BRITISH STANDARD

BS EN 61131-7:2001
IEC 61131-7:2000

Programmable controllers —

Part 7: Fuzzy control programming

The European Standard EN 61131-7:2000 has the status of a British Standard

ICS 25.040.40; 35.060; 35.240.50

National foreword

This British Standard is the official English language version of EN 61131-7:2000. It is identical with IEC 61311-7:2000.

The UK participation in its preparation was entrusted by Technical Committee GEL/65, Measurement and control, to Subcommittee GEL/65/2, Elements of systems, which has the responsibility to:

— aid enquirers to understand the text;

— present to the responsible international/ European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;

— monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this subcommittee can be obtained on request to its secretary.

From 1 January 1997, all IEC publications have the number 60000 added to the old number. For instance, IEC 27-1 has been renumbered as IEC 60027-1. For a period of time during the change over from one numbering system to the other, publications may contain identifiers from both systems.

Cross-references

Attention is drawn to the fact that CEN and CENELEC Standards normally include an annex which lists normative references to international publications with their corresponding European publications. The British Standards which implement these international or European publications may be found in the BSI Standards Catalogue under the section entitled “International Standards Correspondence Index”, or by using the “Find” facility of the BSI Standards Electronic Catalogue.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the EN title page, pages 2 to 57 and a back cover.

The BSI copyright notice displayed in this document indicates when the document was last issued.

This British Standard, having been prepared under the direction of the Electrotechnical Sector Committee, was published under the authority of the Standards Committee and comes into effect on 15 April 2001

ISBN 0 580 36843 2

Amendments issued since publication

Amd. No.	Date	Comments

EUROPEAN STANDARD **EN 61131-7**

NORME EUROPÉENNE

EUROPÄISCHE NORM December 2000

ICS 35.240.50;25.040.40

English version

Programmable controllers
Part 7: Fuzzy control programming
(IEC 61131-7:2000)

Automates programmables
Partie 7: Programmation en logique floue
(CEI 61131-7:2000)

Speicherprogrammierbare Steuerungen
Teil 7: Fuzzy-Control-Programmierung
(IEC 61131-7:2000)

This European Standard was approved by CENELEC on 2000-11-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CENELEC
European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Ref. No. EN 61131-7:2000 E

Foreword

The text of document 65B/406/FDIS, future edition 1 of IEC 61131-7, prepared by SC 65B, Devices, of IEC TC 65, Industrial-process measurement and control, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61131-7 on 2000-11-01.

The following dates were fixed:

– latest date by which the EN has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2001-08-01

– latest date by which the national standards conflicting with the EN have to be withdrawn (dow) 2003-11-01

Annexes designated "normative" are part of the body of the standard.
Annexes designated "informative" are given for information only.
In this standard, annex ZA is normative and annexes A, B, C, D and E are informative.
Annex ZA has been added by CENELEC.

———————

Endorsement notice

The text of the International Standard IEC 61131-7:2000 was approved by CENELEC as a European Standard without any modification.

———————

CONTENTS

INTRODUCTION

The theory of fuzzy logic in the application of control is named fuzzy control. 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 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 or when the model is too complicated to evaluate 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 realized in the area of fuzzy control differ widely in terms of terminology (definitions), features (functionalities) and implementation (tools).

Fuzzy control is used from small and simple applications up to highly sophisticated and complex projects. To cover all kinds of usage in this part of IEC 61131, the features of a compliant fuzzy control system are mapped into defined conformance classes.

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. This standard does not force all compliant systems to realize 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 realized 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 part of IEC 61131 and not to hinder the progress of future development, this standard permits also additional non-standard features which are not covered by the basic class and the extension class. 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.

PROGRAMMABLE CONTROLLERS –

Part 7: Fuzzy control programming

1 Scope and object

This part of IEC 61131 defines a language for the programming of Fuzzy Control applications used by programmable controllers.

The object of this part of IEC 61131 is to offer the manufacturers and the users a well-defined common understanding of the basic means to integrate fuzzy control applications in the Programmable Controller languages according to IEC 61131-3, as well as the possibility to exchange portable fuzzy control programs among different programming systems.

To achieve this, annex A gives a short introduction to the theory of fuzzy control and fuzzy logic as far as it is necessary for the understanding of this part of IEC 61131. It may be helpful for readers of this part of IEC 61131 who are not familiar with fuzzy control theory to read annex A first.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61131. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 61131 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

IEC 60050-351:1998, *International Electrotechnical Vocabulary (IEV) – Part 351: Automatic control*

IEC 61131-3:1993, *Programmable controllers – Part 3: Programming languages*

3 Definitions

For the purpose of this part of IEC 61131, the following definitions apply.

Further definitions for language elements are given in IEC 61131-3.

NOTE Terms defined in this clause are italicized where they appear in the text of definitions.

3.1
accumulation
result aggregation

combination of results of *linguistic rules* in a final result

3.2
aggregation
determination of degree of firing

combination of membership degrees of all individual subconditions in a rule to calculation of the degree of accomplishment of the *condition* of a rule

3.3
activation

process by which the degree of fulfilment of a *condition* acts on an output fuzzy set

3.4
conclusion
consequent

output of a *linguistic rule*, i.e. the actions to be taken (the THEN part of an IF..THEN fuzzy control rule)

3.5
condition
antecedent

expression comprising *subconditions* combined with *fuzzy operators* AND, OR, NOT

3.6
crisp set
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
determination of *degrees of membership* of the crisp input value of the *linguistic terms* defined with each input *linguistic variable*

3.10
fuzzy control
type of control in which the control algorithm is based on *fuzzy logic*

[IEV 351-17-51, modified]

3.11
fuzzy logic
collection of mathematical theories based on the notion of *fuzzy set*. *Fuzzy logic* is a kind of infinite-valued logic

3.12
fuzzy 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 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
function which defines the *degree of membership* over the universe of discourse for a given *fuzzy set*
[IEV 351-17-52, modified]

3.19
singleton
fuzzy set whose *membership function* is equal to one at one point and equal to zero at all other points

3.20
subcondition
elementary expression in the form of a variable or as term "*linguistic variable* IS *linguistic term*"

3.21
rule base
collection of *linguistic rules* to attain certain objectives

3.22
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 shall be encapsulated in Function Blocks (or Programs) as defined in IEC 61131-3. The concept of Function Block Types and Function Block Instances given in IEC 61131-3 applies to this standard.

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 Function Block Instances shall contain the specific data of the fuzzy control applications.

Function Blocks defined in Fuzzy Control Language FCL may be used in Programs and Function Blocks written in any of the languages of IEC 61131-3, for example, 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 1.

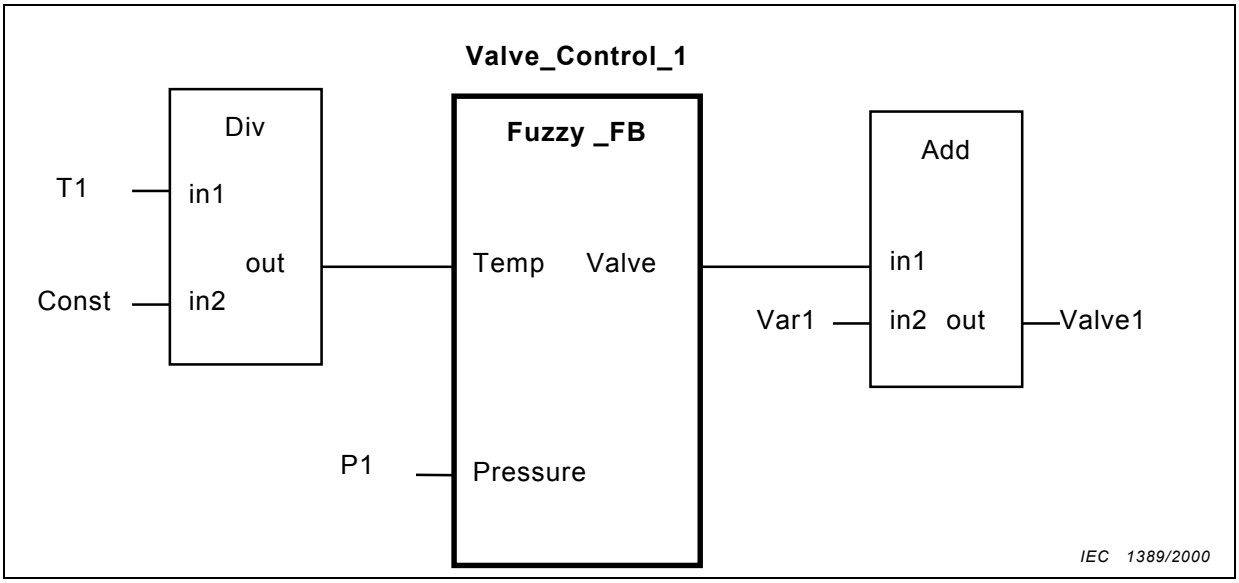


Figure 1 – Example of a fuzzy control Function Block in FBD representation

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. 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 IEC 61131-3.

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 IEC 61131-3. 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 IEC 61131-3. 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 this clause.

The language elements of FCL standardize a common representation for data exchange among fuzzy control configuration tools of different manufacturers shown in figure 2. Using this common representation, every manufacturer of programmable controllers may keep his hardware, software editors and compilers. The manufacturer has only to implement the data interface into his specific editor. The customer would be able to exchange fuzzy control projects between different manufacturers.

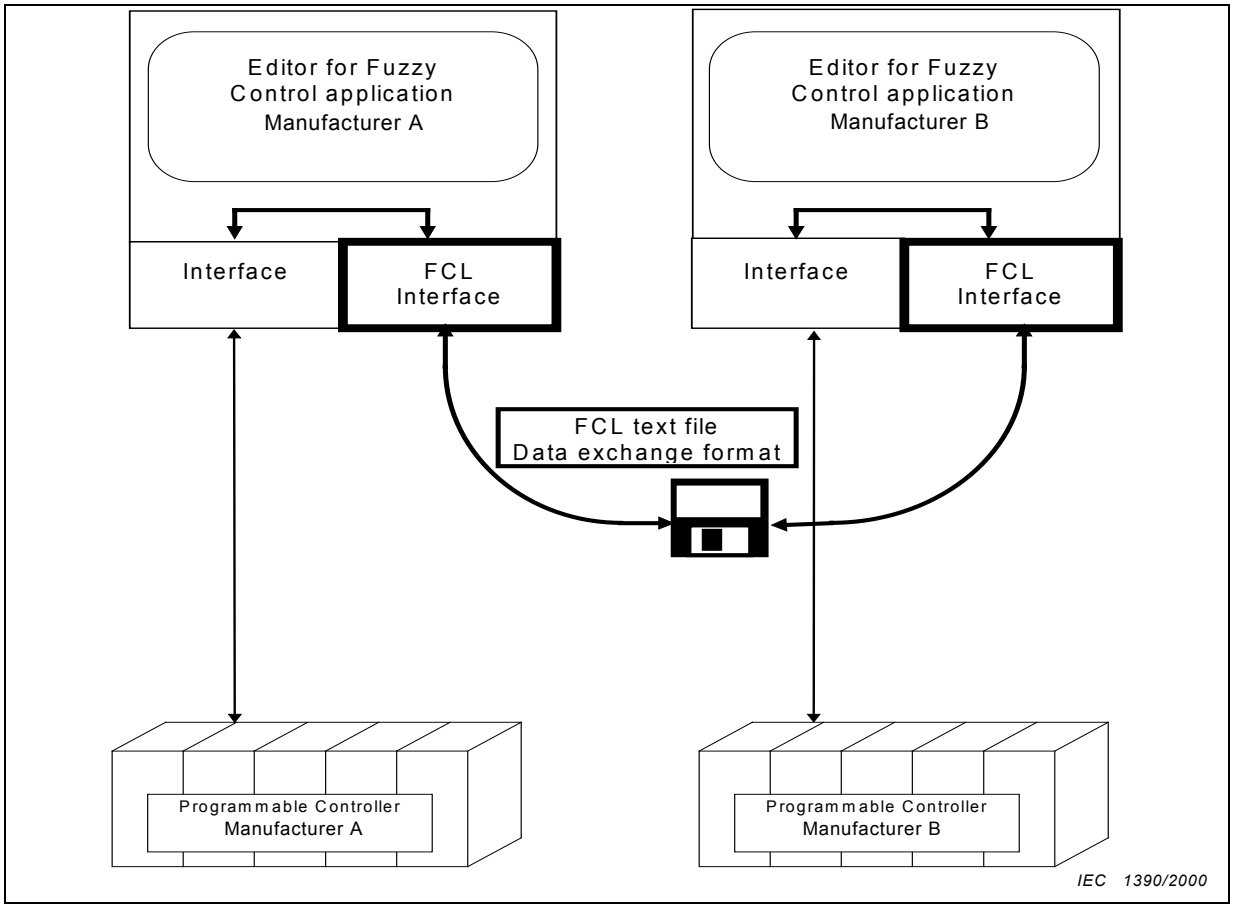


Figure 2 – Data exchange of Programs in Fuzzy Control Language (FCL)

5.2 Fuzzy Control Language elements

Fuzzy control language elements in this subclause are described using examples. The detailed production rule is given in 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 IEC 61131-3 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 IEC 61131-3.

Figure 3 shows an example of 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
```

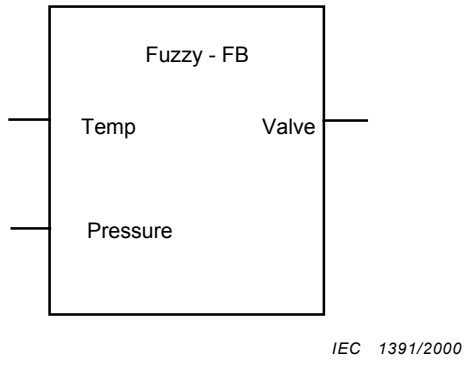
</td>
<td> 
</td>
</tr>
<tr>
<td>Structured Text (ST)</td>
<td>Function Block Diagram (FBD)</td>
</tr>
</table>

Figure 3 – 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, for example ramp and triangle, may 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 minimum number of points is two. The maximum number is restricted according to clause 6 conformance classes.

Example of *membership function* with three 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 look-up 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 look-up table, all values greater than the last point in the look-up 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
```

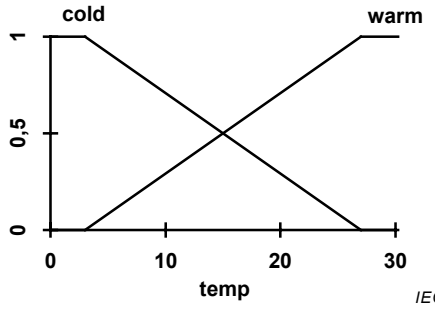


Figure 4 – Example of ramp terms

NOTE The data type of the points of membership functions is not defined. The manufacturer must provide a compiler that accommodates any necessary conversion.

In order to adapt the fuzzy control application on-line, the base points in the membership functions can be modified. This may 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 for the membership functions is given in figure 5.

NOTE The values of membership function 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
                                                              IEC 1393/2000
```

Figure 5 – 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
        RANGE(min..max);
        TERM term_name:= membership_function;
        defuzzification_method;
        default_value;
END_DEFUZZIFY
```

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

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 6, examples of terms are given.

Definition of *singletons:*

```
DEFUZZIFY valve
RANGE(–100,100);
        TERM drainage := –100;
        TERM closed   :=  0;
        TERM inlet    := 100;
END_DEFUZZIFY
```

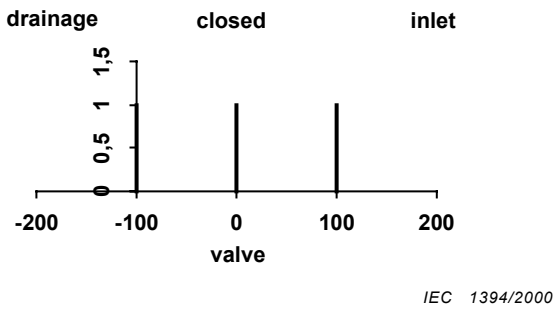


Figure 6 – Example of singleton terms

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

```
METHOD: defuzzification_method;
```

The following defuzzification methods are allowed (see tables 1 and 2).

Table 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 (note 4)
RM	Right Most Maximum (note 4)
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.	
NOTE 4 LM and RM defuzzification methods are asymmetrical about zero.	

Table 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	$U = \inf(u'), \mu(u') = \inf_{u \in [Min, Max]} \mu(u)$

where

U	is the result of defuzzification;
u	is the output variable;
p	is the number of singletons;
μ	is the membership function of accumulated fuzzy sets;
i	is the index;
Min	is the minimum value for defuzzification defined in RANGE. In the case of singletons, Min = –infinity;
Max	is the maximum value for defuzzification defined in RANGE. In the case of singletons, Max = +infinity;
sup	is the largest value;
inf	is the 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 allowed 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 keyword 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 limiting each membership function to the range of each output variable. If singletons are used for output membership functions, the RANGE has no effect.

If there is no range defined the default range shall be the range of the data type of the variable specified in IEC 61131-3.

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 allowed. Each rule block shall have 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; for example MAX shall be used for OR if MIN is used for AND.

Table 3 – 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 may be used (see table 4):

Table 4 – Activation methods

Name	**Keyword**	**Algorithm**
Product	PROD	$\mu_1(x)\ \mu_2(x)$
Minimum	MIN	Min($\mu_1(x)$, $\mu_2(x)$)
NOTE The activation method is not relevant for singletons.		

The following language element defines the method of the accumulation:

ACCU: *accumulation_method*;

The following accumulation methods may be used (see table 5):

Table 5 – Accumulation methods

Name	Keyword	Formula
Maximum	MAX	MAX ($\mu_1(x)$, $\mu_2(x)$)
Bounded sum	BSUM	MIN (1, $\mu_1(x) + \mu_2(x)$)
Normalized sum	NSUM	$\dfrac{\mu_1(x) + \mu_2(x)}{\text{MAX}\,(1, \text{MAX}_{x' \in X}\,(\mu_1(x') + \mu_2(x')))}$

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.

The rules are defined inside the rule block. 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 allowed 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 keywords IF and THEN, and combined by the operators with the keywords AND, OR or NOT.

The priority of the operator (see table 6) is handled according to Boolean algebra given in table 3.

Table 6 – 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 may 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*

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 may be used.

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

The *conclusion* may 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 := *linguistic_ variable* IS *linguistic_ 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 allowed to give each subconclusion a *weighting factor* which is a number with a value between 0.0 and 1.0, or a variable. 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 sub-conclusion 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* may 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 control 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 to clause 6.

5.3 FCL example

An example in Fuzzy Control Language is given in figure 7.

```
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;
        METHOD: CoGS;
        DEFAULT         := 0;
END_DEFUZZIFY
RULEBLOCK No1
        AND: MIN;
        ACCU: MAX;
        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
```

IEC 1395/2000

Figure 7 – Example for fuzzy function block

5.4 Production rules and keywords of the Fuzzy Control Language (FCL)

Annex A of IEC 61131-3 defines the specification method for textual languages for programmable controllers. This specification method is used here for FCL.

Annex B of IEC 61131-3 defines the formal specification of language elements for the textual programming languages of IEC 61131-3. For FCL the following subset of language elements of annex B of IEC 61131-3 is used:

B.1.1 Letters, digits and 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 IEC 61131-3, the following language elements may 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
                                       [range]
                                       {linguistic_term}
                                       defuzzification_method
                                       default_value
                                 'END_DEFUZZIFY'

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

option_block ::=                 'OPTION'
                                       any manufacturer specific parameter
                                 'END_OPTION'

linguistic_term ::=              'TERM' term_name ':=' membership_function ';'

membership_function ::=          singleton | points
```

NOTE For use of singletons, refer to 5.2.3.

```
singleton ::=                    numeric_literal  | variable_name

points ::=                       {'('numeric_literal | variable_name ','
                                    numeric_literal ')'}
```

NOTE Refer to 5.2.2 for the allowed number of points.

```
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’)] ‘;’

NOTE See table 3 for paired algorithms.

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

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

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

condition ::= x{(‘AND’ x)|(‘OR’ x)}

x ::= [‘NOT’] (subcondition | (‘ (‘ condition ’)’))

subcondition ::= variable_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

ruleblock_name ::= identifier

term_name ::= identifier

f_variable_name ::= identifier

variable_name ::= identifier

numeric_literal ::= integer_literal | real_literal

input_declarations ::= see IEC 61131-3, annex B

output_declarations ::= see IEC 61131-3, annex B

var_declarations ::= see IEC 61131-3, annex B

identifier ::= see IEC 61131-3, annex B

5.4.2 Keywords

Table 7 – Reserved keywords for FCL

Keyword	Meaning	Subclause
( )	Parentheses in condition, term, range	5.2.4
ACCU	Accumulation method	5.2.4
ACT	Activation 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, OR operator, Bounded sum	5.2.4
CoA	Centre of area defuzzification method	5.2.3
CoG	Centre of gravity defuzzification method	5.2.3
CoGS	Centre of gravity defuzzification of singletons	5.2.3
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	5.2.2
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, OR operator	5.2.4
METHOD	Method of defuzzification	5.2.3
MIN	Minimum as AND operator, activation method	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, activation method	5.2.4
RANGE	Limits for membership functions	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 8. The hierarchy consists of the following three levels.

– Basic Level including the definitions of the function block and the data types of IEC 61131-3.

– Extension level with optional features, listed in table 9 and optionally allowed within this level.

– Open Level encompassing additional features, not defined in this part of IEC 61131, but listed by the manufacturer.

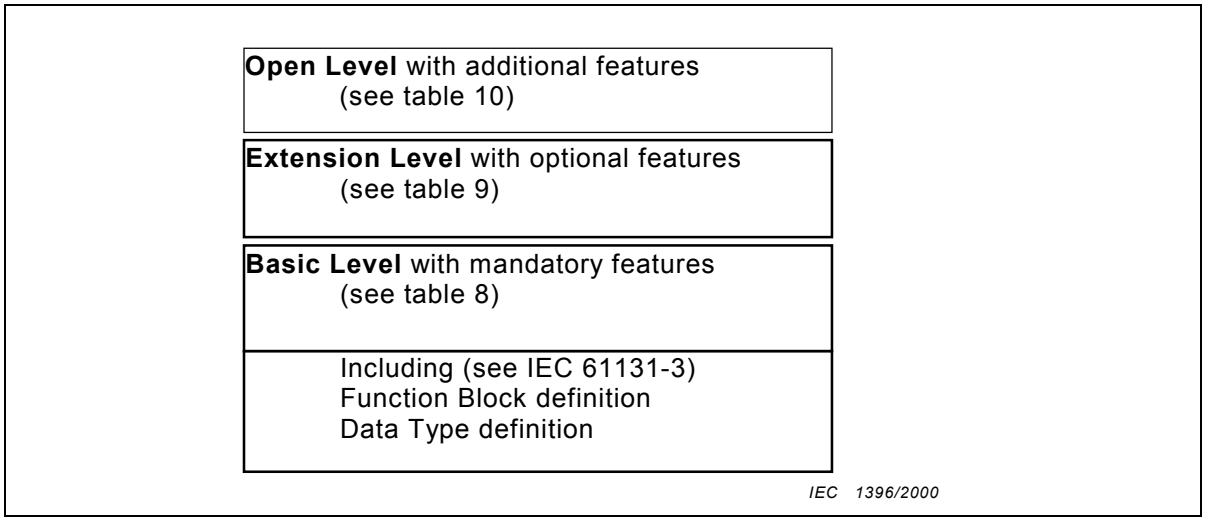


Figure 8 – Levels of conformance

A Control System using the Fuzzy Control Language FCL claiming to conform with this part of IEC 61131 shall comply with the following rules:

a) it shall use the Function Block features according to IEC 61131-3 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 fuzzy control function block shall be in accordance with IEC 61131-3;

b) all features of fuzzy control functionality defined in table 8 shall be implemented according to the provisions of this part. This table defines the set of Basic Level elements which all control systems complying with this part of IEC 61131 shall have in common;

c) a subset of the Extension Level elements defined in table 9 are additional elements which may be implemented optionally, in addition to all features of the Basic Level. The implementation shall strictly comply with the provisions of this part. These features shall be marked as Standard Extensions, and a list of realised features in the form of table 9 shall be part of the system documentation;

d) further features exceeding the Basic Level and the Extended Level may be realized 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 10 shall be part of the system documentation;

e) 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 provisions in this standard. This format shall be made available on systems complying with this part of IEC 61131 as input and output formats;

f) 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 may be realized by any graphical or textual means.

The elements in table 8 are the basic set of features, which shall be realized in all fuzzy control systems complying with this part of IEC 61131.

Table 8 – 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 constant points (degree of membership coordinate = 0 or 1)
	output variable: TERM	constant singletons only
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 value	DEFAULT	NC, value
ruleblock	RULEBLOCK	one ruleblock only
condition	IF ... IS ...	n subconditions
conclusion	THEN	only one subconclusion

The elements in table 9 are the extended set of features, which may be optionally realized in a standard fuzzy control system (e.g. for the AND operator the algorithm PROD or BDIF or both might be chosen). These optional features are in addition to all features of the Basic Level.

Table 9 – 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 constant or variable points (degree of membership co-ordinate = 0 or 1)
	output variable:TERM	maximum of four constant or variable 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
range for fuzzification	operator: RANGE	RANGE (minimum value..maximum value), limits the range of the membership functions for the output variable
defuzzification method	operator: METHOD	algorithm: CoG , CoA , LM , RM
ruleblock	operator: RULEBLOCK	n rule blocks
condition	IF	n subconditions, n input variables
conclusion	THEN	n subconclusions, n output variables
weighting factor	WITH	constant value and value assigned to variable in the declaration part VAR_INPUT.....END_VAR

The table 10 shows an example of a list of language elements in the Open Level. This list shall be a part of the system documentation.

Table 10 – Examples of a 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
variable membership values

6.2 Data check list

This data check list (see table 11) shall be delivered within the technical documentation. In this list, a manufacturer of programmable controllers, fuzzy control programming tools and application software shall describe specific performance features of the fuzzy control system. In order to facilitate the transfer of fuzzy control applications among different manufacturers' systems, the following Data check list is the means of verifying a possible program transfer. This list is not intended to be comprehensive but serves as an example only and could be enhanced by manufacturers.

Table 11 – 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*
execution time (ms)	20, 30
memory requirements (kb)	3, 4
mapping of the variable values of weighting factors and membership degrees from 0,0 to 1,0 onto the range of integer values	0-200, 0-400
length of identifiers (e.g. name of variables, rule blocks, 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
(informative)

Theory

This annex is 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. 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 may not be given for "adult".

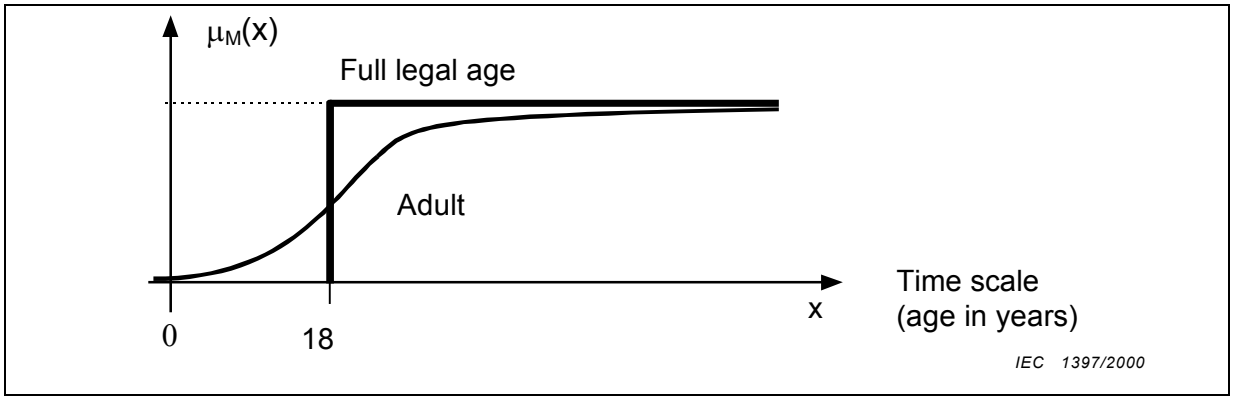


Figure A.1 – Membership functions of the terms "full legal age" and "adult"

As an example, figure A.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*.

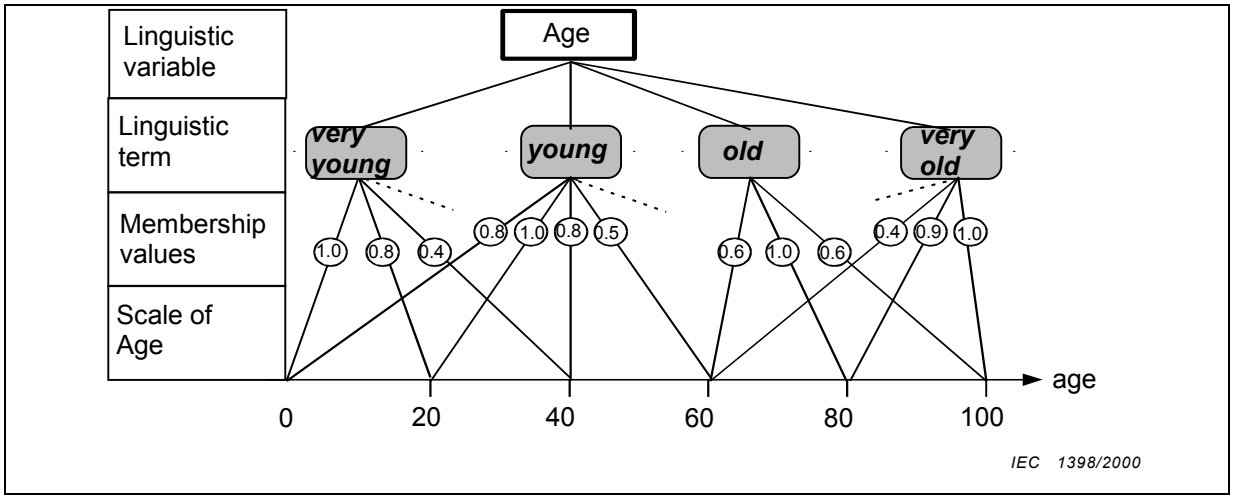


Figure A.2 – Description of the linguistic variable "Age" by linguistic terms and their hierarchy on the time scale (age in years)

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

– the definition via the rectangle (e.g. interval) in order to describe *values*, as well as

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

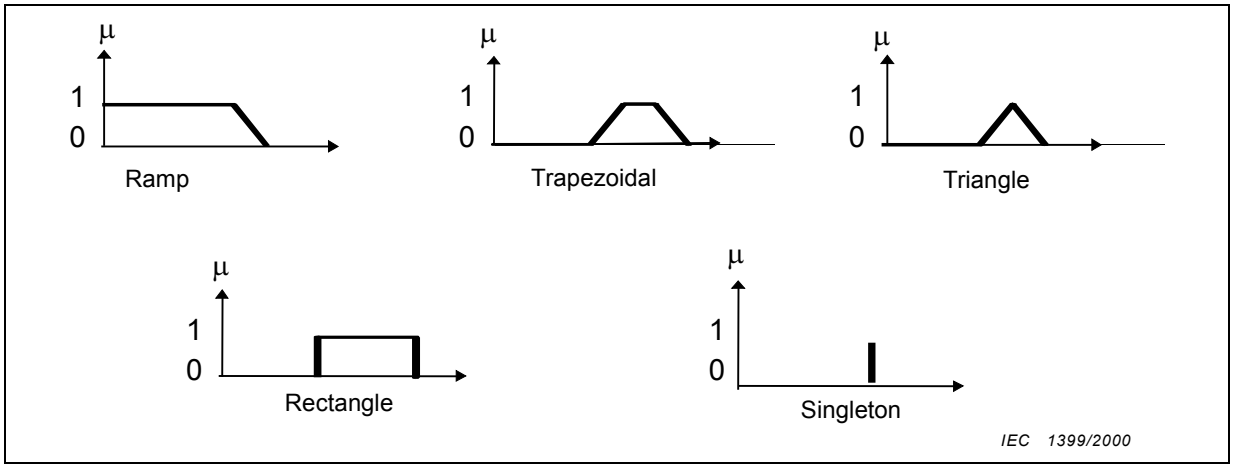


Figure A.3 – Commonly used shapes of membership functions

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 may 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,

$$\text{is true if } \mu_A(x) = \mu_B(x), \quad \text{for all } x \in G \tag{A.6}$$

complete inclusion $A \subseteq B$,

$$\text{is true if } \mu_A(x) \leq \mu_B(x), \quad \text{for all } x \in G \tag{A.7}$$

partial inclusion $A \subset B$,

$$\text{is true if } \mu_A(x) \leq \mu_B(x), \quad \text{for all } x \in G$$
$$\text{and } \mu_A(x) < \mu_B(x) \quad \text{for at least one } x \in G. \tag{A.8}$$

The following *operations* may be stipulated between two *fuzzy sets*, A and B, whose elements x originate from a basic set G:

the intersection

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

where I is called the intersection operator

the union

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

where U is called the union operator

the complement

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

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

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

Elementary algorithms for mathematical implementation of intersection, union and complement are the following (see also figure A.4):

for the *intersection,* the minimum

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

for the *union*, the maximum

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

for the *complement*, subtraction from one

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \qquad x \in G \tag{A.14}$$

There are a lot of possible algorithms for AND and OR fuzzy operators. It is worth noting here, that AND and OR operators cannot be chosen in an arbitrary way.

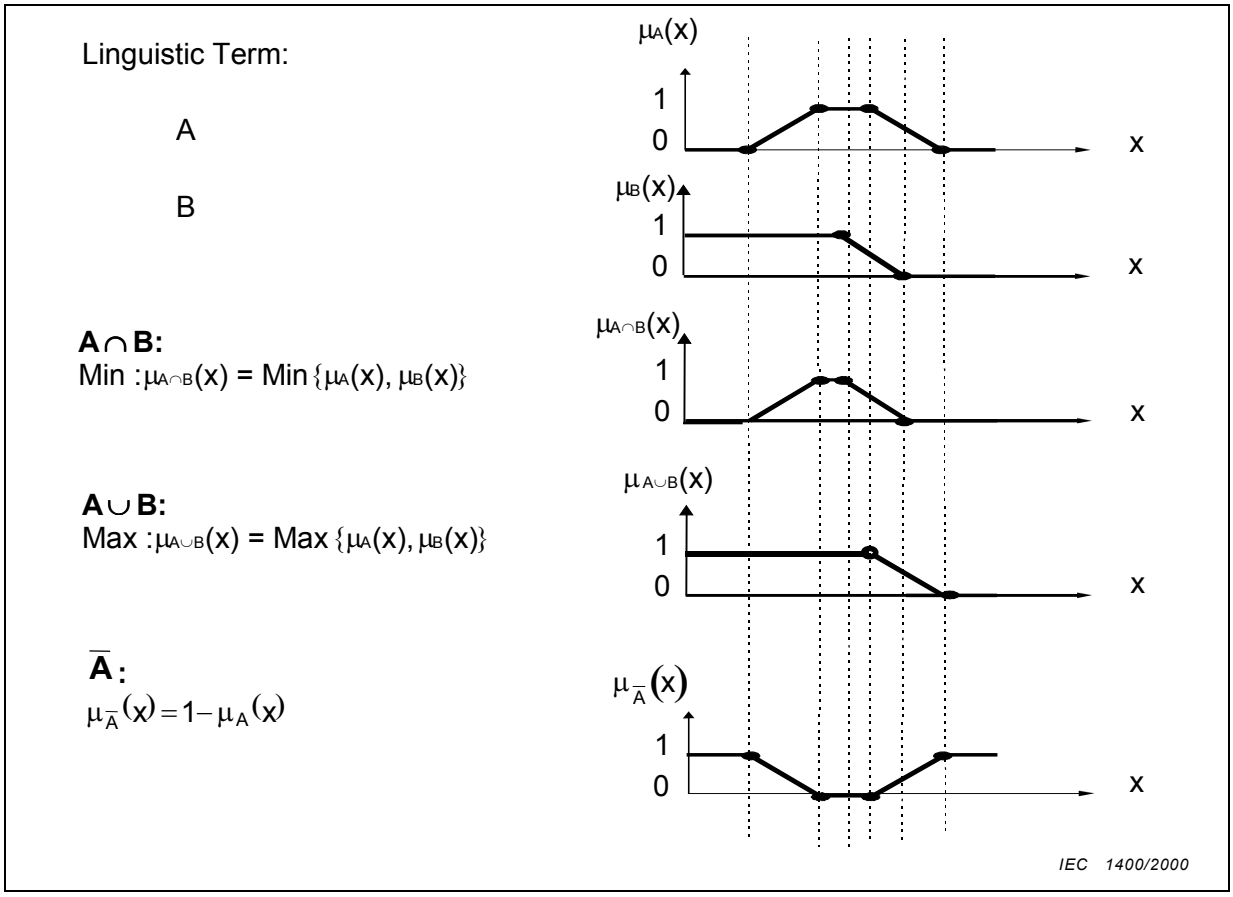


Figure A.4 – Algorithms for implementing operations between two membership functions

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.

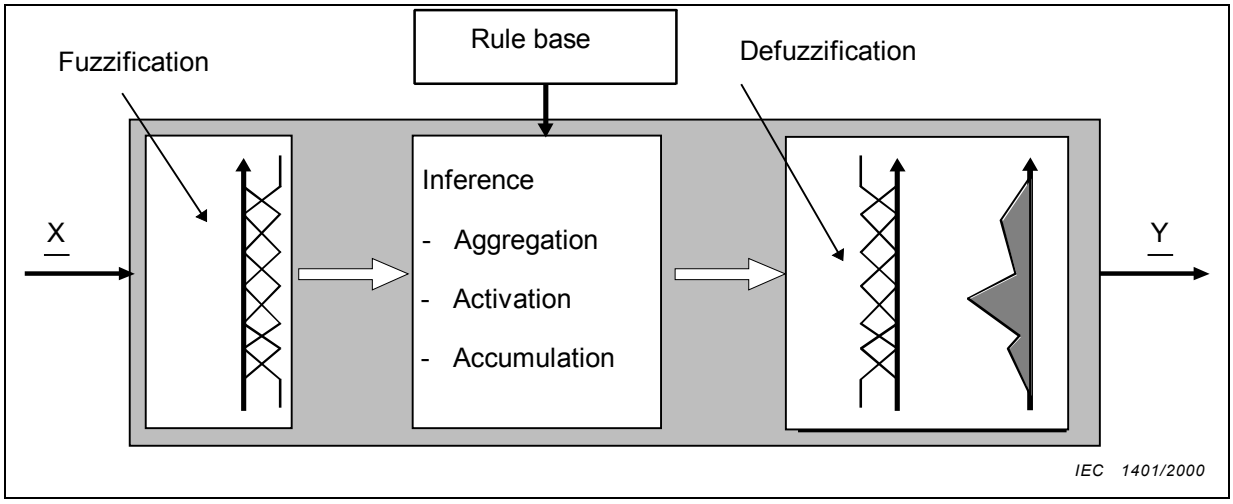


Figure A.5 – Structure and functional elements of fuzzy control

The functional elements of *fuzzy control*, mentioned above and represented in figure A.5, 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.6 shows an example of *fuzzification.*

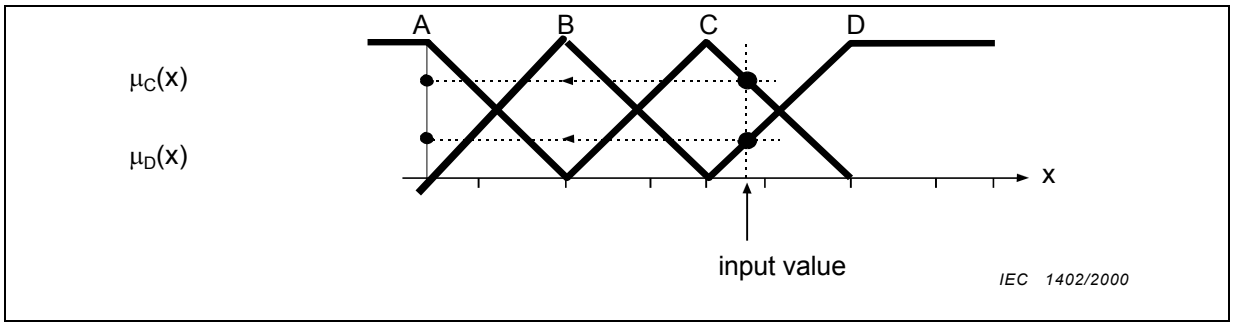


Figure A.6 – The principle of fuzzification (as an example)

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. Provided that the AND fuzzy operator is MIN and the OR fuzzy operator is MAX, it can easily be shown that a fuzzy rule R_j based on an OR combination of m statements may be represented by m rules, whose statements are only combined by AND. Figure A.7 and figure A.8 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})

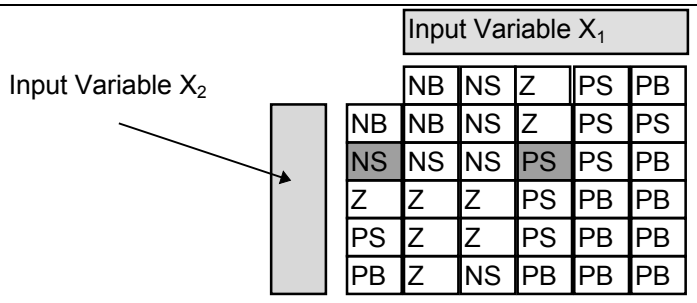
and, for conclusion, there may be more than one, e.g.:

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

IEC 1403/2000

Figure A.7 – Representation of the knowledge base in linguistic 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* may 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.



IEC 1404/2000

Figure A.8 – Matrix representation of two variables

A.2.3 Inference

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 discussed here.

The Inference consists of the three subfunctions *aggregation*, *activation* and *accumulation* shown in figure A.9. See also figure A.10.

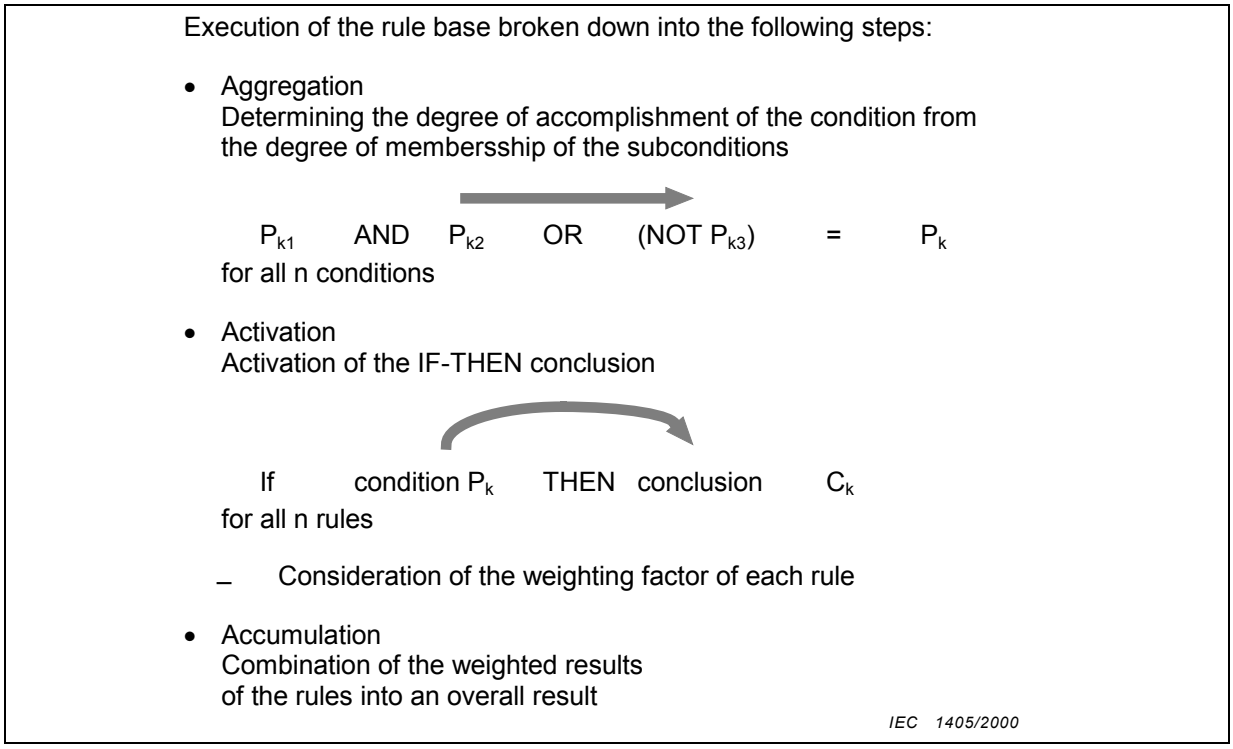


Figure A.9 – Elements of inference

– *Aggregation:*
If the condition consists of a single subcondition, then the validity of the *condition* is identical to that of the subcondition, i.e. the degree of accomplishment of the *condition* corresponds to the subcondition. 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 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 may be implemented by means of multiplication.

$$c_k^* = w_k \times c_k \qquad \text{(A.15)}$$

– *Accumulation:*
The results of the *rules* are combined to obtain an overall result. The *maximum algorithm is* usually used for *accumulation*. Table A.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.1 – Inference steps and commonly used algorithms

Inference step	Operators	Algorithms
Aggregation		
for AND	*Minimum*	$a_k = \text{Min}\{a_{K1}(x1), a_{K2}(x2)\}$
for OR	*Maximum*	$a_k = \text{Max}\{a_{K1}(x1), a_{K2}(x2)\}$
Activation		
conversion of the IF-THEN-conclusion		
	Minimum	$c_k' = \text{Min}\{a_K, \mu_{\alpha}(u)\}$
weighting factor of each rule		
	Multiplication	$c_k = \text{Mult}\{\omega_K, c_K'\} = \omega_K \times c_K'$
Accumulation	*Maximum*	$\mu = \text{MAX}\{c_i(u)\}$

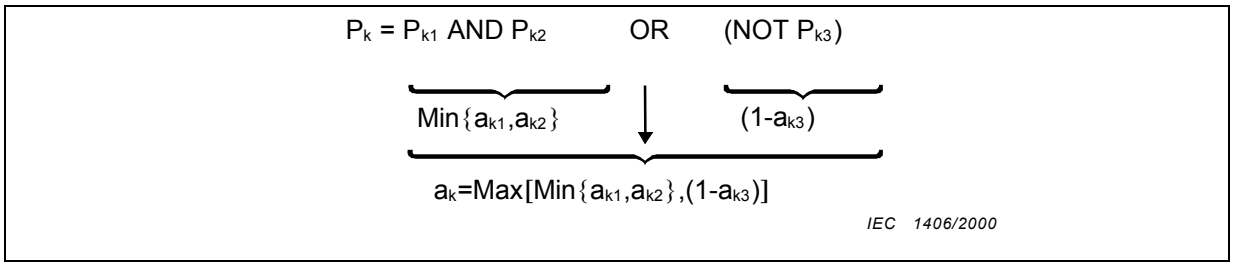


Figure A.10a – An example showing the principles of aggregation

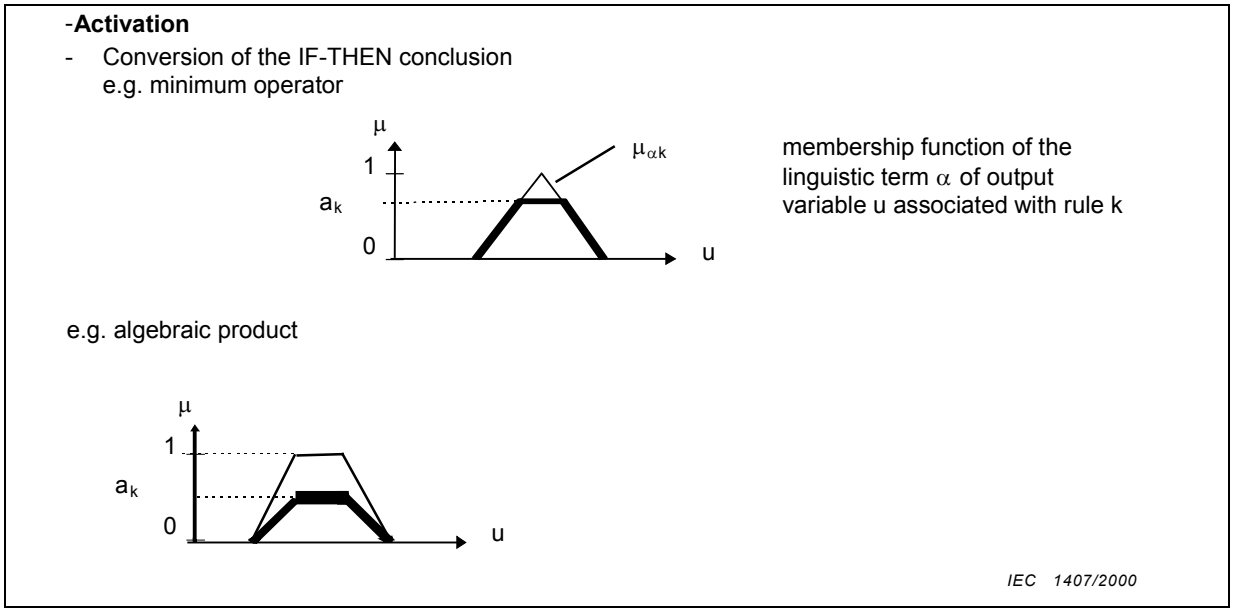


Figure A.10b – The principles of activation (as an example)

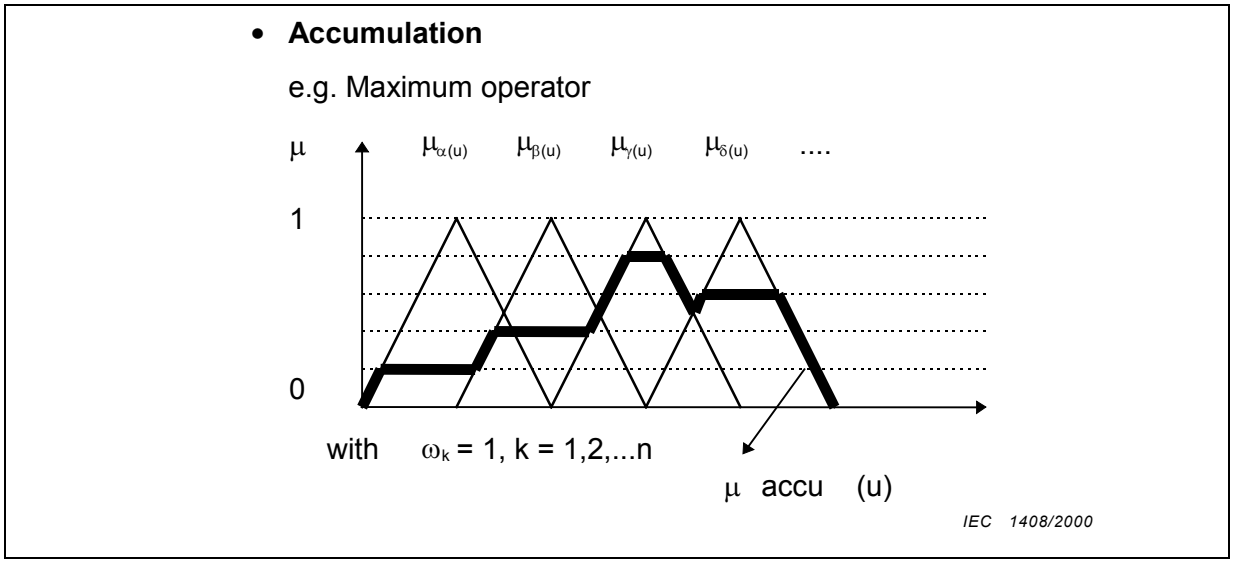


Figure A.10c – The principles of accumulation (as an example)

A.2.4 **Defuzzification** (see figure A.11)

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*.

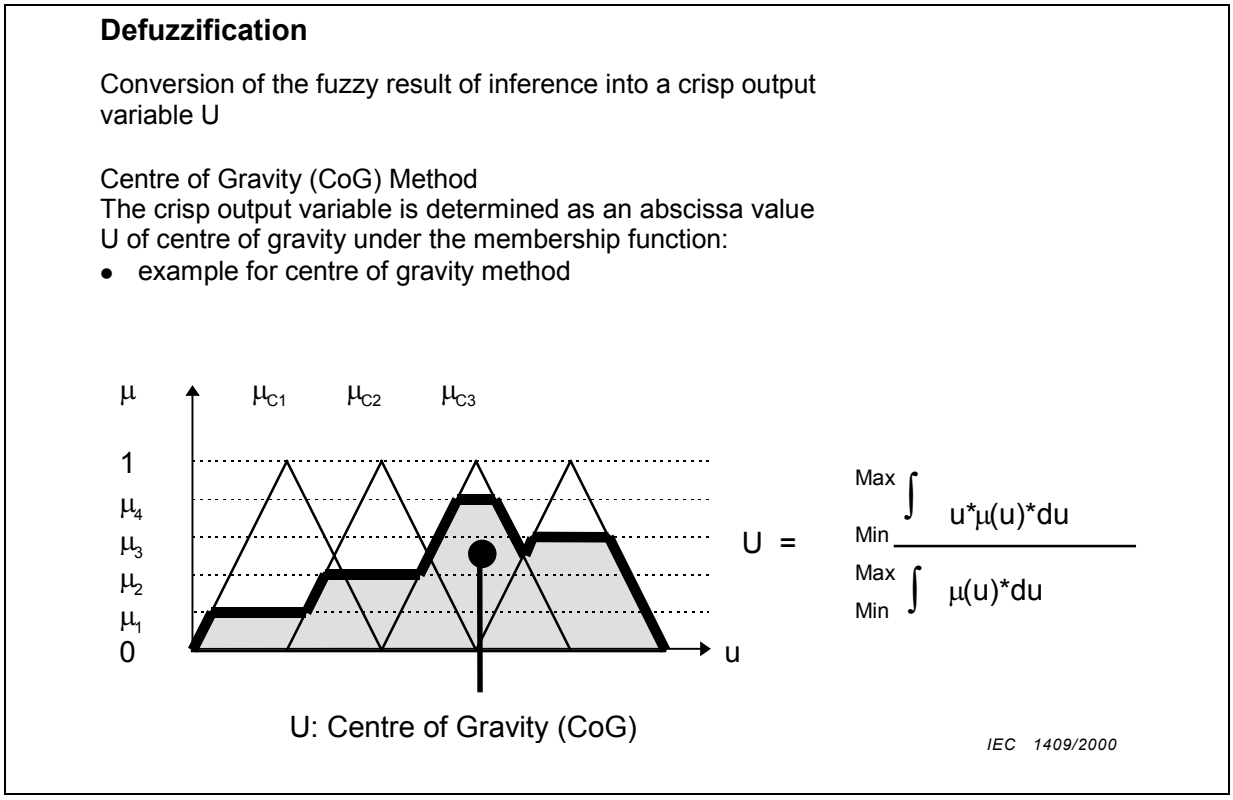


Figure A.11a – Methods of defuzzification

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.

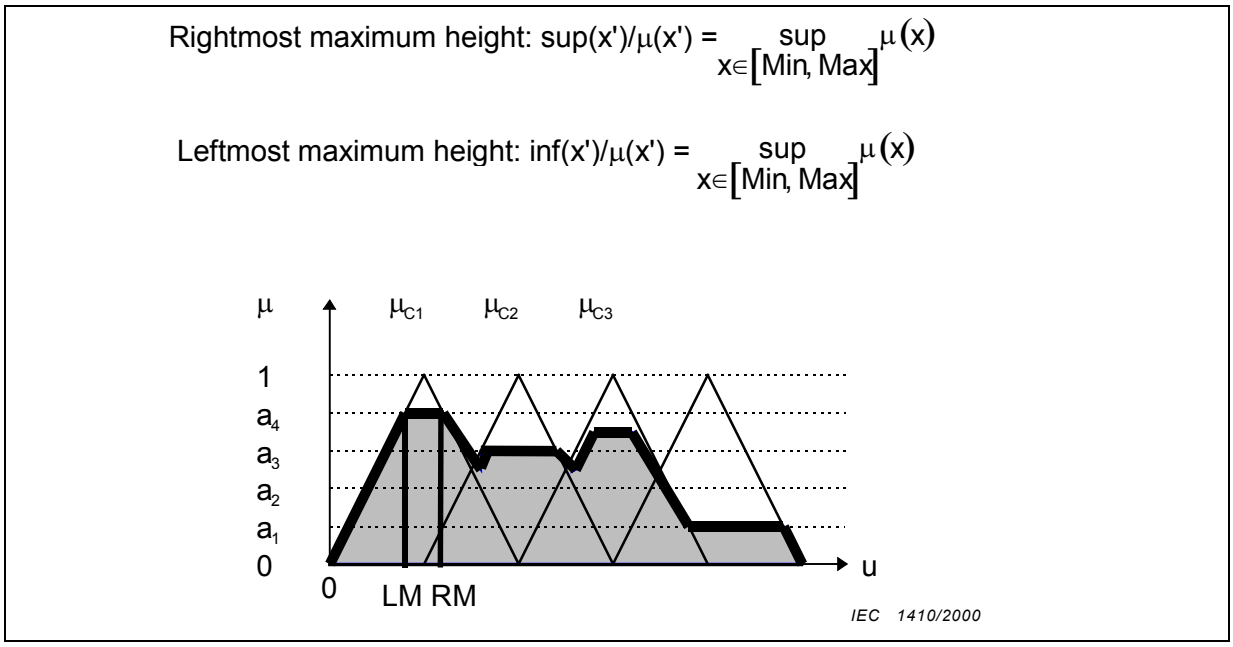


Figure A.11b – Difference between Left Most Maximum and Right Most Maximum

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 equivalent to Bisector of Area.

NOTE 3 CoA is not applicable if singletons are used.

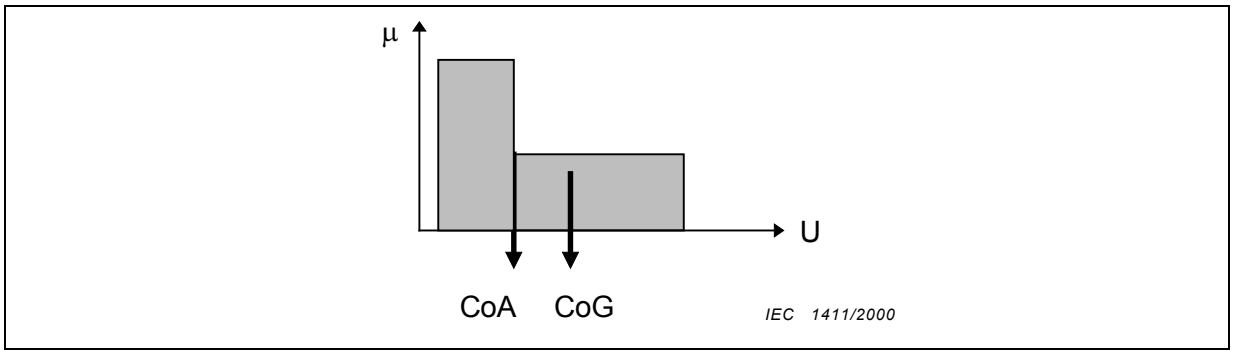


Figure A.11c – Difference between Centre of Area and Centre of Gravity

NOTE Formulae of the Defuzzification methods CoA and CoG are given in table 2.

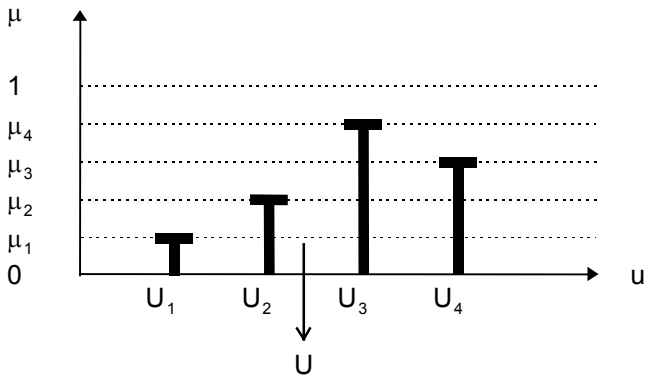
Defuzzification

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

- Centre of Gravity Method for Singleton (COGS)

$$U = \frac{\sum_{i=1}^{p} [U_i * \mu_i]}{\sum_{i=1}^{p} [\mu_i]}$$

where μ are the results of accumulation



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

$$U = \frac{[U_1*\mu_1 + U_2*\mu_2 + U_3*\mu_4 + U_4*\mu_3]}{[\mu_1 + \mu_2 + \mu_3 + \mu_4]}$$

IEC 1412/2000

Figure A.11d – 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.12 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 error, which likewise has to be formed outside fuzzy control. Conversely, the output variables of fuzzy control may be passed to operators for processing of the correcting variables, for example an integration element for speed algorithms, or distributed among different control elements.

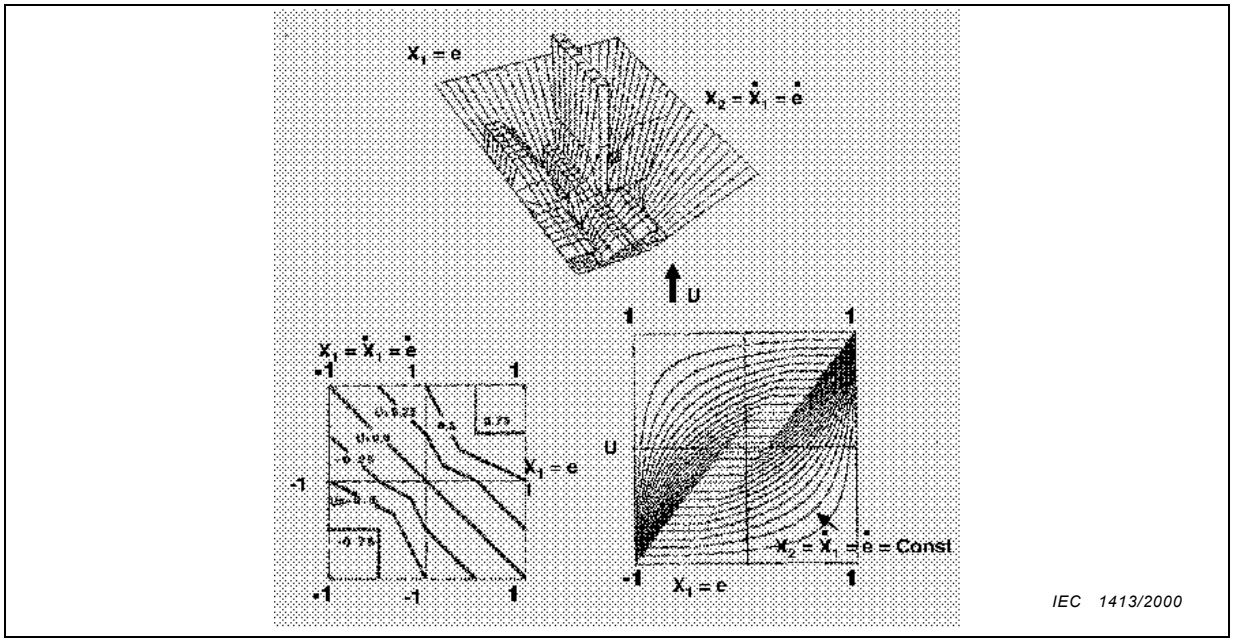


Figure A.12 – Examples of fuzzy control characteristic curves

The fundamental structure of a fuzzy-based controller is shown in figure A.13a, while figure A.13b gives an example: the control error (e) is formed from the difference between the setpoint and the process variable. This difference, as well as its derivative with respect to time, is transmitted to fuzzy control as two 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 PI.

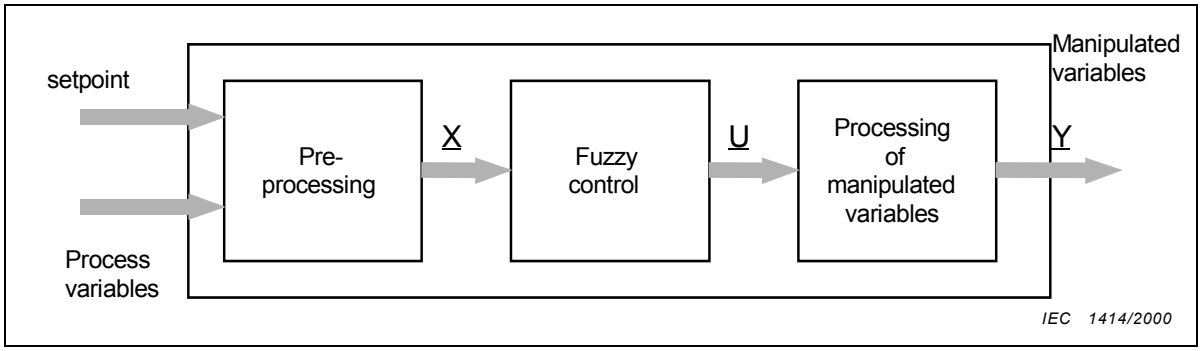


Figure A.13a – Fuzzy-based controller: Fundamental structure

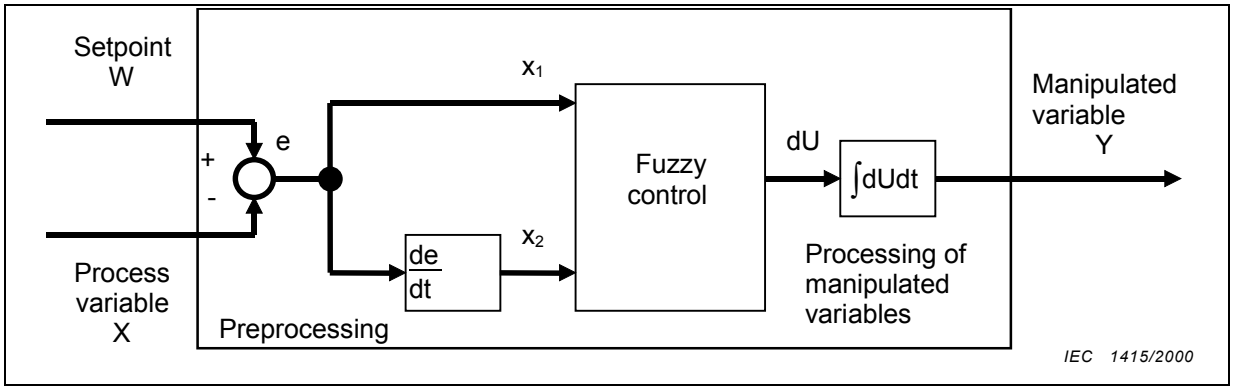


Figure A.13b – Example of a Fuzzy-based controller

Annex B
(informative)

Examples

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 may be used.

B.1 Pre-control

The fuzzy controller supplements the conventional closed loop controller by a corrective signal for the manipulated value (see figure B.1).

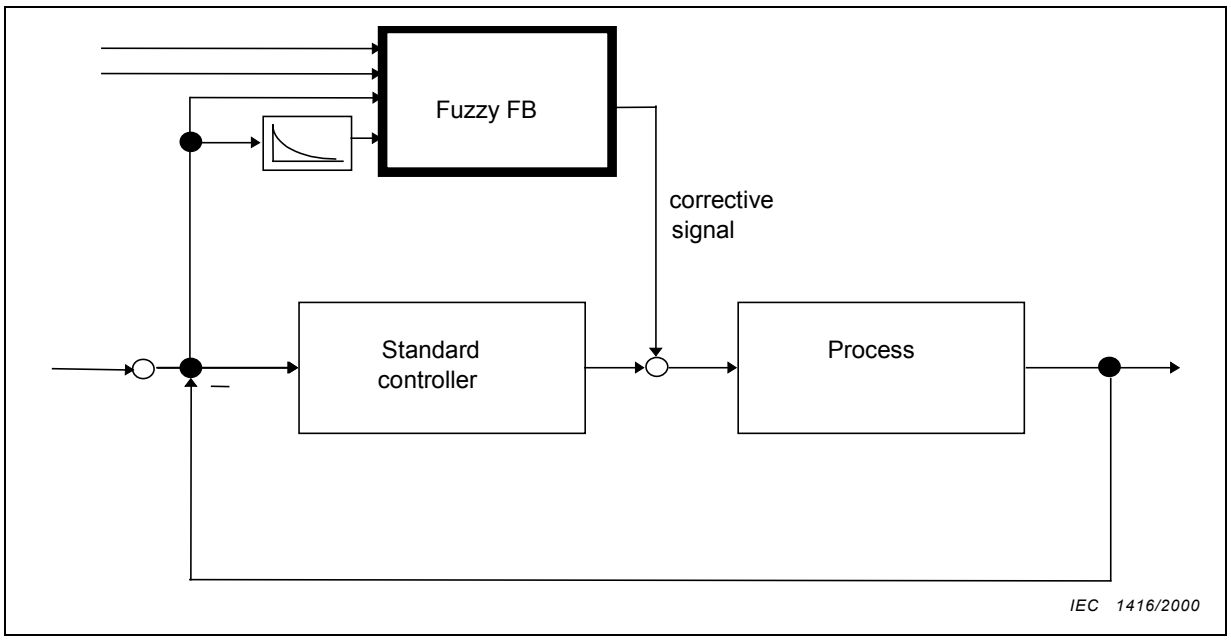


Figure B.1 – Example of a pre-control

B.2 Parameter adaptation of conventional PID controllers

The fuzzy controller is used to adapt the control parameters of a PID controller (see figure B.2).

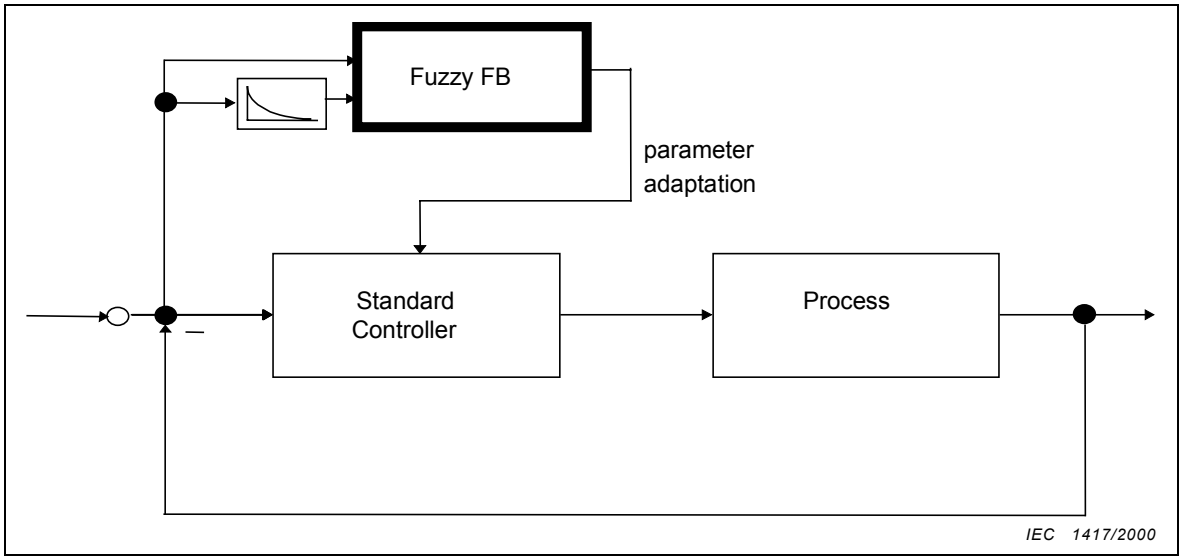


Figure B.2 – Example of a parameter adaptation

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 (see figure B.3).

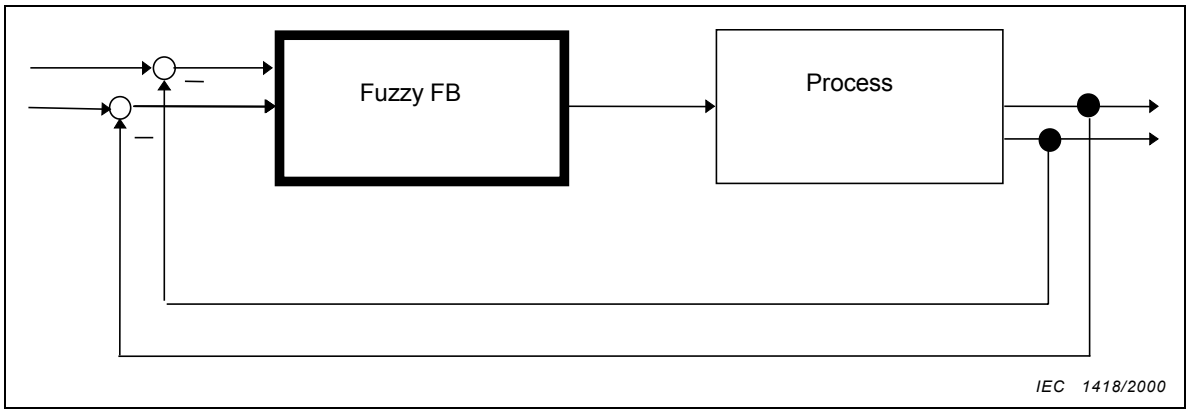


Figure B.3– Example of a direct fuzzy control

Annex C
(informative)

Industrial example – Container crane

Container cranes are used to load and unload containers to and from ships in most harbours. They pick up single containers with 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.

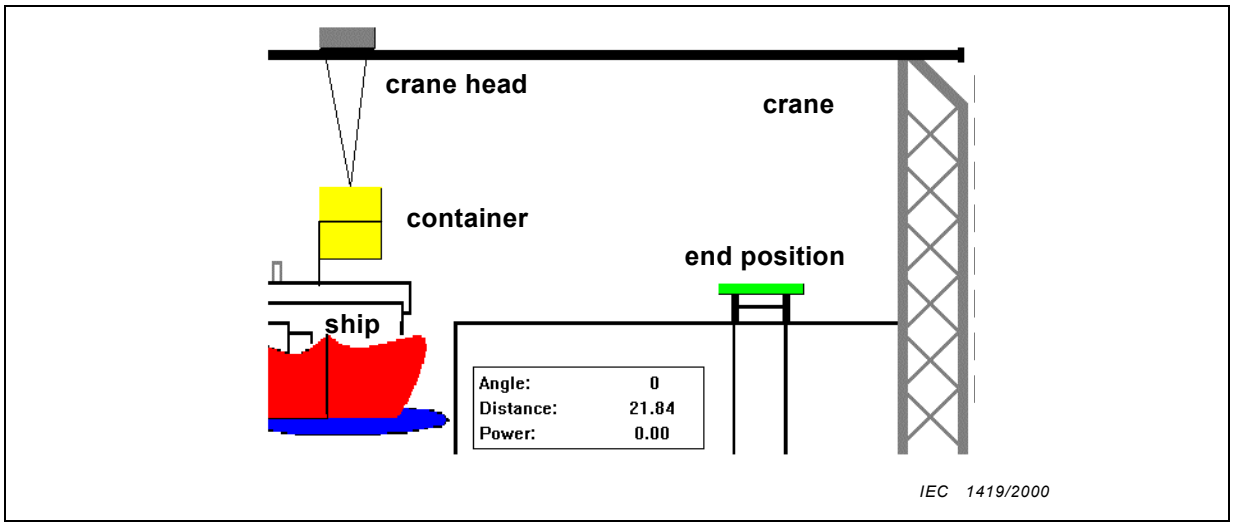


Figure C.1 – Industrial example – Container crane

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

a) Start with medium power.

b) 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.

c) If you are closer to the target, reduce power so the container gets a little ahead of the crane head.

d) When the container is close to target position, use +/– medium power, depending upon sway direction.

e) 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 (see figures C.2, C.3 and C.4) are divided into five 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.

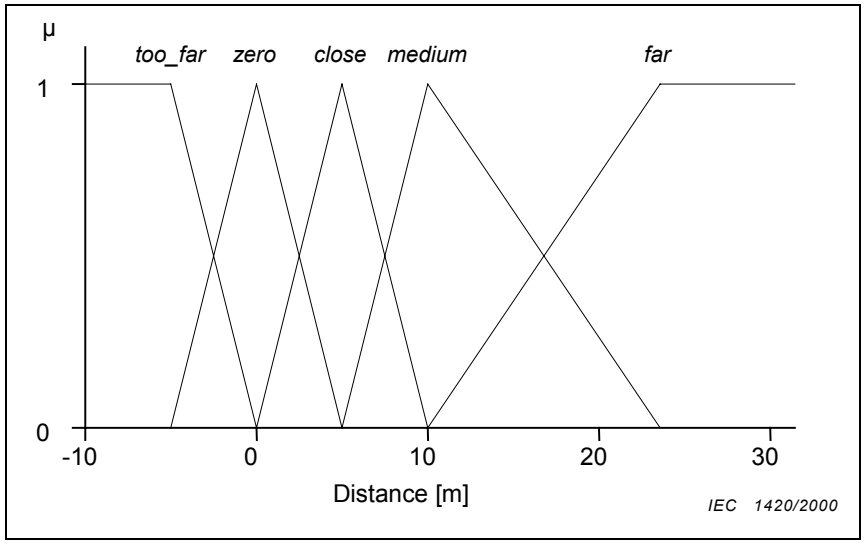


Figure C.2 – Linguistic variable "Distance" between crane head and target position

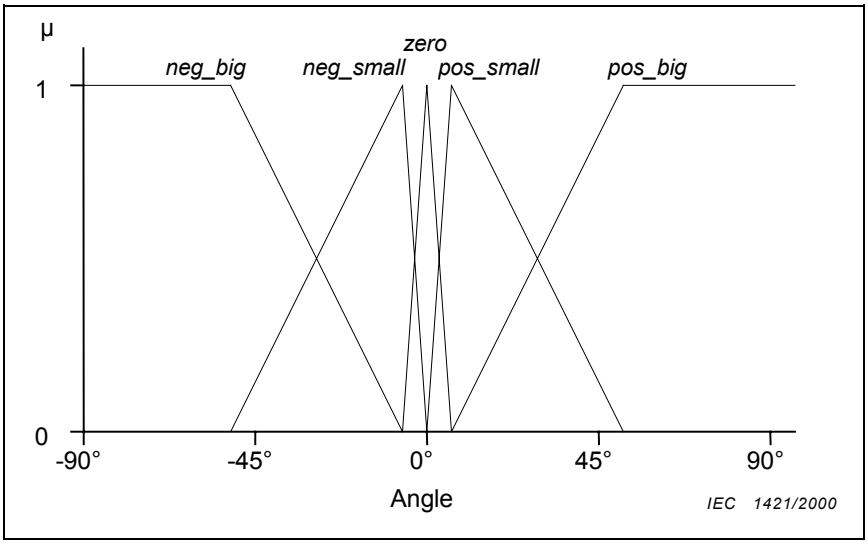


Figure C.3 – Linguistic variable "Angle" of the container to the crane head

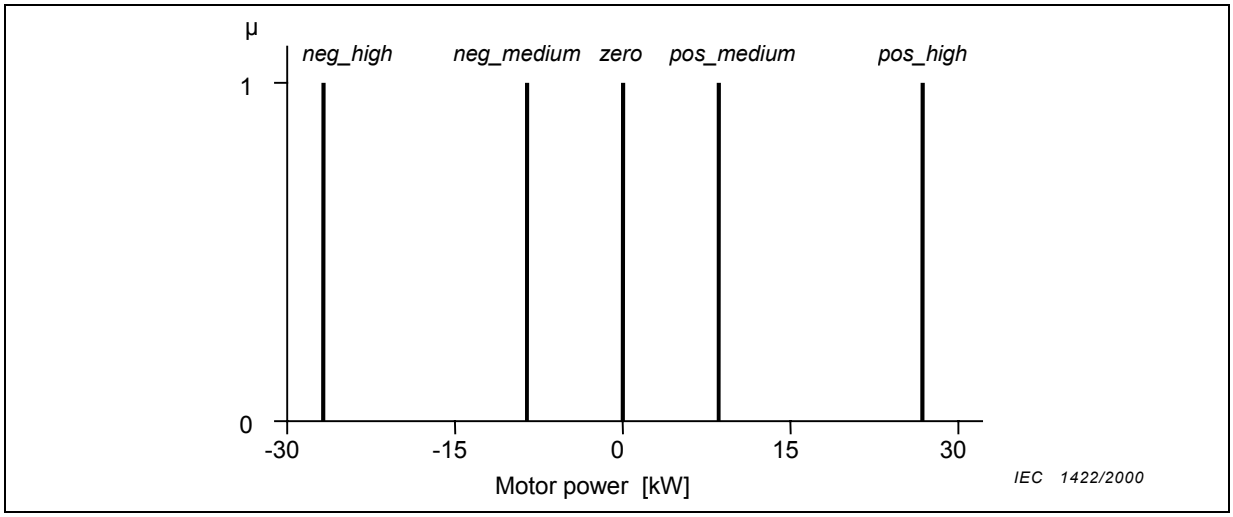


Figure C.4 – Linguistic variable "Power"

Using these linguistic terms to describe the current condition of the crane, the five rules of thumb may be translated into a rule base. Figure C.5 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. Note that some of the defined membership functions are not used here and the system should be checked for undefined outputs like Distance: too far; Angle: pos_big.

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

IEC 1423/2000

Figure C.5 – Rule base

Table C.1 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 m and the angle of the container is +4°. For illustration, a subset of three rules will be assumed. Figures C.6 and C.7 show how the *fuzzification* is computed for this case.

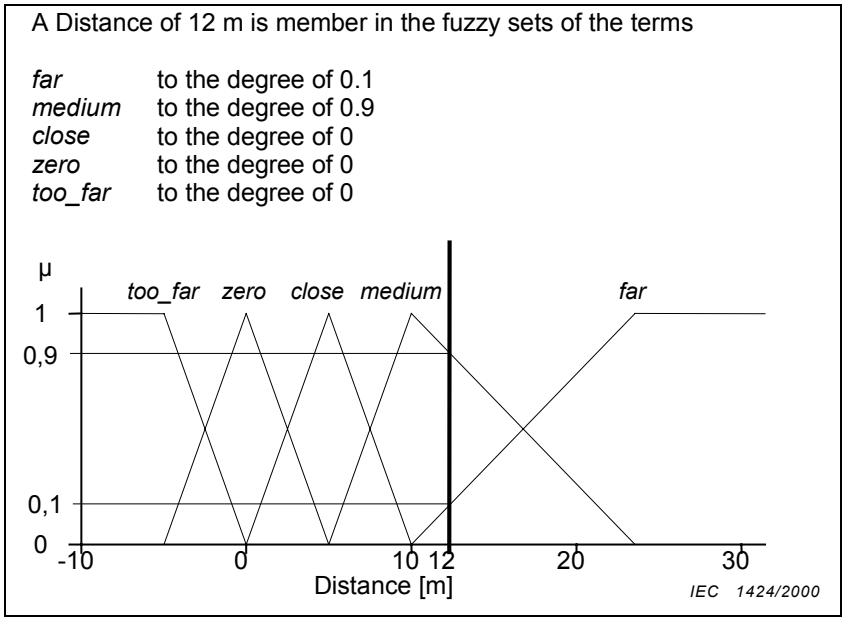


Figure C.6 – Fuzzification of the linguistic variable "Distance"

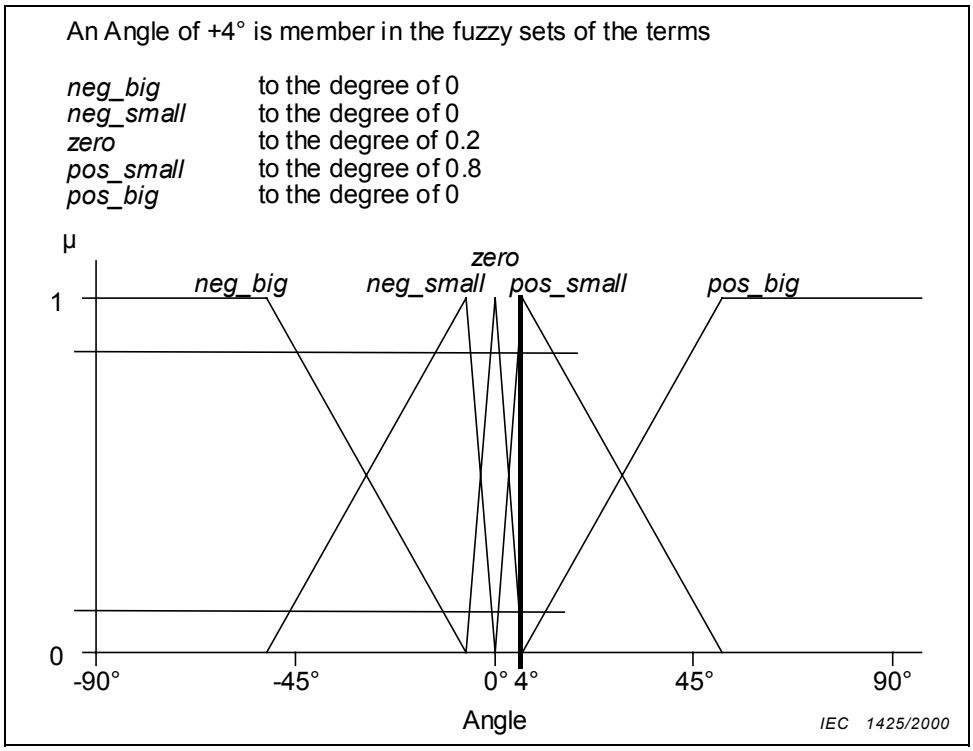


Figure C.7 – Fuzzification of the linguistic variable "Angle"

The distance of 12 m is translated into the linguistic variable value of {0.1, 0.9, 0, 0, 0} which may 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 may be interpreted as "positive small, somewhat zero".

– Inference
  Now that all input variables have been converted to linguistic variable values, the fuzzy *inference* step may identify the rules that apply to the current situation and may compute the values of the output linguistic variable. Figure C.8 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

IEC 1426/2000

Figure C.8 – Subset of three rules

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

– Aggregation (see figure C.10)
  Determining the degree of conformance of the premise from the degree of membership of the elementary statements

Rule 1: Distance = medium	AND	Angle = pos_small	=	P_1
Rule 2: Distance = medium	AND	Angle = zero	=	P_2
Rule 3: Distance = far	AND	Angle = zero	=	P_3

for all premises

IEC 1427/2000

Figure C.9 – Elements of aggregation

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

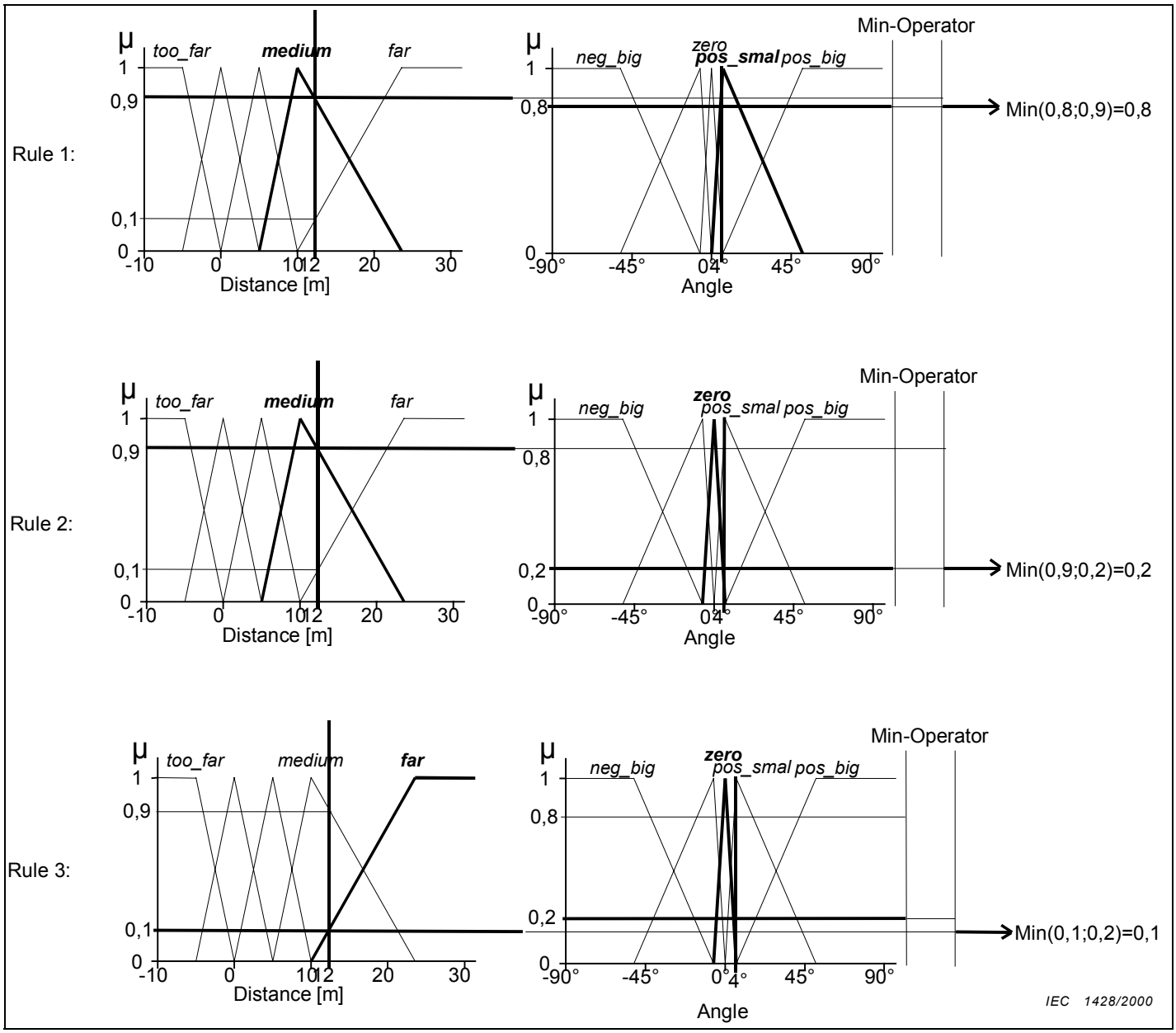


Figure C.10 – Principles of aggregation

– Activation (see figure C.12)
Conversion of the IF-Then conclusion

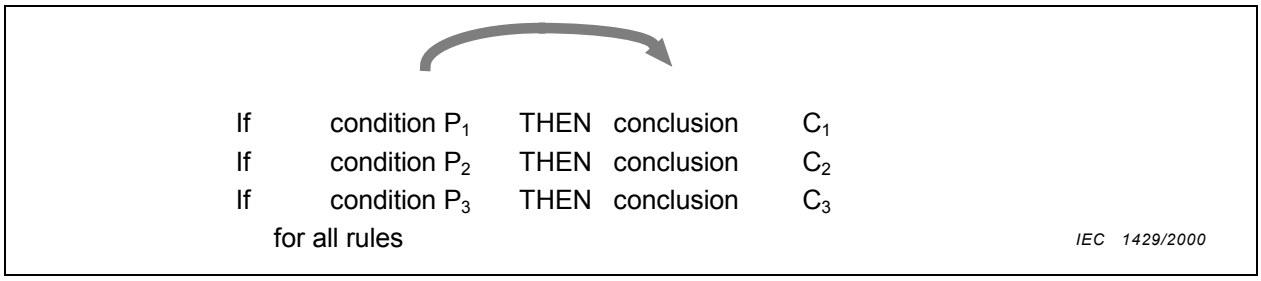


Figure C.11 – Elements of activation

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.

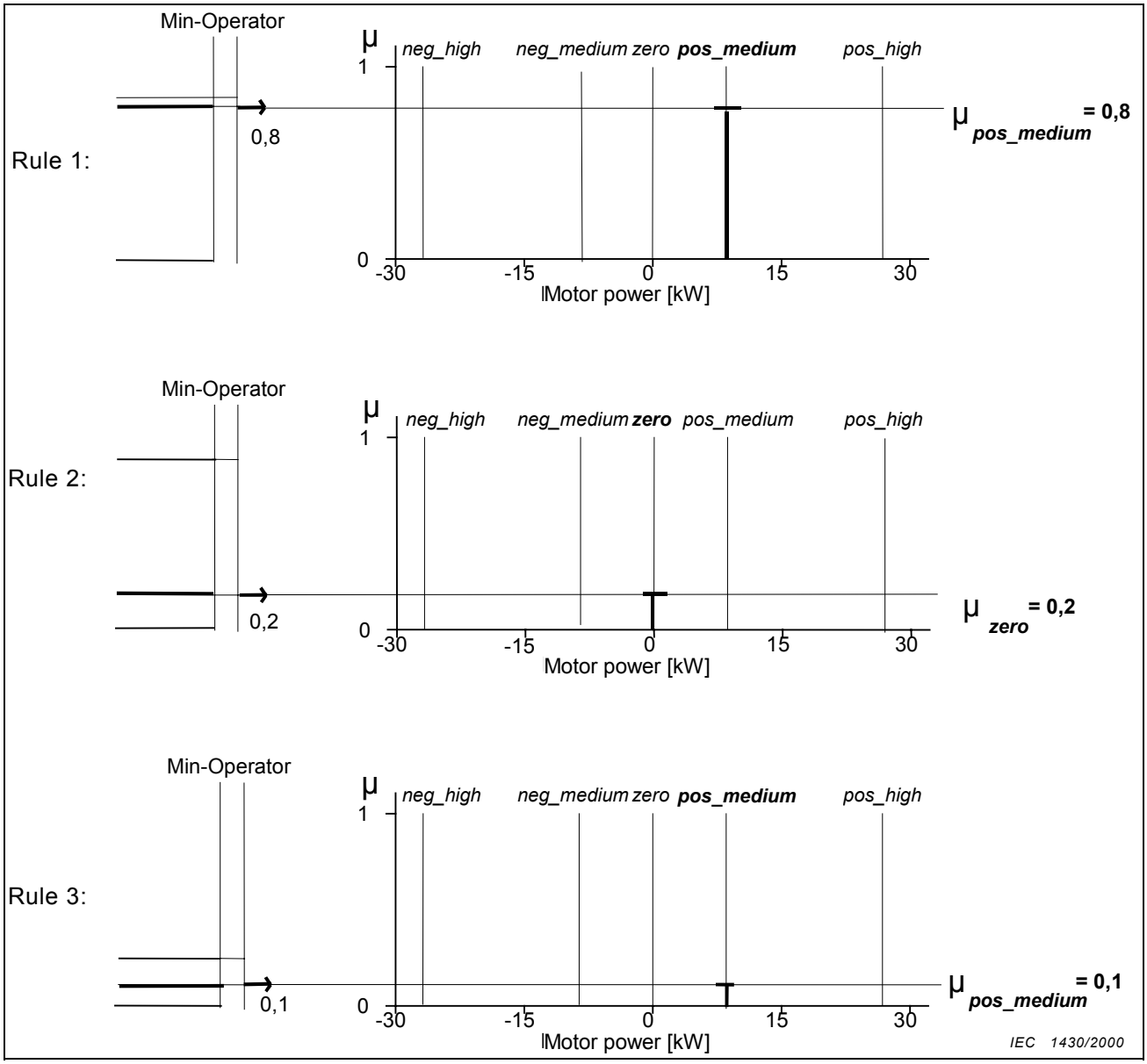


Figure C.12 – Principles of activation

– Accumulation (see figure C.14)
Combination of the weighted results of the rules into an overall result

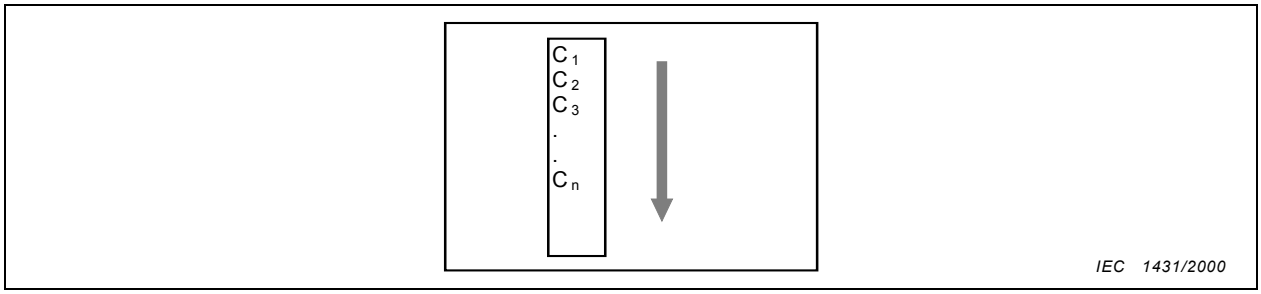


Figure C.13 – Elements of accumulation

The result of the accumulation of rules 1 to 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.

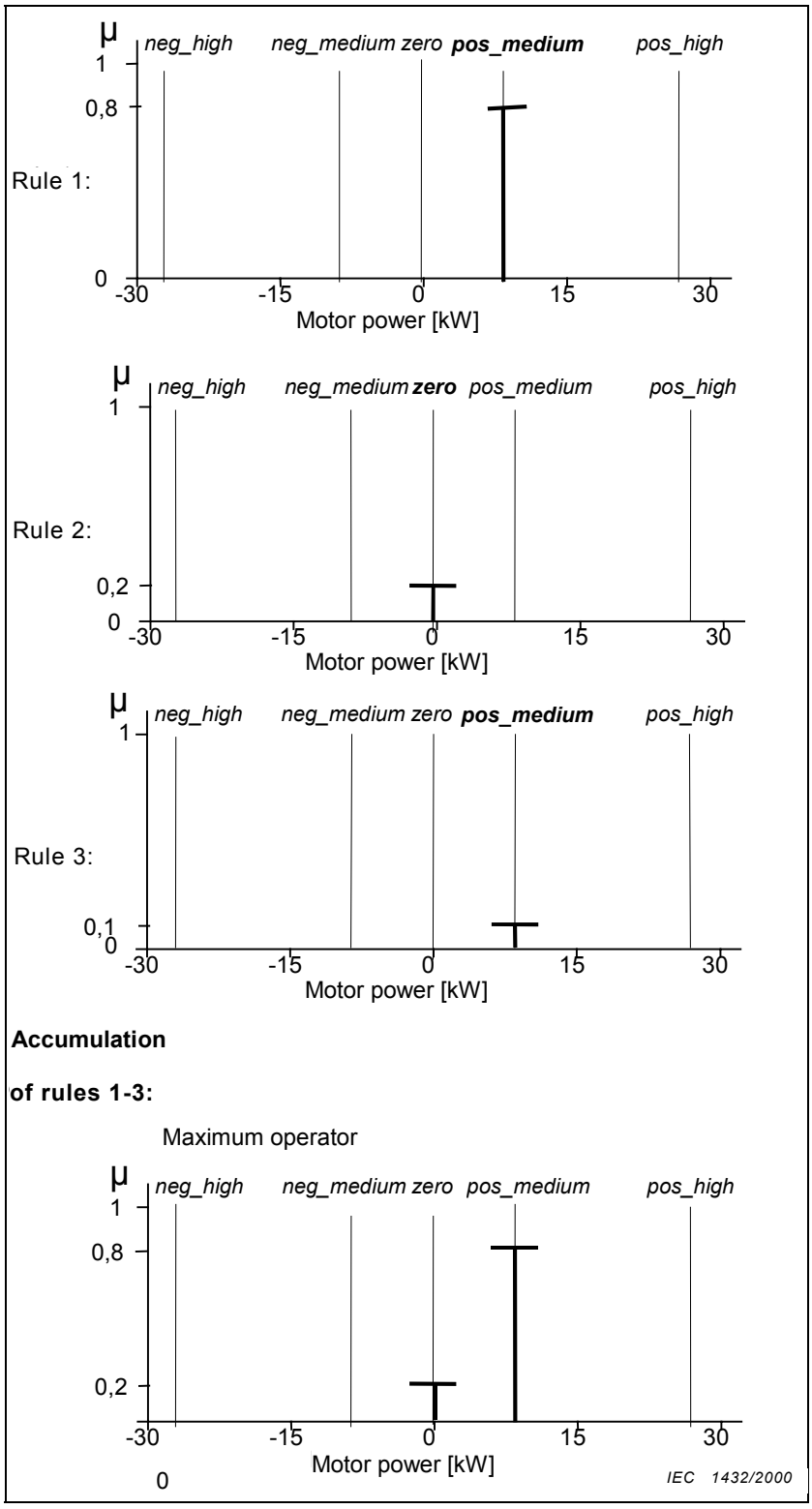


Figure C.14 – Principles of accumulation

– Defuzzification (see figure C.15)

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:

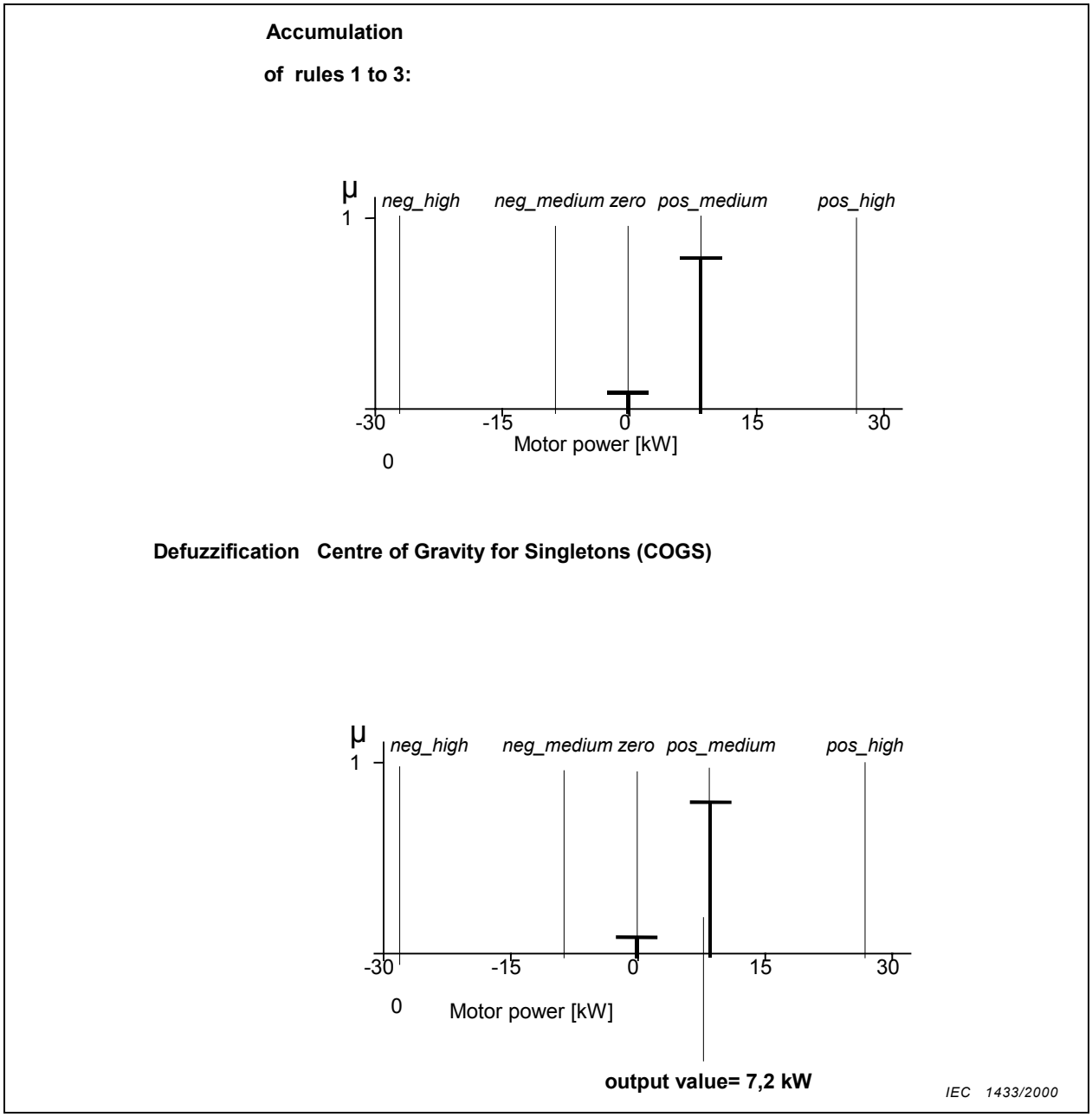


Figure C.15 – Defuzzification

– Implementation of the container crane example in FCL (see figure C.16):

```
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     :=  -9;
        TERM zero           :=   0;
        TERM pos_medium     :=   9;
        TERM pos_high       :=  27;
        METHOD: CoGS;
        DEFAULT:= 0;
END_DEFUZZIFY

RULEBLOCK No1
        AND: MIN;
        ACCU: MAX;
        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
```

IEC 1434/2000

Figure C.16 – Example in SCL

The FCL function block shall be invoked in the same manner as with programs according to IEC 61131-3. The reason for this is that from the external viewpoint, the fuzzy block has its only interface via its variables.

According to IEC 61131-3, 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
(informative)

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 may 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 figures D.1 and D.2.

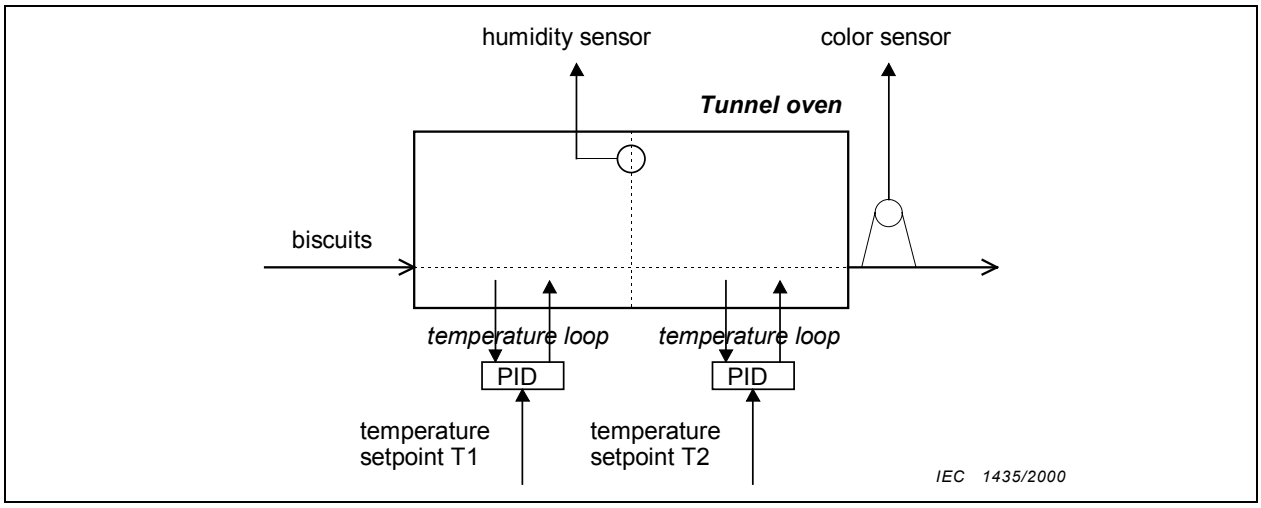


Figure D.1 – Principle of the controlled system

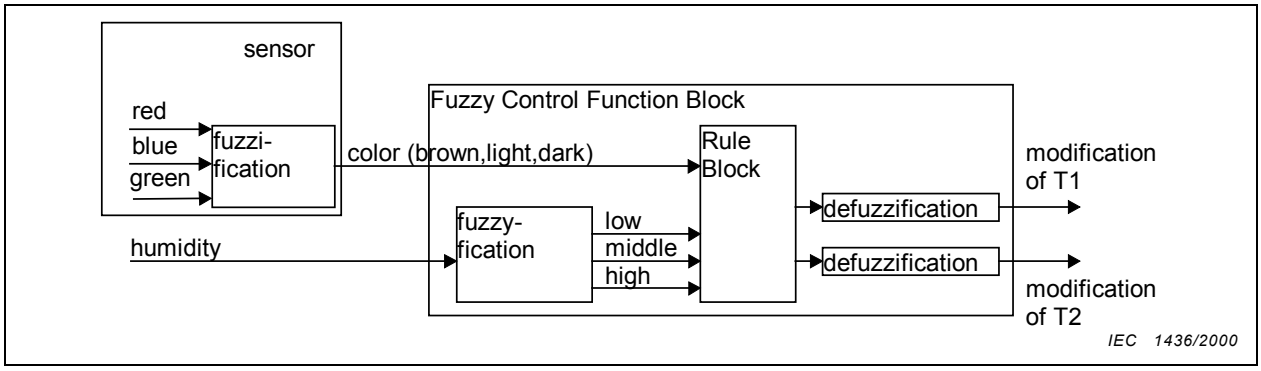


Figure D.2 – Principle of the fuzzy based control of the oven

The rule block contains the following five rules (see figure D.3):

IF humidity IS middle AND color IS 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 color IS light	THEN dT2 IS positive
IF humidity IS middle AND color IS dark	THEN dT2 IS negative

IEC 1437/2000

Figure D.3 – Rule block

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

NOTE Instead of using the three variables light, brown and dark, only one variable of the enumerated data type of color may also be used as shown below. In this example, dT2 is undefined if humidity is high or low.

```
TYPE
STRUCT color_type
        brown: REAL;
        light:   REAL;
        dark:    REAL;
END_STRUCT
END_TYPE

FUNCTION_BLOCK oven_control
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;
        METHOD: CoGS;
        DEFAULT:= 0;
END_DEFUZZIFY

DEFUZZIFY dT2
        TERM negative       := –3:
        TERM zero           :=  0;
        TERM positive       :=  3;
        METHOD: CoGS;
        DEFAULT:= 0;
END_DEFUZZIFY

RULEBLOCK inference

        AND: MIN;
        ACCU: MAX;
        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                                                IEC  1438/2000
```

Figure D.4 – Example in FCL

Annex E
(informative)

Symbols, abbreviations and synonyms

Table E.1 – Symbols and abbreviations

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

Table E.2 – Synonyms

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

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

This European Standard incorporates by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this European Standard only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies (including amendments).

NOTE When an international publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

Publication	Year	Title	EN/HD	Year
IEC 60050-351	1998	International Electrotechnical Vocabulary Part 351: Automatic control	-	-
IEC 61131-3	1993	Programmable controllers Part 3: Programming languages	EN 61131-3	1993

BS EN 61131-7:2001
IEC 61131-7:2000

BSI — British Standards Institution

BSI is the independent national body responsible for preparing British Standards. It presents the UK view on standards in Europe and at the international level. It is incorporated by Royal Charter.

Revisions

British Standards are updated by amendment or revision. Users of British Standards should make sure that they possess the latest amendments or editions.

It is the constant aim of BSI to improve the quality of our products and services. We would be grateful if anyone finding an inaccuracy or ambiguity while using this British Standard would inform the Secretary of the technical committee responsible, the identity of which can be found on the inside front cover. Tel: 020 8996 9000. Fax: 020 8996 7400.

BSI offers members an individual updating service called PLUS which ensures that subscribers automatically receive the latest editions of standards.

Buying standards

Orders for all BSI, international and foreign standards publications should be addressed to Customer Services. Tel: 020 8996 9001. Fax: 020 8996 7001. Standards are also available from the BSI website at http://www.bsi-global.com.

In response to orders for international standards, it is BSI policy to supply the BSI implementation of those that have been published as British Standards, unless otherwise requested.

Information on standards

BSI provides a wide range of information on national, European and international standards through its Library and its Technical Help to Exporters Service. Various BSI electronic information services are also available which give details on all its products and services. Contact the Information Centre. Tel: 020 8996 7111. Fax: 020 8996 7048.

Subscribing members of BSI are kept up to date with standards developments and receive substantial discounts on the purchase price of standards. For details of these and other benefits contact Membership Administration. Tel: 020 8996 7002. Fax: 020 8996 7001. Further information about BSI is available on the BSI website at http://www.bsi-global.com.

Copyright

BSI
389 Chiswick High Road
London
W4 4AL