The concept of LFLC 2000—its specificity, realization and power of applications

Antonín Dvořák, Hashim Habiballa, Vilém Novák*, Viktor Pavliska

Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, 30 Dubna 22, 701 03 Ostrava 1, Czech Republic

Received 8 September 2002; accepted 1 February 2003

Abstract

Linguistic Fuzzy Logic Controller (LFLC) 2000 is a complex tool for the design of linguistic descriptions and fuzzy control based on these descriptions. Unique methodology and theoretical results upon which is LFLC 2000 based are presented. Then, main purposes of it are sketched and some implementation aspects are discussed. Presentation of existing and perspective applications concludes the paper.

© 2003 Published by Elsevier Science B.V.

Keywords: Fuzzy logic; Approximate reasoning; Logical deduction; Approximation of functions; Fuzzy control

1. Introduction

The core of applications of soft computing methods in the practice lays in the use of approximate reasoning methods. These are very general techniques, which can be applied in control, decision making, classification, pattern recognition and elsewhere, providing that a description of the given system (situation) is at disposal. The latter is supposed to be based on using set(s) of rules, each of which characterizes a very specific detailed behavior of the system and may be expressed imprecisely. Imprecision raises from various sources—too large complexity, insufficient amount of precise information, presence of human factor, necessity to spare time or money, etc. Very often, combination of more such factors is present.

The necessary theory is already sufficiently developed and thus, it is possible to implement it in a software. There already exists a lot of software packages aimed at applications of approximate reasoning, possibly joined with (fuzzy) neural networks.

In this paper, we will present a concept of fuzzy logic implementation which differs from the other implementations of fuzzy logic known to the authors till now. It is specific mainly by two aspects: semantics of certain natural language expressions is implemented in such a way that computer behaves as if “understanding” them, and linguistically formulated conditional statements are used in genuine logical deduction.

The whole concept is called Linguistic Fuzzy Logic Controller (LFLC). Its software implementation now exists in two versions. The old one, LFLC 1.5 is written in Borland Pascal under MS-DOS. We have obtained a lot of experiences with it including several real applications. The new version, LFLC 2000 is written in C++ under Windows and it is fully object oriented system, which is now joined with MATLAB/Simulink to enable simulation of wide class of systems.
In Section 2, we will briefly overview the theoretical background. Further, we provide a short description of the main features, representation of the data and outline the inner structure of the software implementation. Finally, we mention applications and outline intentions for future development.

2. Theoretical background

2.1. Linguistic descriptions and their elaboration

The theoretical background of LFLC lays in formal fuzzy logic in broader sense (FLb), which is an extension of that in narrow sense (FLu) (for the detailed presentation of both logics see [12]). The theory provides elaboration of that part of the semantics, which consists of the so-called evaluating and conditional linguistic expressions. The former are expressions such as “small, roughly medium, very big”, etc. The latter are the well-known fuzzy IF–THEN rules. These are usually gathered into sets called linguistic descriptions which take the form

\[ \mathcal{R}_1 := \text{IF } X \text{ is } \mathcal{A}_1 \text{ THEN } Y \text{ is } \mathcal{B}_1 \]

...

\[ \mathcal{R}_m := \text{IF } X \text{ is } \mathcal{A}_m \text{ THEN } Y \text{ is } \mathcal{B}_m \]

where \( \mathcal{A}_j, \mathcal{B}_j \) are the mentioned evaluating linguistic expressions. They characterize property of some features of objects, for example size, volume, force, strength, etc. Since usually we are not interested in the concrete objects and their features, we replace them by some real numbers which are then represented by the variables \( X \) and \( Y \). Thus, values of \( X \) and \( Y \) represent, e.g., values of temperature, pressure, price, etc. The linguistic expression of the form “\( X \) is \( \mathcal{A} \)” is called the evaluating linguistic predication.

Fuzzy IF–THEN rules serve as a basis for approximate reasoning, which is a method for finding a conclusion on the basis of the imprecise initial information concentrated in the form of linguistic description and some new information. There are two fundamental approximate reasoning methods:

(a) Linguistically based fuzzy logical deduction, i.e. finding a formal conclusion when fuzzy IF–THEN rules are treated as linguistically characterized logical implications.

(b) Fuzzy approximation of a function, i.e. finding a function which approximates some only imprecisely known function, whose course is estimated using the linguistic description.

The interpretation of the linguistic description significantly depends on the above chosen method.

The usual implementations of approximate reasoning focus on the method (b). Our concept of LFLC implements both methods but its main strength lays in the method (a).

2.2. Fuzzy approximation of a function

In this case, each evaluating predication “\( X \) is \( \mathcal{A} \)” is assigned some formula \( A(x) \) of predicate fuzzy logic. The whole linguistic description is then assigned one of two special formulas called the disjunctive and conjunctive normal form.

The disjunctive normal form is the formula

\[ \text{DNF}(x, y) := \bigvee_{j=1}^{m} (A_j(x) \land B_j(y)). \]  

In this case, each rule is assigned a conjunction of formulas \( A_j(x) \) and \( B_j(y) \) and all of them are joined by disjunction. We speak also about functional interpretation of the linguistic description.

The alternative possibility is the conjunctive normal form

\[ \text{CNF}(x, y) := \bigwedge_{j=1}^{m} (A_j(x) \Rightarrow B_j(y)). \]  

In this case, each rule is assigned an implication between the formulas \( A_j(x) \) and \( B_j(y) \) and all of them are joined by conjunction. We speak about logical interpretation of the linguistic description. Recall, however, that the main goal is still fuzzy approximation of a function.

Both Eqs. (1) and (2) correspond to certain fuzzy relations after the following assignment. Let us consider a couple of sets

\[ w = \langle U, V \rangle, \]

\[ 1 \text{ In fact, the problem is more complex since we must precisely specify the language, the structure and assignments of all symbols from the language. For the purpose of this paper, we simplify the explanation. The interested reader is referred to [12–14].} \]
which will be taken as a model. In the practice, we usually consider $U, V$ to be some closed intervals of real numbers (as mentioned above these represent, e.g. temperature, angles, prices, etc.). Furthermore, each formula $A_i(x)$ is assigned a fuzzy set $A_{w,i} \subseteq U$ and $B_j(y)$ is assigned a fuzzy set $B_{w,j} \subseteq V, j = 1, \ldots, m$. Then the conjunctive normal form (1) is assigned a fuzzy relation $R_{\text{CNF}}(u, v) \subseteq U \times V$ given by the membership function

$$R_{\text{DNF}}(u, v) := \bigwedge_{j=1}^{m} (A_{w,j}(u) \land B_{w,j}(v)), \tag{4}$$

and the conjunctive normal form (2) is assigned a fuzzy relation $R_{\text{CNF}}(u, v) \subseteq U \times V$ given by the membership function

$$R_{\text{CNF}}(u, v) := \bigwedge_{j=1}^{m} (A_{w,j}(u) \land \neg B_{w,j}(v)), \tag{5}$$

where $a \rightarrow b = \min(1, 1 - a + b), a, b \in [0, 1]$ is the so-called Lukasiewicz implication. Note that (4) is the well-known Mamdani-Assilian formula used in most applications of fuzzy control.

Now, let some value $u_0 \in U$ be given (i.e. this is some precise measurement of, say, temperature, on the basis of which we should find a proper control action). Then using (4) or (5) we derive a fuzzy set $B_{u_0} \subseteq V$ with the membership function

$$B_{u_0} = \{R(u_0, v) \mid v \in V\} \tag{6}$$

The result of this kind of elaboration of fuzzy IF-THEN rules is an approximating function $f^A : U \rightarrow V$ given by the formula

$$f^A(x) = \text{DEF}(B_x), \quad x \in U, \tag{7}$$

where DEF is a defuzzification function. Recall that the defuzzification function is a function $\text{DEF} : \mathcal{F}(U) \rightarrow U$ where $\mathcal{F}(U)$ is the set of all fuzzy sets on $U$.

It has been proven that every continuous function on a compact set can be approximated using either conjunctive or disjunctive normal form with arbitrary precision, independently on the chosen defuzzification methods (cf. [12]). However, a certain classification of defuzzification methods can be provided and the best possible one is Center of Gravity (COG) method. More about approximation properties can be found also in [13, 14].

### 2.3. Linguistically based fuzzy logical deduction

The most specific feature of LFLC is the possibility to realize a fuzzy logical deduction when the rules are interpreted as linguistically characterized logical implications.

#### 2.3.1. Linguistic aspect

In the concept of LFLC, we deal with the mentioned evaluating linguistic expressions (possibly with signs) which have the general form

$$(\text{linguistic modifier})(\text{atomic term}), \quad (8)$$

where (atomic term) is one of the words “small”, “medium”, “big”, or “zero” (possibly also arbitrary symmetric fuzzy number) and (linguistic modifier) is an intensifying adverb such as “very”, “roughly”, etc.

The linguistic modifiers in (8) are of two basic kinds, namely those with narrowing and widening effect. Narrowing modifiers are, for example, “extremely”, “significantly”, “very” and widening ones are “more or less, roughly, quite roughly, very roughly”. We will take these modifiers as canonical. Note that narrowing modifiers make the meaning of the whole expression more precise while widening ones do the opposite. Thus, “very small” is more precise than “small”, which, on the other hand, is more precise than “roughly small”.

The meaning of each linguistic expression $\mathcal{A}$ has two constituents: the intension $\text{Int}(\mathcal{A})$ and extension $\text{Ext}(\mathcal{A})$ in some model (this is often called the possible world).

Intension of the linguistic expression is a formal characterization of the property denoted by it on the level of formal syntax. It can be interpreted as a fuzzy set of special formulas. However, it is a rather abstract concept, which in concrete situation (context) determines some fuzzy set of elements. Mathematically this means that a model $w$ is given whose support is some set $U$ (taken usually as a closed interval of real numbers $[t_0, t_1]$).

---

2 Alternatively, it can be any residuation operation based on some continuous $t$-norm.

---

3 They have the form $A : = [a_i/A_i[t] \in M]$ where $A(x)$ is a formula, $M$ is a set of constants and at is an evaluation of the instance $A_i[t]$. For the details see [12].
numbers). Then the extension of \( \mathcal{A} \) is some fuzzy set of elements \( \text{Ext}_w(\mathcal{A}) \subseteq U \), which is determined by its intension \( \text{Int}(\mathcal{A}) \). Note that for each concrete situation, different model should be considered. However, intension is still the same.

Note that the concepts of intension and extension formalizes the following intuitive situation: we can speak about high temperature, high pressure, high tree, etc. But high temperature may mean 100 °C at home or 1000 °C in metal melting process, and similarly in other cases. This cannot be satisfactorily formalized without the mentioned concepts.

In the terminology used in LFLC, we speak about linguistic context in which the given evaluating expression is used since in the practice, it requires setting the minimal and maximal possible values which can be attained by the used variables.

Let us stress that the extensions of the evaluating expressions are fuzzy sets of the form of the so-called S- and II-curves, as is depicted in Fig. 1. More on the formal theory of evaluating linguistic expressions can be found in [9,12].

### 2.3.2. Fuzzy logical deduction

Unlike fuzzy approximation, where we deal with fuzzy sets in a model (i.e. on the level of semantics), logical deduction must proceed on syntax. Instead of the detailed formal description, we will demonstrate the behavior of the logical deduction on an example.

Let us consider a linguistic description consisting of two rules:

\[
\begin{align*}
\mathcal{R}_1 &:= \text{IF } X \text{ is small AND } Y \text{ is small THEN } Z \text{ is big}, \\
\mathcal{R}_2 &:= \text{IF } X \text{ is big AND } Y \text{ is big THEN } Z \text{ is small}.
\end{align*}
\]

These rules are assigned intensions \( \text{Int}(\mathcal{R}_1), \text{Int}(\mathcal{R}_2) \), which can schematically be written as

\[
\begin{align*}
\text{Int}(\mathcal{R}_1) &= (\text{Sm}_x \land \text{Sm}_y) \Rightarrow \text{Bi}_z, \quad (9) \\
\text{Int}(\mathcal{R}_2) &= (\text{Bi}_x \land \text{Bi}_y) \Rightarrow \text{Sm}_z. \quad (10)
\end{align*}
\]

Furthermore, let \( X, Y, Z \) be interpreted in a model which will consist of three sets \( U = V = W = [0, 1] \).

Then small values are some values around 0.3 (and smaller) and big ones some values around 0.7 (and bigger). Of course, given the input, e.g. \( X = 0.3 \) and \( Y = 0.25 \) then we expect the result \( Z \approx 0.7 \) due to the rule \( \mathcal{R}_1 \). Similarly, for \( X = 0.75 \) and \( Y = 0.8 \) we expect the result \( Z \approx 0.25 \) due to the rule \( \mathcal{R}_2 \).

The value 0.3 is represented in the formal system by a certain intension \( \text{Sm}'_x \) and similarly, the value 0.25 is represented by \( \text{Sm}'_z \).

Then the inference rule of modus ponens is applied on \( \text{Sm}'_x, \text{Sm}'_y \) and the implication (9). The result is the intension \( \text{Bi}'_z \). The latter is to be interpreted as some fuzzy set \( B' \subset W \).

To obtain one concrete value, the resulting fuzzy set \( B' \) should further be defuzzified. However, we deal with evaluating linguistic expressions, whose interpretation has always one of the three possible forms depicted in Fig. 1. Therefore, standard defuzzification methods such as COG do not work properly. Instead, we have developed a special method, which we call Defuzzification of Evaluating Expressions (DEE).

This method classifies first the type of the membership function and then decides the defuzzification, as is depicted in Fig. 1. There are two versions of the DEE method, namely simple which first classifies the resulting fuzzy sets in types “small”, “medium” and “big” and then defuzzifies it using Last of Maxima,

\[
\begin{align*}
\text{DEE(very small)} &\leq \text{DEE(medium)} \leq \text{DEE(big)}
\end{align*}
\]

Fig. 1. Form of fuzzy sets corresponding to the meaning of the evaluating linguistic expressions and the DEE defuzzification.
Center of Gravity or First of Maxima methods, respectively. The second one uses a sophisticated algorithm to choose a value close to these dependingly on the specific shape of the membership function. In our case, when the input is \( X = 0.3 \) and \( Y = 0.25 \) then both values correspond to “small” and thus, with respect to the rule \( \mathcal{R}_1 \), the resulting linguistic corresponds to “big” and thus, after its interpretation in the model and defuzzification using the DEE method, we obtain the result \( Z \approx 0.7 \), i.e. a value being intuitively big. In other words, we obtain the result which, on the basis of the form of the given rules, should be expected. Similarly, the input values \( X = 0.75 \) and \( Y = 0.8 \) would lead to the value \( Z \approx 0.25 \) due to the rule \( \mathcal{R}_2 \).

To summarize: in the case of fuzzy approximation, we form the special formulas DNF or CNF on the level of syntax, interpret them in some model and then find the approximation on the level of semantics only. In the case of linguistically based fuzzy logical deduction we interpret the rules on the level of syntax, transform measurement also to this level, realize formal logical deduction and then interpret the result in some model.

3. Purpose of LFLC

Recall that the original idea of the fuzzy controller proposed by Zadeh [18], and Mamdani and Assilian [8] is to translate knowledge of the human control operator into mathematical description in a way to mimic a successful course of his/her control. Since such knowledge is usually expressed using evaluating linguistic expressions, the main problem consists in translation of the linguistic expressions into mathematical form.

The main purpose of LFLC software system is the design, testing and learning of linguistic descriptions. These descriptions can be further used in control, decision support and other applications (see Section 7). We can distinguish the following main tasks realized by LFLC:

- **Design of linguistic descriptions**: LFLC allows to use various pre-defined linguistic expressions (e.g. \( small \), \( about \) 5, \( more \) or \( less \) medium, etc.). It has also means for analysis of several properties of linguistic descriptions (sorting, detection of identical or inconsistent IF–THEN rules, etc.).

- **Design and modification of user expressions**: In addition to pre-defined linguistic expressions the user can also define and use his own expressions in situations when standard ones are for some reasons insufficient.

- **Testing of inference over designed linguistic descriptions**: LFLC allows the user to visualize the behavior of the linguistic description for various (crisp) observations. He/she can select various inference and defuzzification methods (cf. Section 2) and see projections with respect to individual variables or three-dimensional control surface. There is also information about IF–THEN rules fired and most suitable linguistic expressions assigned to crisp values from input or output intervals.

- **Learning of linguistic description from experimental data**: LFLC implements two methods for learning of linguistic description from data sets. The first method is based on the ability to find proper evaluation linguistic expression corresponding to the given value. The resulting linguistic description should be used for finding conclusions using logical deduction.

The second possibility is based on the theoretical results published in [13,14] and it enables to find a linguistic description interpreted by DNF with the prescribed accuracy of approximation of the data understood as specification of some function.

4. Important data structures and algorithms

The LFLC software system is implemented in C++ programming language with full use of object-oriented methodology. In the following we describe the most important data structures used for the implementation of linguistic descriptions and related notions. In this section, we use the common C++ terminology such as class, method, instance, etc.

Fig. 2 shows the hierarchy of the main classes of LFLC. There are classes for representation of fuzzy sets (CFuzzySet and its derivatives), classes which allow assignments of fuzzy sets to the linguistic
expressions (CLinguisticSettings), classes for representation of operations on truth degrees and induced operations on fuzzy sets (C1ArgOper, C2ArgOper and their derivatives), class which represents an overall linguistic description CRuleBase and several other auxiliary classes.

There are three important concepts, namely the fuzzy set, the semantical rule which assigns fuzzy sets to linguistic expressions and, finally, the linguistic description. Each of these concepts has a corresponding counterpart in the implementation, namely the class. These classes CFuzzySet, CLinguisticSettings and CRuleBase are not, however, on the same level of generality. Instances of the CFuzzySet are members of CLinguisticSettings and, similarly, instances of the CLinguisticSettings are members of CRuleBase.

4.1. Representation of fuzzy sets

The class CFuzzySet is the basis for representation of all types of fuzzy sets and implements their two common properties, namely the linguistic context (see Section 2.3) and the membership function. Therefore, it contains as its member an instance of the class CUniverse which is designed for representation of the linguistic context (or universe) of (generally multi-dimensional) fuzzy set. Further, it has the method GetMembDeg that returns membership degree for every point from the context. This method is pure, no particular membership function is defined in the class CFuzzySet itself. It has to be defined by classes derived from it.

There are four basic types of fuzzy sets derived from CFuzzySet which can be distinguished by implementation of the membership function:

- Discrete fuzzy sets, represented by the class CFuzzyDiscreteSet, define fuzzy sets on discretized universe which are widely used especially in inference, defuzzification and other computational routines. There is also special implementation of discrete fuzzy set CCompressedFuzzyDiscreteSet designed for better memory usage, where the zero membership values lying outside of the support of the fuzzy set are not stored in memory.

- Parametric fuzzy sets implemented in the classes CFuzzyQuadraticSet, CFuzzyTriangleSet and CFuzzyTrapezoidalSet are most often used as extensions of evaluating linguistic expressions.

- Fuzzy sets with membership function represented by means of $\alpha$-cuts (see, e.g. [7]) implemented by class CFuzzyAlphaCutSet. Each $\alpha$-cut is represented by class CAlphaCut.

- General fuzzy sets with membership function defined using arbitrary function given by the user are implemented in the class CFuzzyFuncDefSet.

Operations on fuzzy sets are implemented by classes derived from C1ArgOper and C2ArgOper. Classes derived from C1ArgOper implement prefix operators:

- CModifier represents modifier of Zadeh’s type (e.g. power).
- CNegation represents negation.
The subclasses of C2ArgOper implement as methods various types of intersections, unions, implications and differences of fuzzy sets. In this way, e.g. intersection have no pre-defined t-norm, but it is possible to use any t-norm implemented by some subclass of C2ArgOper.

4.2. Representation of semantical rule

The semantical rule which assigns extensions (fuzzy sets) to evaluating linguistic expressions is represented by the derivatives of the class CSettings. It is pure class which contains the extensions of atomic evaluating expressions (array of instances of classes derived from CFuzzyParamSet) and meanings of modifiers (array of instances of class CModifier) on appropriate context (class CUniverse), i.e. all the necessary information needed for the computation of the meaning of linguistic expressions (see Section 2). Derived classes have to implement the fundamental method GetMeaning that should return appropriate extension of a given linguistic expression. Derived classes have the following structure:

- CBasicSettings—base class for the situations when evaluating expressions are parametrically defined families of fuzzy sets:
  - CLinguisticSettings—meanings of evaluating expressions are fuzzy sets with quadratic membership function,
  - CTriangSettings—triangular membership functions,
  - CTrapezoidSettings—trapezoidal membership functions.
- CLinCSettings—meaning is a linear combination of input variables, this class is used for the evaluation of the output in Takagi–Sugeno models [17].

4.3. Representation of linguistic description

The linguistic description is represented by the class CRuleBase and the overall structure of that class is shown in Fig. 3. It contains arrays of antecedent and succedent fuzzy variables (that corresponds to inputs and outputs of fuzzy controller), and pointers to inference and defuzzification routines. The intended purpose of the classes is the following:

- CFuzzyVariable represents one fuzzy variable and an appropriate row from linguistic description, which is an array of instances of the class CValue.
- CValue represents the linguistic expression as a character string and also the extension of the lin-

![Fig. 3. Structure of CRuleBase.](image)
There are two derived classes:

- **CLinCValue** represents the meaning of succeeding expression in Takagi–Sugeno model (cf. [17]) in the form of array of real numbers coefficients of linear combination of input variables,
- **CFuzValue** represents the extension of linguistic expression by means of the instance of the class CFuzzyDiscreteSet.

### 4.4. Inference and defuzzification

The LFLC software system implements various inference and defuzzification methods. For thorough discussion, see Section 2 and [12]. It allows the user to compare different combinations of them and, consequently, types of approximate reasoning mechanisms. The inference methods are (cf. Section 2):

- Fuzzy logical deduction (see [4]).
- Fuzzy approximation using disjunctive normal form (the linguistic description is used to characterize a crisp function, but IF–THEN rules are interpreted as implications).
- Fuzzy approximation using conjunctive normal form (the linguistic description is also used to characterize a crisp function).

Defuzzification methods implemented in LFLC are:

- **Center of Gravity (COG)**,
- **Mean of Maxima (MOM)**,
- **Modified Center of Gravity (ModCOG)**,
- **Defuzzification of Linguistic Expressions (DEE)**,
- **Simple Defuzzification of Linguistic Expressions (SDEE)**.

For the detailed discussion of the defuzzification methods, see [5]. Modified COG is described in [2]. Recall that the DEE method is a special defuzzification method designed for use in conjunction with Fuzzy Logical Deduction inference method (see also [4]).

### 5. User interface

The user interface of the LFLC application is designed under 32-bit Windows platform. Therefore, it could be used under the whole family of MS Windows operating systems (Windows 95, Windows 98, Windows NT/2000, etc.). It provides all of the
tasks mentioned in Section 3 in the standard manner of windows application.

The simplest way how to describe the user interface is to show some screenshots of the running application. Fig. 4 shows the most important dialog window in which the user can edit rules of the linguistic description (rulebase). Note the indication that there are some duplicate or inconsistent rules.

Fig. 5 contains the “test screen” dialog where the user can test a behavior of the inference on the basis of the designed linguistic description. On the left-hand side the user can interactively enter the crisp inputs to the inference for each input variable. On the bottom of left side he/she can see the input rulebase file; this is ordinary text file.

On the right-hand side of the dialog window, the user can change the defuzzification and inference methods (cf. Section 4.4). Further he/she can see the output fuzzy set of the inference for previously entered crisp inputs together with the defuzzified value and also the number of fired rules in the inference. Last but not least item is the projection over one chosen variable.

6. Interface to other software systems

For communication with other software systems the standard COM object *RBaseCOM* is designed. It encapsulates the core methods of the LFLC package. Definition of its interface follows:

- **LoadFromFile(BSTR FileName)**: Loads rulebase with the specified *FileName*.
- **int NumInputVars()**: Returns the number of input variables from the loaded rulebase.
- **double HiBoundOfVar(int VarIndex)**: Returns high bound of the variable with *VarIndex*.
double LoBoundOfVar(int VarIndex): Returns low bound of the variable with VarIndex.

BSTR VarName(int VarIndex): Returns name of the variable with VarIndex.

double Inference(TVariantInParam Inputs): Performs inference on Inputs values.

The COM Object RBaseCOM is installed and registered into MS Windows system, hence any other application can make use of services provided by it.

A MATLAB S-Function for Simulink package has also been developed, which provides link between LFLC and MATLAB/Simulink software products. It uses the COM Object mentioned above, which runs in the background and the user of MATLAB environment need not worry about it. This S-Function is very useful, because it enables simulation, e.g. of control of various kinds of processes using the LFLC system in MATLAB/Simulink software system.

Besides all of it the MATLAB user can use special MEX-Function (LFLCInfer), which was designed for a use in his/her own MATLAB programs. This function can be called from MATLAB programs, such as any other internal MATLAB functions. It is programmed in C++ and it again uses services of RBaseCOM Object. The function LFLCInfer allows to execute inferences over linguistic descriptions made out in stand-alone application LFLC 2000.

7. What applications can be realized

The concept of LFLC has wide potential for applications. Let us stress that the possibility to use pure linguistic expressions have been appreciated in all cases. The user thinks only linguistic terms, makes all modifications using them only and thus, he/she can forget about fuzzy sets and the necessity to modify them. This feature tremendously increases explicative strength of the linguistic descriptions. Since they are easily understood even long time after the application project is finished, their modification is quite easy.

The previous version, LFLC 1.5, has been used in several real applications, among which the most sophisticated as large scale application of fuzzy control in the Metallurgical plant Břidličná (see [11]). The problem was to control an aluminium smelting furnace. The biggest difficulty for control of the production was discontinuity of the process. This means that the sequence “filling the furnace—smelting—emptying” continuously repeats in regular periods of the length of approximately 8 h. Moreover, the smelting phase is extremely, low with great inertia and it is influenced by the latent heat in negative way. The main phase comes when the temperature of the melted aluminium reaches the technological level (approximately 740–800 °C). The it is important to keep it for about 1 h. The process is highly nonlinear and before using fuzzy control, several other control techniques including adaptive control have been applied with various success. The best success has been obtain using fuzzy control based on LFLC concept.

The applications of fuzzy control in Břidličná started in 1995. After good experiences with the first furnace fuzzy control, it was decided to apply it on the other four furnaces one by one, too. At present, the system works in the whole enterprise, which consist of five furnaces.

Except for pure fuzzy control (several tens of simulations with fuzzy control of various kinds of processes have been realized), LFLC can be applied also in multicriteria decision making. For example, several studies for its application in decision making of the bank concerning creditworthiness of its clients have been prepared. At present, a study for application in control of traffic junction is being elaborated. The results are good and encouraging for the more detailed elaboration.

We would like to stress that the software system LFLC 2000 implements original scientific results and unique methodology systematically developed in our institute, and offers possibilities which are not feasible in other fuzzy logic software systems known to us, including Fuzzy Logic Toolbox of MATLAB. Let us remark that all these systems use only the fuzzy approximation of a function method (cf. Section 2.2) based on the design of the disjunctive normal form. Tuning of the linguistic description is thus based only on modification of the membership functions used for interpretation of linguistic labels which are by no means understood as genuine linguistic expressions.

To the contrary, the LFLC system is designed in such a way that its user works most of the time on linguistic level, i.e. with evaluating linguistic expressions. Only exceptionally it is necessary to change the fuzzy sets—meanings of these expressions. The main
approximate reasoning method is thus the logical deduction described in Section 2.3. For example, when designing the control, the man–machine interaction works as if the computer were a partner who is taught how to realize the control using the natural language in the same way as we explain to somebody how to make this task. Of course, LFLC 2000 allows also the fuzzy approximation of functions in the way implemented in the above mentioned software systems.

It follows from our experiences with the system LFLC, that using one linguistic description, successful control of a wide class of dynamical systems of various orders and characteristics is possible (including the unstable first-order system). The only necessary information which must be supplied to the system is the linguistic context (cf. Section 2.3).

At the same time all significant advantages of fuzzy control are kept, in particular the robustness, i.e. insensitivity with respect to changes of conditions and to errors. The work with linguistic descriptions is simple and intuitive. Potential future changes or adaptations are not difficult, because even after a time it is possible to understand easily what is tile linguistic description doing. This demonstrates a great power and potential of our system, which consistently respects the original idea behind fuzzy logic—to enable the work in natural language and to learn computers to understand it.

8. Conclusion—further development

Further development of the software is supposed in several directions. The main goal is to make possible the design of hierarchical structures, which may consist of various kinds of units, each of them possibly realizing different inference mechanism. It is also necessary to complete the system by the possibility to form Takagi–Sugeno rules (they are already prepared in the kernel of the system).

A very important part with wide potential for applications is learning. Besides the second learning method noticed in Section 3, we suppose to implement learning abilities based on neural nets. Another very powerful technique which is supposed to be implemented is that of fuzz transform developed by Perfilieva [15]. This technique gives very good results in filtering the data and thus, in connection with fuzzy logic deduction it can be applied in sophisticated approximation methods, fitting and prediction of time series and also in solving differential equations (see [10,16]).

Last but not least, we also plan to extend the linguistic power of the system, besides others by generalized (fuzzy) quantifiers, which would thus extend the applicability of the system to summarization of data (see [3]).

References

Antonín Dvorák received the master degree from Technical University Brno (VUT) in control engineering in 1993. Now he is assistant researcher at Institute for Research and Applications of Fuzzy Modeling at University of Ostrava, where he is finishing his PhD in applied mathematics. His research interests include fuzzy logic, approximate reasoning and theoretical semantics.

Hashim Habiballa is a lecturer at Department of Computer Science, University of Ostrava and part-time research assistant at Institute for Research and Applications of Fuzzy Modeling. He holds MSc in computer science and technology and has been attending postgraduate course of pedagogy at University of Ostrava and of distance education at University of Palacky, Olomouc. Now he studies in doctoral program at University of Nitra, Slovak Republic, in theory of computer science education; his research interests include theory of formal languages and automata, logic, programming and didactics of computer science.

Prof. Ing. Vítem Novák, DrSc graduated in Mining University, Ostrava (Czech Republic) as a system engineer. He obtained PhD in mathematical logic in Charles University, Prague and DSc (doctor of sciences) in computer science in Polish Academy of Sciences, Warsaw. He obtained a full professorship in Masaryk University, Brno (awarded by the president of the Czech Republic). He worked in the Mining Institute of the Czechoslovak Academy of Sciences and since 1995 he has been working in the University of Ostrava in Department of Mathematics and at the same time is the director of the Institute for Research and Applications of Fuzzy Modeling. His research interests are fuzzy logic, modeling of natural language semantics, fuzzy control and modeling. He is author or co-author of 4 monographs and more than 140 scientific papers. He is a member of the European Society for Fuzzy Logic and Technology (EUSFLAT) and also a member of the Editorial Board of the International Journal of Fuzzy Sets and Systems, Journal of Soft Computing and Journal of Nonclassical Logics and Applications.

Viktor Pavlíška is a research assistant at Institute for Research and Applications of Fuzzy Modeling, University of Ostrava and associate lecturer at Department of Computer Science. He holds MSc in computer science and technology and now he studies in applied mathematics doctoral program at University of Ostrava; his research interests include mathematics, theory of formal languages and automata, astronomy, physics and programming.