle IS zero	THEN power IS pos_medium
RULE 2: IF distance IS far	AND angle IS neg_small	THEN power IS pos_big
RULE 3: IF distance IS far	AND angle IS neg_big	THEN power IS pos_medium
RULE 4: IF distance IS medium	AND angle IS neg_small	THEN power IS neg_medium
RULE 5: IF distance IS close	AND angle IS pos_small	THEN power IS pos_medium
RULE 6: IF distance IS zero	AND angle IS zero	THEN power IS zero

Figure C-5: Rule base

The next table shows the inference steps and the applied operators, respectively.

Table C-1: Inference steps and assigned operator

Inference step	*Operators*
Aggregation	
AND	*Minimum*
Activation	
conversion of the IF-THEN-conclusion	
	Minimum
Accumulation	*Maximum*

Consider a current situation of the crane, where the Distance of the crane head to the target position is 12 yards and the Angle of the container is +4°. For illustration a subset of three rules will be assumed.

Figures C-5 and C-6 show how the *fuzzification* is computed for this case.

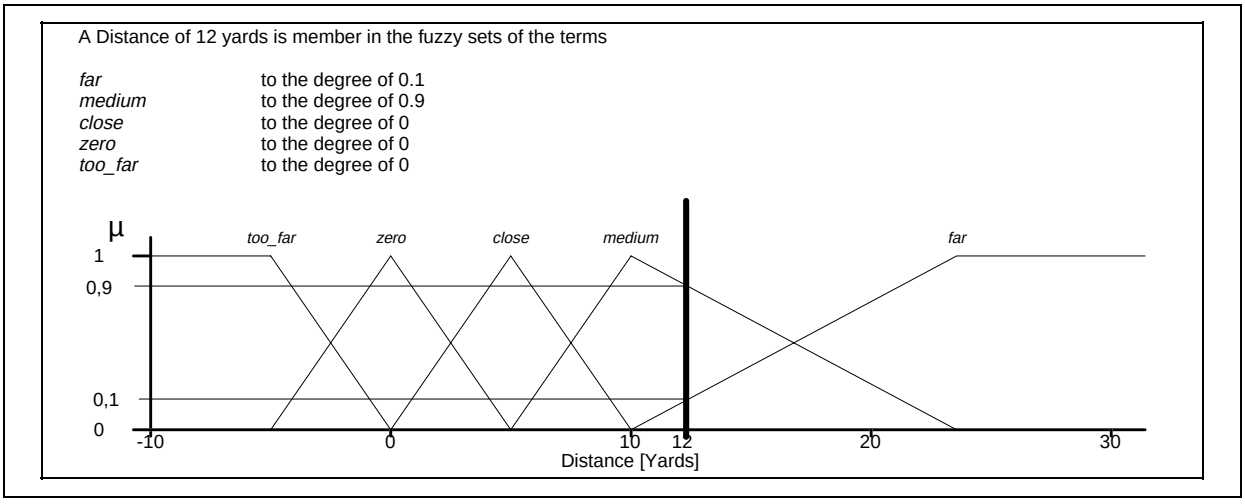


Figure C-5: Fuzzification of the linguistic variable distance

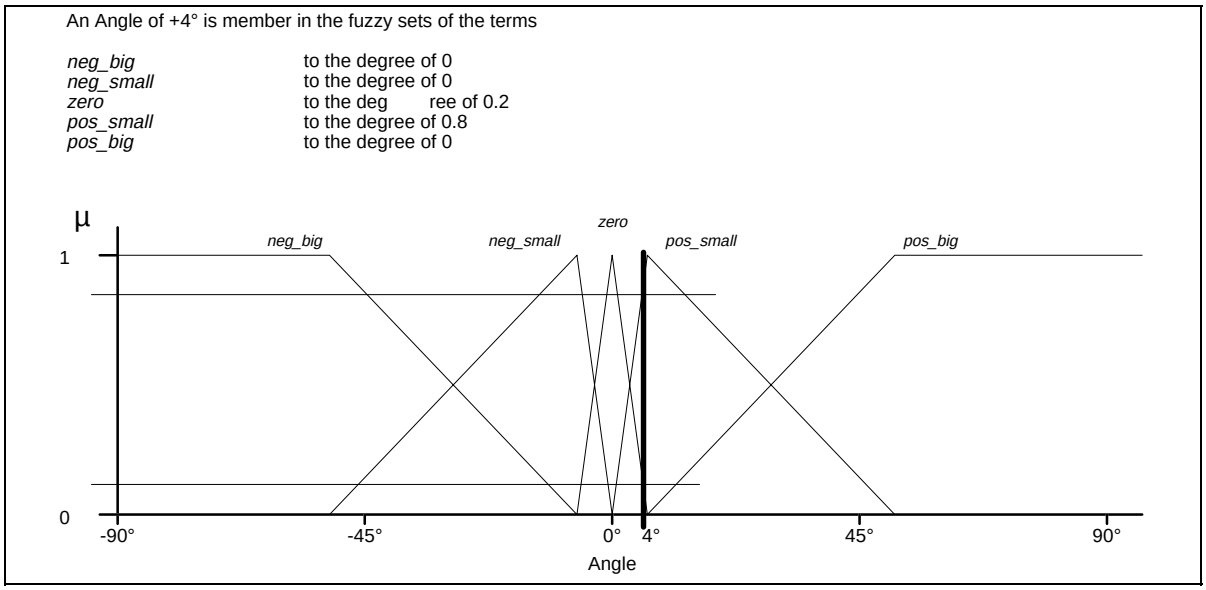


Figure C-6: Fuzzification of the linguistic variable angle

The Distance of 12 yards is translated into the linguistic variable value of {0.1, 0.9, 0, 0, 0} which can be interpreted as "still medium, just slightly far". The Angle of +4° is translated into the linguistic value of {0, 0, 0.2, 0.8, 0} which can be interpreted as "positive small, somewhat zero".

- Interference

Now that all input variables have been converted to linguistic variable values, the fuzzy *inference* step can identify the rules that apply to the current situation and can compute the values of the output linguistic variable. Figure C.7 shows a subset of three rules for illustration:

Rule 1: IF distance IS medium	AND angle IS pos_small	THEN power IS pos_medium
Rule 2: IF distance IS medium	AND angle IS zero	THEN power IS zero
Rule 3: IF distance IS far	AND angle IS zero	THEN power IS pos_medium

Figure C-7: Subset of three rules

The Inference consists of the three subfunctions aggregation, activation and accumulation.

- Aggregation

Determining the degree of conformance of the premise from the degree of membership of the elementary statements

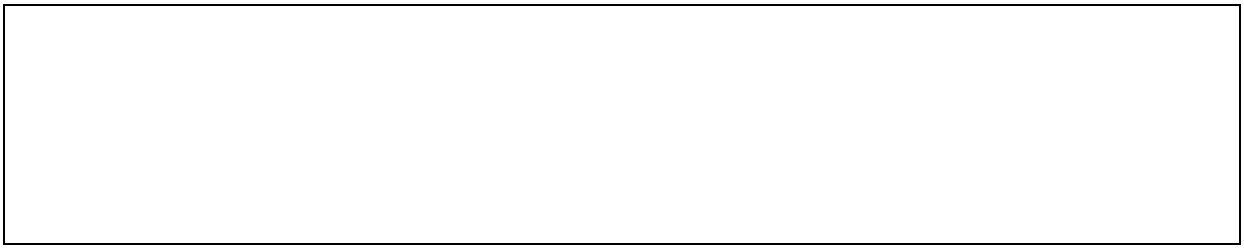


Figure C-8: Elements of aggregation

The Min-Operator corresponds to the AND-Aggregation. Figure C-9 shows how the *aggregation* is computed for this case.

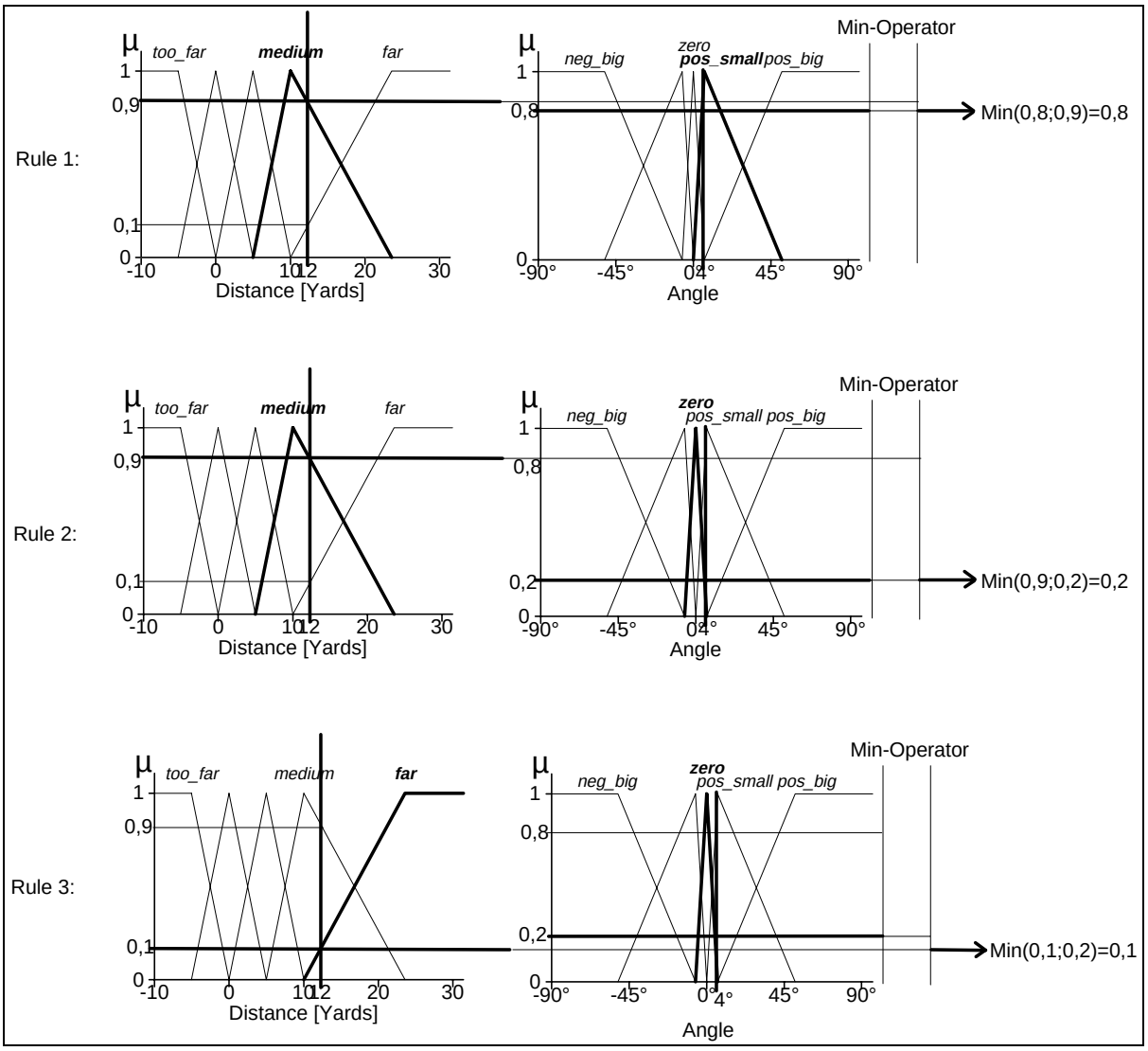


Figure C-9: Principles of aggregation

- Activation

Conversion of the IF-Then conclusion

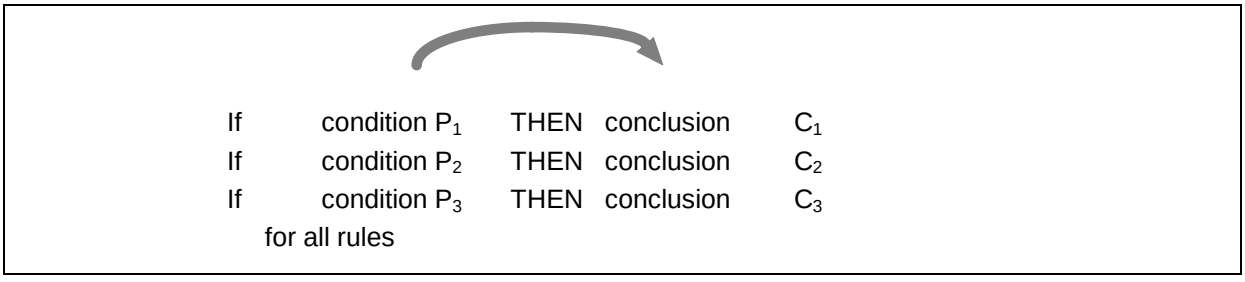


Figure C-10: Elements of activation

Figure C-11 shows how the activation is computed for this case. The result of the aggregation is described on the left side, the result of the activation on the right side.

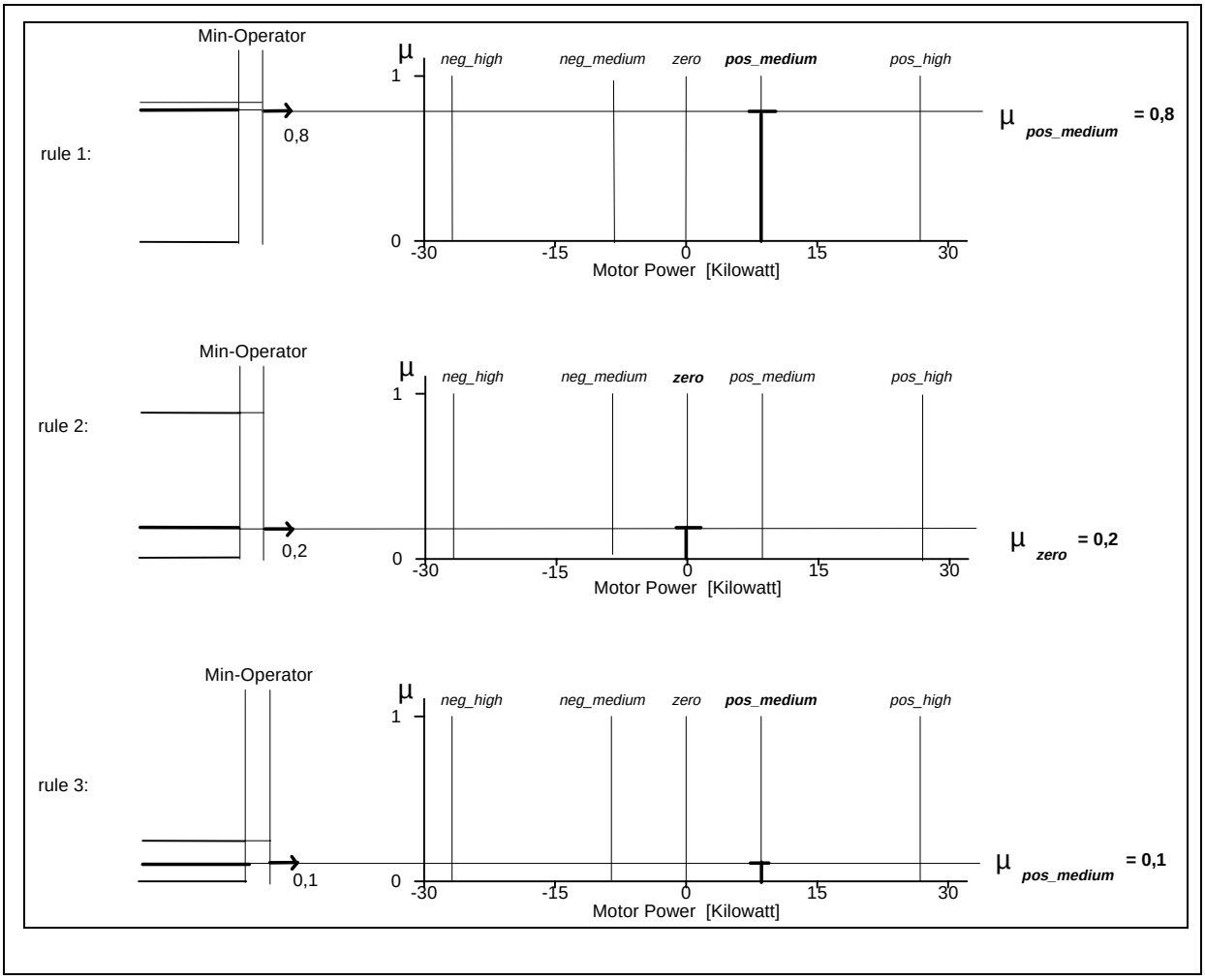


Figure C-11: Principles of activation

-Accumulation
Combination of the weighted results of the rules into an overall result

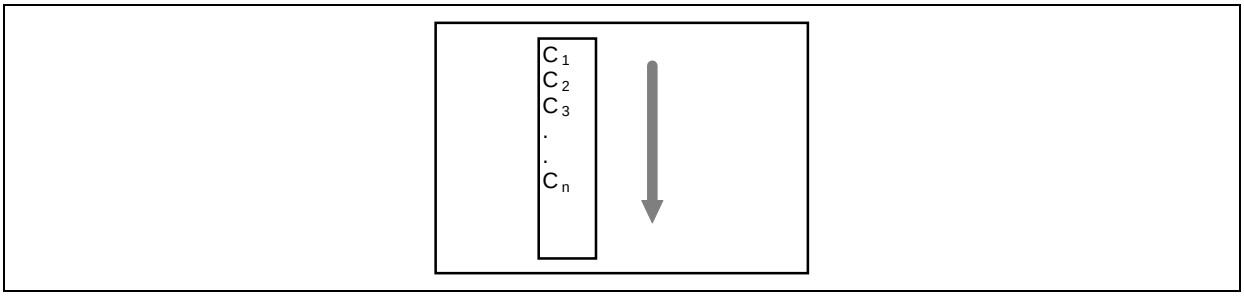


Figure C-12: Elements of Accumulation

The result of the accumulation of the rules 1-3 is shown at the bottom of the Figure C-13. The result of the singleton *pos-medium* for example, is calculated as Max(0,8; 0,1) = 0,8.

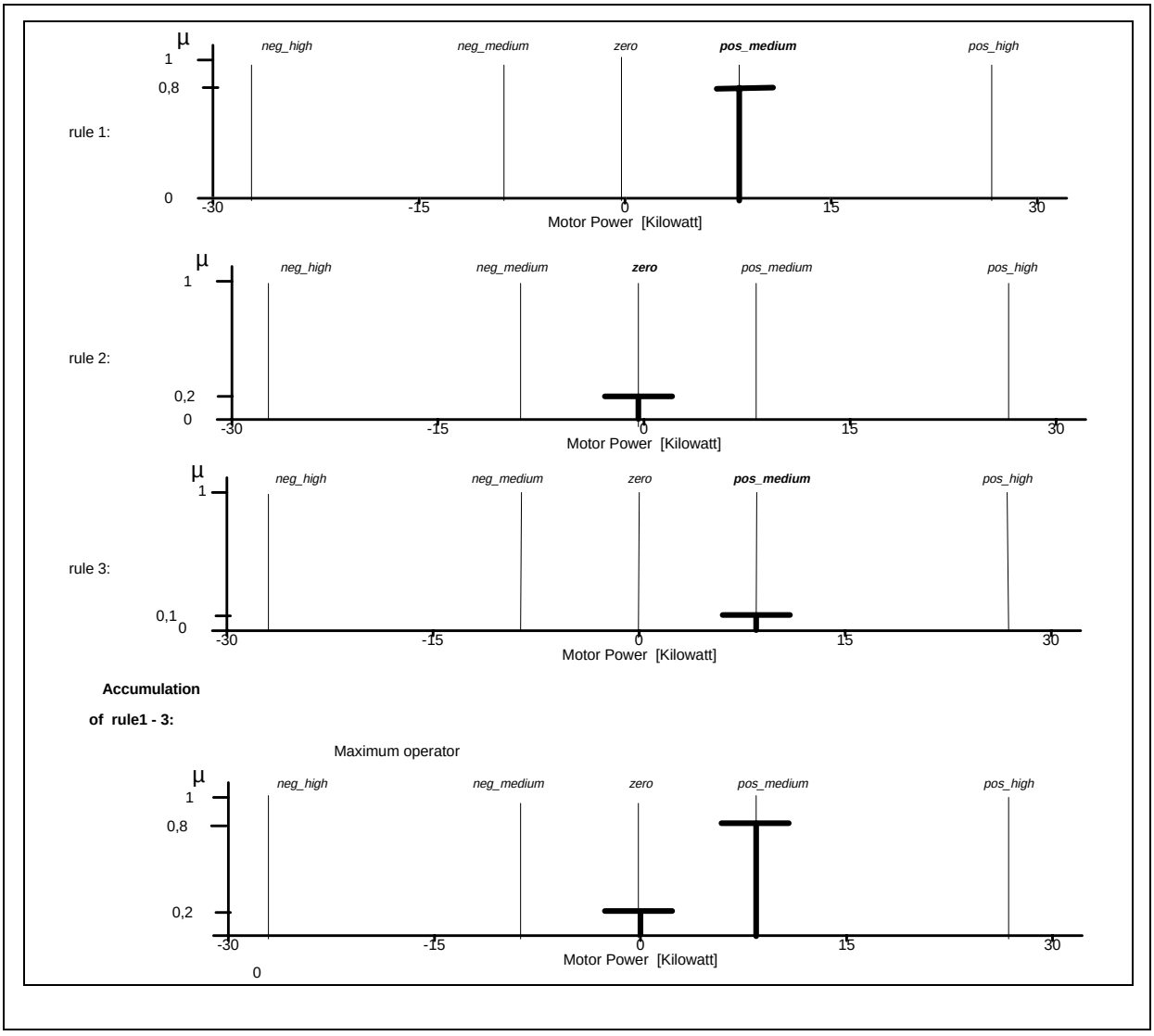


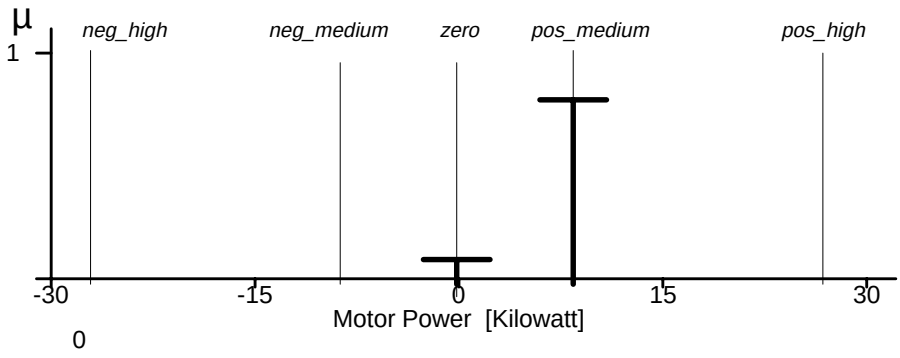
Figure C-13: Principles of Accumulation

- Defuzzification

Inference supplies a *Fuzzy Set* or its *membership function* as a result. To use it to set the motor power, it has to be converted into a *crisp* numerical value. In this context, the *value* to be determined (generally a real number) should provide the best possible representation of the information contained in the obtained *Fuzzy Set*. Using the *max height method*, the crisp output variable motor power is calculated as follows:

Accumulation

of rule1 - 3:



Defuzzification Max Height Method

In the Max Height Method, the value of the output variables is determined by the membership function of the output set with maximum degree of membership.

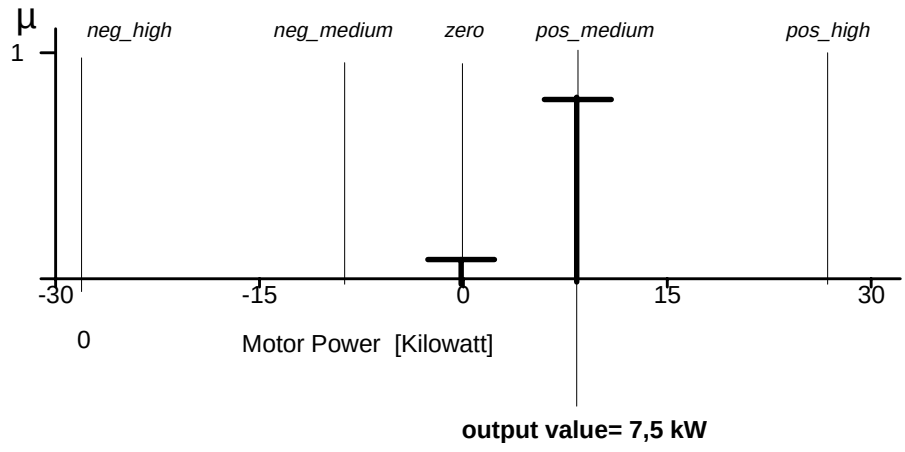


Figure C-14: Defuzzification

- Implementation of the container crane example in FCL:

```
FUNCTION_BLOCK container_crane

VAR_INPUT
        distance:       REAL;
        angle:          REAL;
END_VAR

VAR_OUTPUT
        power:          REAL
END_VAR

FUZZIFY distance
        TERM  too_far           := (-5, 1) ( 0, 0);
        TERM  zero              := (-5, 0) ( 0, 1) ( 5,0);
        TERM  close             := ( 0, 0) ( 5, 1) (10,0);
        TERM  medium            := ( 5, 0) (10, 1) (22,0);
        TERM  far               := (10, 0) (22,1);
END_FUZZIFY

FUZZIFY angle
        TERM  neg_big           := (-50, 1) (-5, 0);
        TERM  neg_small         := (-50, 0) (-5, 1) ( 0,0);
        TERM  zero              := ( -5, 0) ( 0, 1) ( 5,0);
        TERM  pos_small         := (  0, 0) ( 5, 1) (50,0);
        TERM  pos_big           := (  5, 0) (50, 1);
END_FUZZIFY

DEFUZZIFY power
        TERM  neg_high          := -27;
        TERM  neg_medium        := -12;
        TERM  zero              :=   0;
        TERM  pos_medium        :=  12;
        TERM  pos_high          :=  27;
        ACCU:  MAX;
        METHOD : COGS
        DEFAULT := 0
END_DEFUZZIFY

RULEBLOCK No1
        AND : MIN;
        RULE 1: IF distance IS far AND angle IS zero THEN power IS pos_medium;
        RULE 2: IF distance IS far AND angle IS neg_small THEN power IS pos_big;
        RULE 3: IF distance IS far AND angle IS neg_big THEN power IS pos_medium;
        RULE 4: IF distance IS medium AND angle IS neg_small THEN power IS neg_medium;
        RULE 5: IF distance IS close AND angle IS pos_small THEN power IS pos_medium;
        RULE 6: IF distance IS zero AND angle IS zero THEN power IS zero;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Figure C.15: Example in SCL

The FCL function block shall be invoked in the same manner as with programs according to Part 3. The reason for this is that from the external viewpoint the fuzzy block has its only interface via its variables.

According to Part 3, Programming languages, the call for the above example could be:

```
container_crane(distance := INP_DIS, angle := INP_ANG);
A := container_crane.power;
```

The variables INP_DIS and INP_ANG could have been bound directly to input words of the controller or be computed from other values.

Annex D Example for Using Variables in the Rule Block

Biscuits are cooked in a tunnel oven. The color of the biscuits is measured through a color sensor at the end of the oven. Color being a three-dimensional measurement, a fuzzy classification method is used to evaluate the membership of the measured color to the three following classes : Brown, Light, Dark. Humidity is also measured in the oven. The oven can be controlled through two temperature loops : one for the first half and one for the second half of the oven.

The principle of the control is given in the following figure.

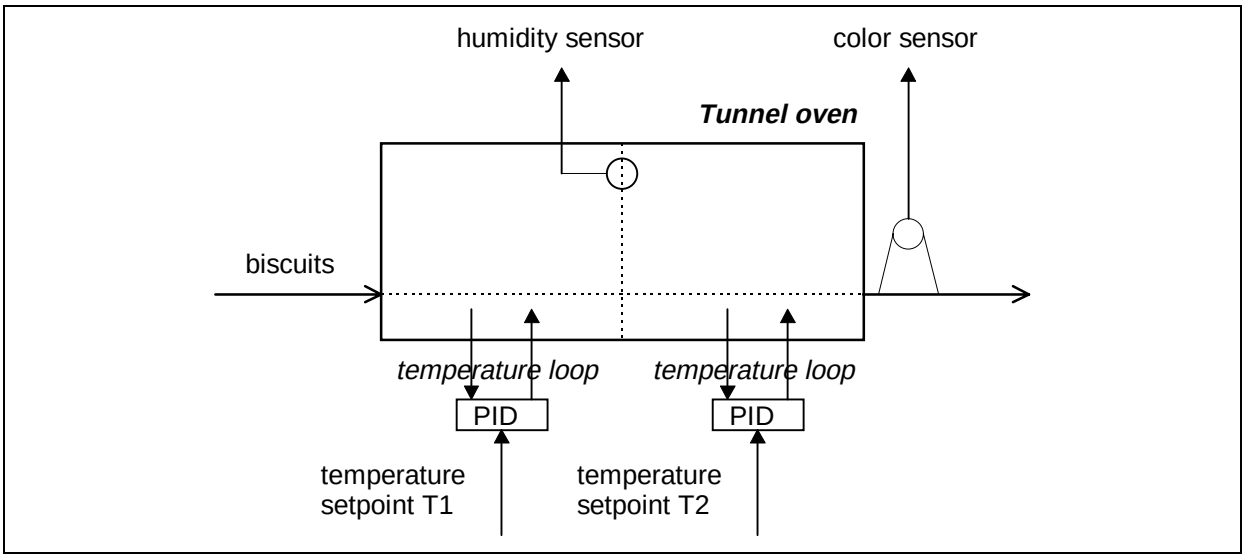


Figure D-1: Principle of the controlled system

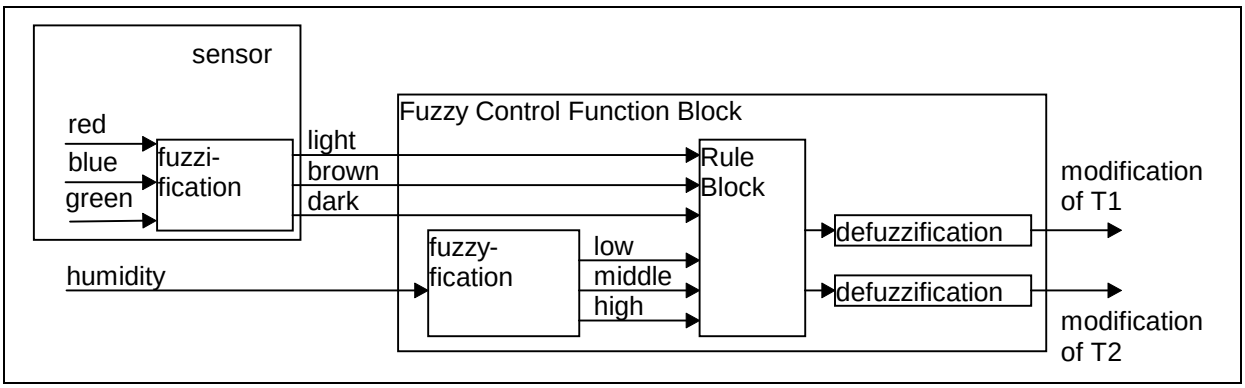


Figure D-2: Principle of the fuzzy based control of the oven

The rule block contains the following 5 rules :

IF humidity IS middle AND brown	THEN dT1 IS zero AND dT2 is zero
IF humidity IS high	THEN dT1 IS positive
IF humidity IS low	THEN dT1 IS negative
IF humidity IS middle AND light	THEN dT2 IS positive
IF humidity IS middle AND dark	THEN dT2 IS negative

Figure D-3: Rule block

The FCL syntax for this example is given in figure D-4.

Note: Instead of using the 3 variables light, brown and dark also only 1 variable of the type structure may be used as shown below.

```
FUNCTION_BLOCK oven_control
STRUCT color_type
        brown : REAL;
        light :   REAL;
        dark :    REAL;
END_STRUCT

VAR_INPUT
        humidity :   REAL;
        color : color_type
END_VAR

VAR_OUTPUT
        dT1 :        REAL;
        dT2 :        REAL;
END_VAR

FUZZIFY humidity
        TERM low          := (30,1) (50,0);
        TERM middle       := (30,0) (50,1) (70,1) (80,0);
        TERM high         := (70,0) (80,1);
END_FUZZIFY

DEFUZZIFY dT1
        TERM negative     := -5
        TERM zero         :=  0
        TERM positive     :=  5
        ACCU : MAX;
        METHOD : COGS;
        DEFAULT := 0
END_DEFUZZIFY

DEFUZZIFY dT2
        TERM negative     := -3
        TERM zero         :=  0
        TERM positive     :=  3
        ACCU : MAX;
        METHOD : COGS;
        DEFAULT := 0
END_DEFUZZIFY

RULEBLOCK inference
        RULE1 : IF humidity IS middle AND color.brown    THEN dT1 IS zero AND dT2 is zero
        RULE2 : IF humidity IS high                      THEN dT1 IS positive
        RULE3 : IF humidity IS low                       THEN dT1 IS negative
        RULE4 : IF humidity IS middle AND color.light    THEN dT2 IS positive
        RULE5 : IF humidity IS middle AND color.dark     THEN dT2 IS negative
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Figure D-4: Example in SCL

Annex E Symbols, Abbreviations, Synonyms

Table E-1: Symbols, Abbreviations

CoA	Center of Area	
CoG	Center of Gravity	
FB	Function Block	
FBD	Function Block Diagram	
FCL	Fuzzy Control Language	
IL	Instruction List	
ISO	International Standardization Organisation	
MAX	Maximum operator	
MIN	Minimum operator	
PROD	Product operator	
ST	Structured Text	
μ	Degree of Membership	
ω	weight factor	

Table E-1: Synonyms

conclusion	consequent
accumulation	result aggregation
activation	composition
membership function	antecedent
Centre of Gravity	Centroid of Area
Centre of Area	Bisector of Area

Annex F Index

[Note: This clause is not yet prepared]