LFLC 2000 Help

University of Ostrava
Institute for Research and Applications of Fuzzy Modeling
30. dubio 22, Ostrava, Czech Republic
tel. +420-59-6160218, fax +420-59-6120 478
email: viktor.pavliska@osu.cz
web: http://irafm.osu.cz/irafm

Description of LFLC 2000

LFLC 2000 (Linguistic Fuzzy Logic Controller) is a specialized software, which is based on fuzzy set theory and fuzzy logic to enable to deduce conclusions on the basis of imprecise description of the given situation using the linguistically formulated *fuzzy IF-THEN rules*. It is specific for this software that it enables to work with genuine linguistically defined rules forming a linguistic description of the given process, decision or classification situation. The user thus may work only with expressions of natural language without necessity to think how they are implemented. Thus, computer behaves as if "partner" understanding the language of human user.

By a fuzzy set A in the universe U (in symbols $A \subseteq U$) we understand a special function

$$A : U \longrightarrow [0, 1]$$

where the value $A(x) \in [0, 1]$ is a membership degree of the element $x \in U$ in the fuzzy set A. This can also be understood as the *truth degree* stating, how much it is *true* that the element $x \in U$ *belongs* to the fuzzy set A.

Literature helpful in understanding this software is:

References

[1] Novák, V.: **Fuzzy množiny a jejich aplikace.** SNTL, Praha 1986 and 1990. (Fuzzy Sets and Their Applications; in Czech)

[2] Novák, V.: **Fuzzy Sets and Their Applications**. Adam Hilger, Bristol 1989.

[3] Novák, V., Perfilieva I. and J. Močkoř: **Mathematical Principles of Fuzzy Logic**. Kluwer, Boston/Dordrecht 1999.

[4] Novák, V.: **Základy fuzzy modelování.** BEN, Praha 2000. (Foundations of fuzzy modeling; in Czech)

[5] Hájek, P.: **Metamathematics of Fuzzy Logic.** Kluwer, Dordrecht 1998.

[6] Klir, G.J. and Bo Yuan: **Fuzzy Sets and Fuzzy Logic. Theory and Applicatations.** Prentice Hall, Englewood Cliffs 1995.

Fuzzy IF-THEN rules

The main task of LFLC 2000 is to work (design and test) behavior of linguistic descriptions. These are sets of fuzzy IF-THEN rules of the form either

$$\mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_1\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_n\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}$$

(*linguistic rule*), or

$$\mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_1\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_n\ \mathsf{THEN}\ Y\ \mathsf{is}\ c$$

(*Takagi-Sugeno singleton rule*), where $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and $\mathcal{B}$ are certain predicates characterizing the variables $X_1, \ldots, X_n$ and Y. They are often specified linguistically. In case of Takagi-Sugeno singleton rule (TS singleton for short), c is a real number. The software enables to work with specific kind of linguistic expressions, or the user may specify his own ones.

Example

An example fuzzy IF-THEN rule is

IF *the obstacle is near* AND *the speed of the car is high*
THEN *the breaking force is very strong*.

The "obstacle", "speed" and "breaking force" are variables while "near", "high" and "very strong" are expressions characterizing vaguely the magnitude of the variable.

The part before THEN is called the *antecedent* and the part after it the *succedent*. The variables $X_1, \ldots, X_n$ are called *input*, or *independent* variables. The variable Y is called *output*, or *dependent* variable.

The fuzzy IF-THEN rules are usually put together to form the *linguistic description*

$$\begin{aligned}
\mathcal{R}_1 :=&\ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_{11}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_{1n}\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_1\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
\mathcal{R}_m :=&\ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_{m1}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_{mn}\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_m
\end{aligned}$$

Besides sophisticated tools for the design of the linguistic descriptions, the software also provides several kinds of approximate reasoning mechanisms, using which a conclusion on the basis of the linguistic description can be obtained. This depends on the user's requirements and specification in relation with the following fundamental approaches to the *explication of IF-THEN rules*.

(i) *Logical explication* – the rules are treated as linguistically characterized logical implications.

(ii) *Functional explication* – the rules are treated as parts of description of some functional dependence in a model.

Logical explication

This is based on the assumption that the fuzzy IF-THEN rules are genuine linguistically characterized logical implications. Let us remark that logical explication of fuzzy IF-THEN rules together with fuzzy logic deduction described further is specific for LFLC 2000 and (according to the authors' knowledge) has no counterpart in other software packages for fuzzy logic.

Functional explication

This is a more traditional possibility. The linguistic description is in this case used for characterization of functional dependency, the existence of which is assumed, but which is not precisely known. The result of inference on the basis of the specified linguistic description is some function approximating the latter dependence.

The linguistic description is as a whole assigned one of the normal forms:
disjunctive normal form

$$\bigvee_{j=1}^{m}(A_{j1}(x_1) \wedge \cdots \wedge A_{jn}(x_n) \wedge B_j(y)),$$

or *conjunctive normal form*

$$\bigwedge_{j=1}^{m}((A_{j1}(x_1) \wedge \cdots \wedge A_{jn}(x_n)) \Rightarrow B_j(y)).$$

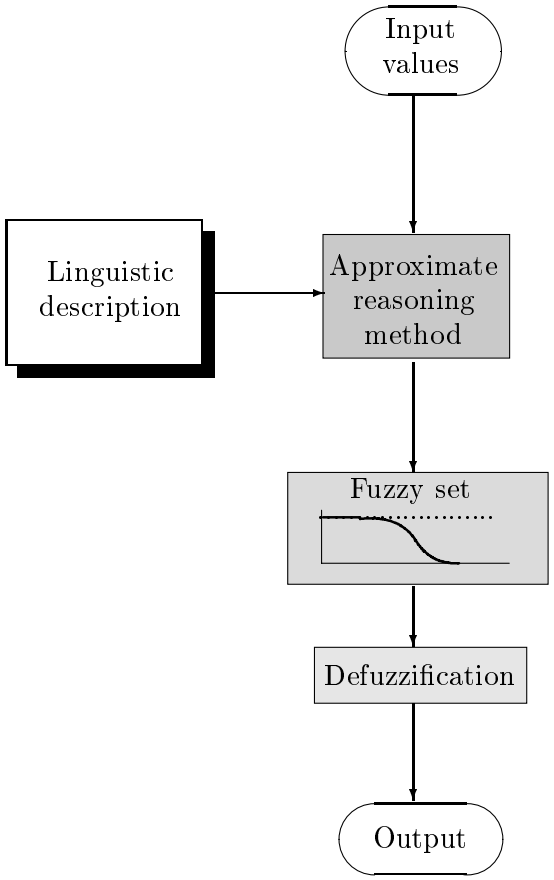
Linguistic expressions

The expressions $\mathcal{A}_1, \ldots, \mathcal{A}_n$, $\mathcal{B}$ inside of the fuzzy IF-THEN rules can be either the, so called, evaluating linguistic expressions or simply labels of some concepts. In the first case, the leading idea is to make computer as if human partner who "understands" descriptions formulated in natural language and whose behavior corresponds to their meaning. The user deals with them exactly as he/she is accustomed to use natural language assuming that the computer "understands" them and thus, without specification of the corresponding fuzzy sets (of course, these can be modified if one wishes to do it).

In the second case, the expressions $\mathcal{A}$ and $\mathcal{B}$ are not explicitly formulated in natural language and are only labels (set by the user) of fuzzy sets representing the meaning of certain concepts in the user's mind. The fuzzy sets can be modified to obtain as close approximation of the user's idea as possible.

Approximate reasoning

The linguistic description is designed to enable approximate reasoning, i.e. a procedure on the basis of which given some information, we derive a conclusion. The general approximate reasoning scheme is depicted on the following figure:



The input values are specific values of the input variables. They are further used by the chosen approximate reasoning method together with the specified linguistic description to produce the output fuzzy set. In most cases, we need one specific value, which should be derived from the output fuzzy set. This is realized using some defuzzification method.

Go to Main Menu

Work with Linguistic Descriptions

Open or New linguistic description

Click on either of:

File → *New* or press **Ctrl N**

File → *Open* or press **Ctrl O**

and choose the file (its extension is ".rb"). If you choose *New*, dialog window appears and you choose either linguistic or TS-Singleton type of linguistic description. This opens a window, in which global information about the concrete linguistic description is specified. The window has the following Tab-pages:

- General settings of the linguistic description
- Input variables
- Output variables
- Rules
- Input/Output

General settings of the linguistic description

In this Tab-page, the following main characteristics of the linguistic description are set.

Name of unit

The name of the linguistic description is specified. This name is identical with the name of the file in which the description is stored.

Type of unit

Either Linguistic or TS-Singleton type is displayed. It is not possible to change type of linguistic descriptions, because different type of expressions is included in the succedent side of rules.

Type of inference

Specification of type of says, how fuzzy IF-THEN rules should be explicated and what kind of approximate reasoning method should used. The following options are available:

- For linguistic type of unit:

  (a) *Logical deduction*

  In this case, the linguistic description is translated into a set of logical implications of the form
  $\mathbf{A}_{1,X_1,\ldots,X_n} \Rightarrow \mathbf{B}_{\mathbf{1}Y}$
  .....
  $\mathbf{A}_{\mathbf{m}X_1,\ldots,X_n} \Rightarrow \mathbf{B}_{m,Y}$
  where $\mathbf{A}_{j,X_1,\ldots,X_n}$, $\mathbf{B}_{j,Y}$ are formal fuzzy representations (we can see them as special formulas) of the corresponding linguistic expressions.

  If some input values $x_{10},\ldots,x_{n0}$ are given then they are transformed into the *most appropriate* formula, say $\mathbf{A}_{k,X_1,\ldots,X_n}$. This formula takes the role of *perception* of the given value. Then the logical modus ponens

  $$\frac{\mathbf{A}_{k,X_1,\ldots,X_n},\quad \mathbf{A}_{k,X_1,\ldots,X_n} \Rightarrow \mathbf{B}_{m,Y}}{\mathbf{B}_{m,Y}}$$

  is realized, the result of which is the formula $\mathbf{B}_{m,Y}$. This is transformed into a fuzzy set $B_m \underset{\sim}{\subseteq} V$. This is finally defuzzified to obtain the output value y_0. Logical deduction should be used especially with evaluating linguistic expressions and either DEE or SDEE defuzzification method.

  Remark: In fact, the real procedure is a slight modification of the above one since both the input $\mathbf{A}_{k,X_1,\ldots,X_n}$ as well as the output $\mathbf{B}_{m,Y}$ can be slightly modified.

  #### Example

  Given the linguistic description
  R_1 : IF X is *very small* THEN Y is *very small*
  R_2 : IF X is *small* THEN Y is *small*
  Let the linguistic context for the variables X, Y be the interval $[0,1]$. For example, the values 0.22 and 0.09 are both small, but 0.09 is at the same time very small while 0.22 is not. Hence, if the input value is $x_0 = 0.09$, according to the given linguistic description, we expect the output value y_0 to be also very small, i.e. something around $y_0 = 0.09$ because of the presence of the rule R_1. Similarly, for the input value $x_0 = 0.22$ we expect y_0 small but not very small, i.e. around $y_0 = 0.22$. Such behavior is assured

by the logical deduction and DEE (SDEE) defuzzification method. Make experiments with the linguistic description MONOTONE.rb to see how this kind of approximate reasoning works.

(b) *Fuzzy approximation with conjunctions*

This is the well known Mamdani-Assilian method. The antecedent of each rule is specified as a fuzzy set $A_1 \times \cdots \times A_n \underset{\sim}{\subseteq} U_1 \times \cdots \times U_n$ and the succedent as a fuzzy set $B \underset{\sim}{\subseteq} V$. The disjunctive normal form assigned to the linguistic description is transformed into a fuzzy relation

$$R = R_1 \cup \cdots \cup R_m$$

where each R_j is a fuzzy relation obtained from the antecedent fuzzy set $A_{j1}, \ldots, A_{jn}$ and succedent fuzzy set B_j, $j = 1, \ldots, m$ using the formula

$$R_j(x_1, \ldots, x_n, y) = A_{j1}(x_1) \wedge \cdots \wedge A_{jn}(x_n) \wedge B_j(y)$$

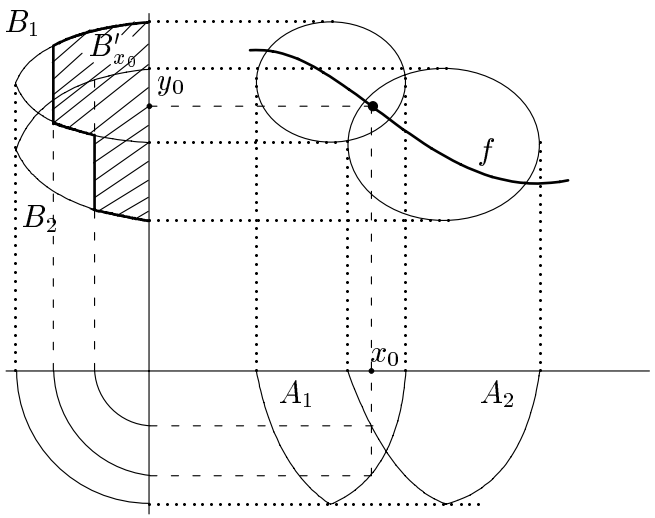
because the fuzzy IF-THEN rules characterize some functional dependence and thus, they can be taken as conjunctions.

Now, given input values $x_1 = x_{10}, \ldots, x_n = x_{n0}$ the output fuzzy set is computed as projection of the fuzzy relation R according to the formula

$$B_{x_{10},\ldots,x_{n0}}(y) = R(x_{10}, \ldots, x_{n0}, y) = \bigvee_{j=1}^{m} (A_{j1}(x_{10}) \wedge \cdots \wedge A_{jn}(x_{n0}) \wedge B_j(y)).$$

The goal is to approximate some function f hidden inside the fuzzy relation R. Thus, the shapes of the fuzzy sets A_j and B_j and the defuzzification method should be modified in such a way to fit f as best as possible. The best behavior of the functional fuzzy approximation can be obtained when the fuzzy sets are specified as fuzzy numbers together with the Center of Gravity defuzzification method.

The whole situation can be well seen from the following picture.



(c) *Fuzzy approximation with implications*

This method keeps interpretation of the rules as logical implications but its goal is approximation of a function.

The antecedent of each rule is specified by a fuzzy set $A \underset{\sim}{\subseteq} U$ and succedent by a fuzzy set $B \underset{\sim}{\subseteq} V$. The conjunctive normal form assigned to the linguistic description is thus transformed into a fuzzy relation

$$R = R_1 \cap \cdots \cap R_m$$

where each R_j is a fuzzy relation obtained from the antecedent fuzzy sets $A_{j1}, \ldots, A_{jn}$ and succedent fuzzy set B_j, $j = 1, \ldots, m$ using the formula

$$R_j(x, y) = (A_{j1}(x_1) \wedge \cdots \wedge A_{jn}(x_n)) \rightarrow B_j(y)$$

where $\rightarrow$ is the Łukasiewicz implication (realized using the formula $a \rightarrow b = \min(1, 1 - a + b)$, $a, b \in [0, 1]$). Now, given input values $x_1 = x_{10}, \ldots, x_n = x_{n0}$, the output fuzzy set is computed as projection of the fuzzy relation R according to the formula

$$B_{x_{10},\ldots,x_{n0}}(y) = R(x_{10}, \ldots, x_{n0}, y) = \bigwedge_{j=1}^{m} (A_{j1}(x_{10}) \wedge \cdots \wedge A_{jn}(x_{n0}) \rightarrow B_j(y)).$$

The role of this method for the practice is somewhat marginal because it is quite difficult to set right shapes of the fuzzy sets and defuzzification method to obtain convincing results.

- In case of TS-Singleton type of unit, there is one inference method, *Takagi-Sugeno inference*, available. Generally, we can characterize Takagi-Sugeno rules as rules,

which describe functional dependencies in vaguely characterized regions. The system of Takagi-Sugeno rules describes the course of some function f^A. We can understand f^A as function approximating some function f. Hence this method fall into the group of *fuzzy approximation* methods.

Suppose we have a system of TS-Singleton rules

$$\begin{aligned} \mathcal{R}_1 := \ & \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_{11}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_{1n}\ \mathsf{THEN}\ Y\ \mathsf{is}\ c_1 \\ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \mathcal{R}_m := \ & \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_{m1}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_{mn}\ \mathsf{THEN}\ Y\ \mathsf{is}\ c_m. \end{aligned}$$

Then the formula for the computation of the value of function f^A at the point $x_{10}, \ldots, x_{n0}$ is:

$$f^A(x_{10}, \ldots, x_{n0}) = \frac{\sum_{j=1}^{m}(A_{j1}(x_{10}) \wedge \cdots \wedge A_{jn}(x_{n0})) \cdot c_j}{\sum_{j=1}^{m}(A_{j1}(x_{10}) \wedge \cdots \wedge A_{jn}(x_{n0}))}$$

Note that in this case we compute directly the value of $f^A(x_{10}, \ldots, x_{n0})$ (no defuzzification is needed).

Let us remark that in all cases we can understand that the linguistic description can be used to generate some function

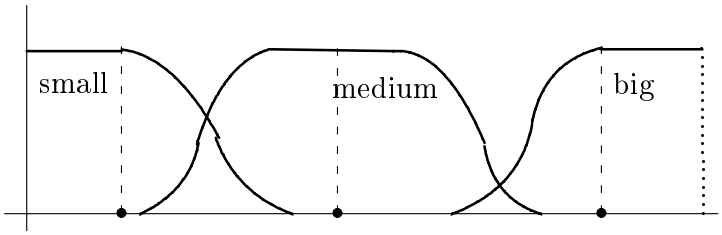
$$g : U_1 \times \cdots \times U_n \longrightarrow V.$$

Defuzzification

This determines the way of defuzzification of the fuzzy set obtained as a result of approximate reasoning. The following defuzzification methods are available:

(a) Simple defuzzification of Evaluating Expressions (SDEE - SimpDefuzzLingExpression)

This method is based on the assumption that the shapes of the fuzzy sets interpreting the evaluating linguistic expressions are of the form of S and Π curves. The defuzzification method defuzzifies according to the given type of the fuzzy set. The defuzzified value is taken as the edge of the kernel of the fuzzy set if it corresponds to "small" or "big" and center of gravity, if it corresponds to "medium" (see the figure below)



(b) Defuzzification of Evaluating Expressions (DEE - DefuzzLingExpression)

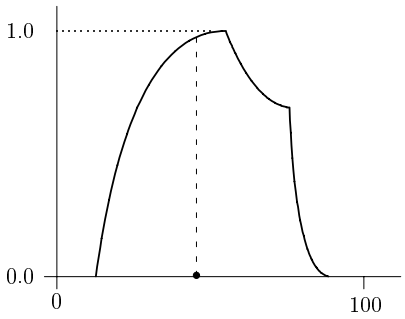
This is a slight modification of the SDEE method; the defuzzified value is found in the surroundings of the edge of the kernel of the S-fuzzy set.

(c) Center of Gravity Method (COG - SimpleCenterOfGravity)

This is classical method according to the formula

$$\mathrm{DEF}(A) = \frac{\sum_{x_i \in U} A(x_i) x_i}{\sum_{x_i \in U} A(x_i)}.$$

The result is depicted on the following figure.



(d) Modified Center of Gravity Method (MCOG - ModifiedCenterOfGravity)

This is a slight modification of the previous method when only upper part of the fuzzy set is taken into account.

(e) Mean of Maxima Method (MOM - MeanOfMaxima)

This is another classical method which takes the defuzzified values as the mean of all the values with maximal membership.

(f) Smooth Defuzzification of Linguistic Expression (SDLE - SmoothDeffuzLingExpr)

This is a method based on the theory of Smooth Logical Deduction which applies the fuzzy transform (F-transform) technic on the outputs obtaining using Logical deduction. Let us remind that logical deduction gives a partially continuous function. However, the course of the output using SDLE is smooth and continuous, this immediately follows from the properties of F-transform.

Description

In this window, the user can write down his/her comment to the linguistic description in concern.

Input and output variables

These tab-pages are used for specification of variables used in the linguistic description. The variables in the antecedent of IF-THEN rules are *input variables* and variables in the succedent are *output variables*.

Each variable is characterized by the following information.

- *Name* of the variable. Except for the full name, it is also possible to specify its short, which is used in headers of the IF-THEN rules.
- *Upper* and *Lower* bound – specification of the interval (universe), in which fuzzy sets interpreting the linguistic expressions will be defined. When the evaluating linguistic expressions are used, we speak about *linguistic context*, because these bounds mean the smallest and highest thinkable values in the given context. Note that in fuzzy control, still another term is often used, namely *scaling*.

  In principle, there are two possibilities. Either we specify the symmetric interval $[-u, u]$ where u is some real number, or we specify some interval $[u_1, u_2]$ where u_1, u_2 are real numbers fulfilling the condition $u_1, u_2 \geq 0$. The symmetric universe case is used mostly in fuzzy control.

  If lower or upper bound is changed, and there are fuzzy numbers of the form about⟨number⟩, dialog window appears with question, whether the numbers ⟨number⟩ have to be recomputed according to the new context. The same window appears if you are changing context of the succedent variable of TS-Singleton unit.
- *Discretization* of the universe. This information specifies number of points on the universe, in which the membership functions are computed. The higher is this number, the higher is the precision of the computation but the longer computation time. If symmetric context is specified then discretization must be odd to have precise center at disposal. Discretization has no sense for succedent variable in TS-Singleton unit.

The above information can be specified for each variable after pressing either of the buttons **Edit variable** or **Add variable**. The specified variable can be deleted by pressing **Delete variable**.

This window makes also possible to edit expressions used for each variable in the fuzzy IF-THEN rules. It is activated by pressing the button **Edit expressions**.

Recall that fuzzy logic enables to work with the linguistic variables, i.e. variables whose values are, in general, linguistic expressions such as “small”, “very big”, “roughly medium” etc. In LFLC 2000 they can be either the preset evaluating linguistic expressions, or arbitrary expressions set by the user. The evaluating linguistic expressions

are preset according to the linguistic analysis and some psychological investigations. However, they can be modified if necessary.

The other possibility is to set own *user expressions*. In principle, the user specifies only shapes of the membership functions and assigns them his/her own labels.

Editing expressions

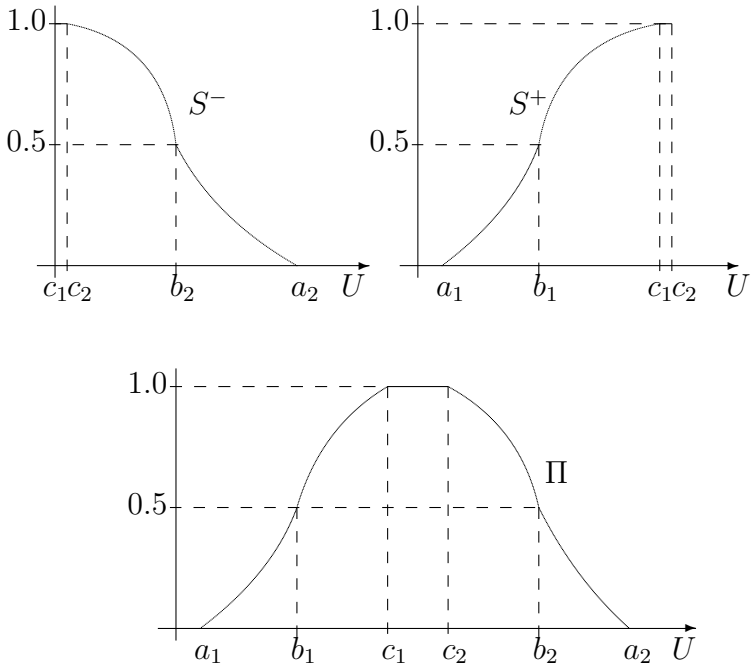
Pressing this button opens the window for editing all the linguistic expressions used for characterization of the values of the given variable. It has no sense for succedent variable in TS-Singleton unit, hence this button is not visible there. There are two tab-pages.

User

On this tab-page, we can specify shapes of membership functions of user defined labels. These shapes are in general given by the function

$$F(x,a_1,b_1,c_1,c_2,b_2,a_2) = \begin{cases} 0, & x \le a_1 \text{ or } x \ge a_2 \\ \frac{1}{2}\left(\frac{x-a_1}{b_1-a_1}\right)^2, & a_1 < x < b_1 \\ 1-\frac{1}{2}\left(\frac{c_1-x}{c_1-b_1}\right)^2, & b_1 \le x < c_1 \\ 1-\frac{1}{2}\left(\frac{x-c_2}{b_2-c_2}\right)^2, & c_2 < x < b_2 \\ \frac{1}{2}\left(\frac{a_2-x}{a_2-b_2}\right)^2, & b_2 \le x < a_2 \\ 1, & c_1 \le x \le c_2, \end{cases}$$

Using this function it is possible to construct three *standard types of membership functions* usually considered in fuzzy set theory, namely S^+, S^- and Π. The first two types are special cases of the third one, namely when setting $a_1 = b_1 = c_1$, we obtain a fuzzy set of type S^- and $c_2 = b_2 = a_2$ gives the fuzzy set of type S^+. All three fuzzy sets are basis for the meaning of the standard linguistic expressions and are depicted on the following figure:



This function can be simplified into trapezoidal (given by the parameters a_1, c_1, c_2, a_2 or even only triangular one ($c_1 = c_2$).

To set one of the above possibilities, one must push any of the buttons **Add Quadratic**, **Add Trapezoid**, **Add Triangular**. First, the name of the expression should be set, which will then be used when specifying the linguistic description on the tab-page Rules.

The shape of the fuzzy set can be specified either graphically by dragging any of the parameters (small squares on the curve) using mouse, or explicitly in the following columns:

Left support	a_1
Left Equilibrium	b_1
Left Kernel	c_1
Right Kernel	c_1
Right Equilibrium	b_1
Right support	a_2

The specified fuzzy set can be copied using the **Ctrl C** and **Ctrl V** keys. To do it, point by mouse the number of the fuzzy set on the left side. This action marks all its parameters. Then press **Ctrl V** key. The parameters of fuzzy sets can be copied also from one variable to another one. This can be done in two ways:

- Mark one or more fuzzy sets (by pressing **Shift** key at the same time), press **Ctrl C**, change the variable and press **Ctrl V**.
- All fuzzy sets are automatically copied from the variable specified as option "Copy from variable" after pressing **Add variable**.

The already specified fuzzy sets can be deleted by pressing **Delete** button.

A specific possibility is the button **Add Uniform**. This is widely used especially in fuzzy control. The result is n-triangular fuzzy sets (n specified by the user), which uniformly cover the universe. The typical number is 7, which are in symmetric context usually assigned the labels *negative big* (NB), *negative medium* (NM), *negative small* (NS), *zero* (ZE), *positive small* (PS), *positive medium* (PM) and *positive big* (PB)

The active fuzzy set is depicted in red and it can be modified in the same way as above. The other fuzzy sets are depicted in gray to enable the user to see the layout of all the possible values.

Standard

On this tab-page, shapes of the fuzzy sets of the standard linguistic expressions, which belong among evaluating linguistic expressions can be displayed. They are modeled using the standard membership function. LFLC 2000 does not make possible to modify them so far.

Go to Main Menu

Rules

This tab window enables to edit the fuzzy IF-THEN rules forming the linguistic description in concern. The window has the following columns:

- *Number of the rule*
  In the column, also a check box is present. This may be used for toggling the given rule as active/inactive. If the rule is made inactive then it is not deleted but it is not used in inference. This makes possible to realize various kinds of experiments when the influence of some rules on the result can be tested without necessity to delete them from the description. The information about active/inactive rules is saved, when *Save* command is used and reloaded again by means of *Open* command.
- Separate columns for each *Independent* (antecedent) and *Dependent* (succedent) variable where the rule is written down using the expressions of natural language (see editing the rules).

- *Group* This column contains information about rules that are equal (if they exist).
- *Inconsistency* This column contains information about inconsistency of rules, i.e. the rules that have the same antecedent but different succedent.
- *Redundancy* These are two separate columns for antecedent and succedent, where the user is informed that some rules are redundant. This may happen in two situations:

  (a) The succedent of both rules is the same but the antecedent of the first one is narrower than the antecedent of the second one. In this case, the former rule is redundant and it can be deleted from the description.

  For example, let us consider two rules

RuleNo	X_1	X_2	Y
1	sm	bi	me
2	ve sm	si bi	me

  Then Rule 2 is redundant since if X_1 is very small and X_2 is significantly big then X_1 is also small and X_2 is also big. Since both rules have the same succedent, the first one does the job of the second one as well.

  (b) The antecedent of both rules is the same but the succedent of the first one is narrower than the succedent of the second one. Since the latter is more precise, the first rule is redundant.

  For example, let us consider two rules

RuleNo	X_1	X_2	Y
1	sm	bi	bi
2	sm	bi	ve bi

  Then Rule 1 is redundant since very big Y is more precise than big Y.

  Decision whether the rules should be deleted or not depends on the user and it is not done automatically. *Redundancy* fields are not visible for TS-Singleton units.

The linguistic description is defined using the available linguistic expressions. For this, either the evaluating linguistic expressions (these are the standard ones) or the user defined ones can be used.

Evaluating linguistic expressions

The structure of the *evaluating linguistic expressions*, which can be used inside the fuzzy IF-THEN rules, is the following:

$$[\langle\text{sign}\rangle]\langle\text{linguistic modifier}\rangle\langle\text{basic expression}\rangle$$

Sign is + or −. This should be used when the corresponding variable has symmetric context.

Basic expressions are

expression	short
small	sm
medium	me
big	bi

The linguistic modifiers are

modifier	short
extremely	ex
significantly	si
very	ve
rather	ra
more or less	ml
roughly	ro
quite roughly	qr
very roughly	vr

The expressions can also be joined using the connectives *and* and *or* to make the complex ones.

Specific expressions, which can be used in the linguistic description, are fuzzy numbers. They can be set by

about⟨number⟩

where ⟨number⟩ is any number which falls within the linguistic context of the given variable. Moreover, a fuzzy zero *ze* is included together with its positive *+ze* and negative *-ze* parts. *+ze* and *-ze* have sense only for symmetric contexts. It is also possible to use linguistic modifier *roughly* with expression *ze*, *+ze* and *-ze*.

Additionally, we have special expressions

expression	short
undefined	undef
ignore	ignor

Value *undef* means, that a value of a given variable is not defined, i.e. it can attain arbitrary value. A rule, which includes variable with undefined value, is used when there are no other rules with the same combination of values of remaining variables.

Consider e.g. linguistic description

$$\mathcal{R}_1 := \text{IF } X_1 \text{ is } \textit{small} \text{ AND } X_2 \text{ is } \textit{very big} \text{ THEN } Y \text{ is } \textit{big}$$
$$\mathcal{R}_2 := \text{IF } X_1 \text{ is } \textit{small} \text{ AND } X_2 \text{ is } \textit{undef} \text{ THEN } Y \text{ is } \textit{small.}$$

Then, rule $\mathcal{R}_1$ will be used if X_1 is small and X_2 is very big and rule $\mathcal{R}_2$ will be used is X_1 is small and X_2 *is not* very big.

Value *ignor* is similar to *undef*, but now the variable is completely ignored for the given combination of values of remaining variables. Hence, if the value of X_2 in rule $\mathcal{R}_2$ in the previous example is *ignor* instead of *undef*, the rule $\mathcal{R}_1$ will be never used (and hence it is is superfluous). The difference between *undef* and *ignor* is in place only for Logical deduction inference method.

Moreover, it is possible to use *negation* of atomic expressions *small*, *medium*, *big* and *zero* (and its signed variants in symmetric contexts.) It is interpreted by operation $\text{not}A(x) = 1 - A(x)$. The syntax of negation operator is

$$\text{not}\langle\text{atomic expression}\rangle.$$

Editing the rules

The above expressions may be used anywhere in the fuzzy IF-THEN rules. An exception is succedent part of TS-singleton unit, where only real numbers can be used. To help the user, it is possible to view all the labels of the defined expressions (including the user ones) by making them visible after checking the appropriate box in the **View** submenu of the main menu. The expressions inside the rules can be then set by clicking the appropriate item of this menu.

Example

Let us consider a linguistic description with two independent variables *temperature* and *pressure*, and the dependent variable *position* (of the control cock). We suppose that the universe of all these variables is symmetric. Let the following two rules be given.

IF *temperature* is *positive small* and *pressure* is *negative big*
THEN *position* is *positive medium*
IF *temperature* is *negative very small* and *pressure* is *positive significantly big* THEN *position* is *positive roughly small*

Then this description can be written down in the *Rules editing window* as follows:

temperature	pressure	position
+sm	− bi	+me
−ve sm	+si bi	+ro sm

It is possible to copy, cut, and paste the rules. To do it, mark one or more rules (using the **Shift** key)) by clicking on the number of the rule. To copy, press **Ctrl C**, to cut press **Ctrl X** and to paste press **Ctrl V**.

In a lot of situations, we need separate rules for positive and negative cases. For example, in fuzzy control we need the values below and above the set point. The corresponding linguistic expressions are usually symmetric and differ only by opposite signs. To simplify the work, we can use the *Invert sign* $+, -$ option in the **Edit** submenu of the main menu. Then mark appropriate rules (and possibly copy them) and press **Ctrl I**. All the described actions can also be done using the **Edit** submenu of the main menu.

Sorting the rules

To get better orientation in the linguistic description, it is possible to sort it. This is achieved by clicking on the name of the corresponding variable. Clicking more variables sorts them in the given order. The values are sorted according to the value (from *small* to *big*) and sharpness of the linguistic hedge (from *extremely* to *very roughly*).

Go to Main Menu

Testing Linguistic Descriptions

This function makes possible to test behavior of the designed linguistic descriptions. If a file with linguistic description is open, press the button **Test**. The testing window is divided into four areas.

Upper left area

This is input area, on which input of each defined antecedent (independent) variable can be set in one of two ways:

- The input value is set graphically on the horizontal line defined for each antecedent variable. Drag a small red triangle along the line and drop it on the required position.
- The concrete value of each variable is specified explicitly in the input field. This should be used if the graphical input is not sufficiently precise due to screen resolution.

To each given value, the corresponding most typical evaluating expression is displayed, i.e. the linguistic expressions which best characterizes the given value in the given linguistic context.

Lower left area

On this area, the text form of the elaborated linguistic description is displayed. Note that the saved linguistic description can be viewed or edited using an ordinary editor (this is not recommended).

Upper right area

This is the output area where the result of inference is continuously displayed depending on the chosen inference and defuzzification method. Namely, the shape of the resulting fuzzy set with the marked defuzzified value, the concrete value and the most typical evaluating expression and finally, all the fired rules are displayed. Note that defuzzification is not performed for TS-Singleton units.

It is also possible to change both the inference as well as the defuzzification and see, how the result changes.

Lower right area

This global output area where the result of inference for all the possible values of the chosen antecedent variable are depicted.

- In case of more than one antecedent variable, we obtain two-dimensional projection of the function g generated by the linguistic description (see section Type of inference) for all the values of the chosen projection variable and some fixed values of the other ones.

  The projection variable can be specified in the corresponding choice field. Since, of course, we cannot go through all the possible values of the projection variable, we must specify number of equidistant points over the set of all possible values of the projection variable. This is set in the field *Enabled steps*.

- It is also possible to display three-dimensional projection of g by clicking the button **Surface**. Two projection variables as well as the number of steps are specified analogously as above. The depicted surface can be turned using mouse by draging the colored dots at the end of each axis.

Input/Output

On this tab page, it is possible to work with data stored in a text file. Two possibilities are available.

- Computing values of the function g generated by the linguistic description (see section Type of inference).
- Learning linguistic description from the data.

Computing function values

Before using this option, the linguistic description must be defined. Then press the button **Import file** to import the text file. Its structure is the following:

Text line 1	Var X_1	Var X_2	...	Var X_n	Var Y
Text line 2	xxxx	xxxx	xxxx	xxxx	xxxx
Data line 1	999.99	999.99	999.99	999.99	999.99
⋮	⋮	⋮	⋮	⋮	⋮
Data line m	999.99	999.99	999.99	999.99	999.99

Each column corresponds to one variable including the independent one. Their number depends on the definition of the linguistic description and on the definition of the variables on the tab-page input and output variables.

Text lines are arbitrary and are ignored. The data are in the fixed point format of arbitrary length but they must be separated by at least one space. If the file contains more columns than is the number of the defined variables in the linguistic description then the additional ones are ignored.

The dependent variable contained in the file is supposed to be used only for comparison and is displayed in the column headed by "est." (estimation). The column headed by Y is used for computation of the functional values of the function g in each data line of the imported file. The computation is realized after pressing the button **Compute**. Note that the values should fall into the defined linguistic context. If not, the corresponding bound (lower or upper one) is automatically taken instead of the value exceeding it.

The imported file can be modified using the buttons **Add Row** and **Delete Row**. The file including the computed values can be saved using the button **Save File**. The file is added the extension '.out'.

Learning linguistic description

Before using this option, define all the antecedent and succedent variables including the linguistic context. Then press **Import file** button to import the text file as above. Let us stress that the number of columns in the file must not be smaller than the number of all defined variables. Finally press **Learn/Linguistic learning** button. The new learned rules are added to the existing linguistic description, which is saved under the prompted name.

Linguistic learning method

This is based on special method for finding an evaluating linguistic expression, which is *typical* for a given value x_0 in the linguistic context of some variable. This means that the interval $[u_1, u_2]$ of all its possible values is *fuzzily* divided into "extremely small", "very small", "roughly small", etc. values, and similarly for "medium" and "big". If the value x_0 falls into such area, it is considered as *typical representative* of the corresponding expression.

During the learning procedure, each item in the file is replaced by the linguistic expression using the above method. Hence, each line of the imported file leads to one linguistically characterized fuzzy IF-THEN rule.

Example

Let the defined variables be X_1, X_2, Y with the respective linguistic contexts $[1, 10]$, $[-5, 5]$, $[0, 1]$. Let the data line be

$$1.7 \qquad -3.25 \qquad 0.83$$

The typical linguistic expressions found by the above method are in the following table:

context	value	typical expression
$[1, 10]$	1.7	very small
$[-5, 5]$	-3.25	roughly big
$[0, 1]$	0.83	big

Thus, the new learned rule is

IF X_1 is *very small* AND X_2 is *roughly big* THEN Y is *big*

Duplicate rules are automatically deleted. Other reduction of the description should be done either manually or semi-automatically using special buttons in the Rules tab-page.

Note that other learning method is in preparation.

Go to Main Menu

LFLC 2000 COM Server

A special COM object RuleBaseCOM is prepared, which provides inference mechanism on the basis of the linguistic description designed using LFLC 2000 for user-written client applications. Before writing the application, the object must be registered into the Windows system. This is normally automatically performed during the installation process.

The RuleBaseCOM object is installed into `RBaseCOM` subdirectory of the LFLC 2000 root directory. This subdirectory contains the following files:

`register.bat`	Batch file which registers the RuleBaseCOM object into Windows system. It uses the "`tregsvr.exe`" Borland Inprise utility. The registration must be done before using the object in any applications. To register, just run this file.
`RBaseCOM.dll`	File containg the RuleBaseCOM object.
`RBaseCOM.tlb`	Type library for importing object into the programming development IDE.
`tregsvr.exe`	Borland Turbo Register Server – COM Server Registration utility. Version 1.1. Copyright (c) 1997, 2000 Inprise Corporation

In case that the COM object has not been registered during installation, run "`register.bat`" file.

After successful registration, the RuleBaseCOM object can be used in own applications. The object communication interface is as follows:

LoadFromFile(BSTR FileName)	Loads rulebase with specified *FileName*
int NumInputVars()	Returns number of input variables of the loaded rulebase
double HiBoundOfVar (**int** VarIndex)	Returns upper bound of the *VarIndex* variable context
double LoBoundOfVar (**int** VarIndex)	Returns lower bound of the *VarIndex* variable context
BSTR VarName (**int** VarIndex)	Returns the name of the *VarIndex* variable
double Inference (TVariantInParam Inputs)	Performs inference on *Inputs* values

To understand the use of the COM object, the following two examples are placed in the `LFLC 2000\RBaseCOM\examples` directory. They demonstrate the use of RuleBaseCOM object in Borland C++ Builder 5.0 development environment.

ConsoleApp	Shows how to use RuleBaseCOM object in console application
WinGUIApp	Shows how to use RuleBaseCOM object in Windows GUI application

LFLC 2000 MATLAB/Simulink client

This is a special MATLAB/Simulink S-Function which can be used in Simulink schemes. The S-Function forming a new block "LFLC Inference" is contained in LFLCLibrary.mdl

file which is installed in the `LFLC 2000\matlab` directory.

Before simulation, it must be instructed to read the name of the file containing the linguistic description (file "`xxx.rb`") and the number of orders of the elaborated rules from the actual linguistic description to be displayed. This should be done in the dialog window "Block Parameters" for S-Function "Inference".

Let us remind that with different inputs the numbers of elaborated rules differs. If you set the value to 0 then no order will be displayed. Any other higher settings will increase the dimension of output array. Then, "LFLC Inference" S-Function is returning the orders of used rules and filling all others free fields of the array by value 0.

Several demonstration examples are provided in the directory `LFLC 2000\LFLCDemo`.

LFLC 2000 MATLAB client

This is a special MATLAB mex-function which can be used in MATLAB programs. The name of this function is LFLCInfer. The main purpose of this function is to compute inference based on the given input values and display the number of elaborated rule (or rules). The LFLCInfer mex-function satisfies the following calling conventions. Before computing inference result it must be initialized by loading appropriate linguistic description ("`xxx.rb`" file). The LFLCInfer function expects two input arguments.

- The first parameter is the ID number of the description. Thanks to this ID number it is possible to work with several linguistic descriptions at the same time. The ID number can range from 0 to 9, so up to ten descriptions can be loaded simultaneously. When only one linguistic description is loaded then this parameter can be omitted the default value 0 will be used in this case.
- The second input argument has two distinct purposes which are recognized depending on its type. The string value is used for saying the filename with the full path, for loading and initializing the appropriate linguistic description. When the second parameter is one dimensional array of double number values, it is recognized as an input into inference and its value is returned as output value of calling LFLCInfer mex-Function.

A short illustration of calling the LFLCInfer function in MATLAB environment follows:

```
% loading of linguistic description
% note that the ID number of description is omitted
LFLCInfer('twovar.rb')
```

```
% computing inference value
result = LFLCInfer([-0.5, -0.5])

% the same result will be obtained in this case when ID is specified
result = LFLCInfer(0, [-0.5, -0.5])

% when using more than one description at the same time there must be
% used the ID number of linguistic description
LFLCInfer(1, 'monotone.rb')

% output from one inferention can be directly used as input to another one
result = LFLCInfer(1, LFLCInfer([-0.5, -0.5]))

% for obtaining information about fired rules it is necessary to change
% the type of required result into an array
[result, rule] = LFLCInfer([-0.5, -0.5])

% sometimes more than one rule is used during inference method and
% information about them could be obtained in similar way
[result, rule1, rule2, rule3] = LFLCInfer([-0.5, -0.5])

% in previous example when rule2 and rule3 is set to 0 it
% means that only one rule were used during inference
```

Note that both LFLC 2000 MATLAB/Simulink client and LFLC 2000 MATLAB client uses the LFLC 2000 COM Server RuleBaseCOM, so it must be registered into Windows system before using this special-written clients for MATLAB environment.

Go to Main Menu