<https://en.wikipedia.org/wiki/Fuzzy_logic>

**Fuzzy logic** is a form of [many-valued logic](https://en.wikipedia.org/wiki/Many-valued_logic) in which the [truth values](https://en.wikipedia.org/wiki/Truth_value) of variables may be any [real number](https://en.wikipedia.org/wiki/Real_number) between 0 and 1 both inclusive. It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false. By contrast, in [Boolean logic](https://en.wikipedia.org/wiki/Boolean_algebra), the truth values of variables may only be the integer values 0 or 1.

The term *fuzzy logic* was introduced with the 1965 proposal of [fuzzy set theory](https://en.wikipedia.org/wiki/Fuzzy_set_theory) by [Lotfi Zadeh](https://en.wikipedia.org/wiki/Lotfi_A._Zadeh" \o "Lotfi A. Zadeh).[[2]](https://en.wikipedia.org/wiki/Fuzzy_logic#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Fuzzy_logic#cite_note-3) Fuzzy logic had, however, been studied since the 1920s, as [infinite-valued logic](https://en.wikipedia.org/wiki/%C5%81ukasiewicz_logic)—notably by [Łukasiewicz](https://en.wikipedia.org/wiki/Jan_%C5%81ukasiewicz" \o "Jan Łukasiewicz) and [Tarski](https://en.wikipedia.org/wiki/Alfred_Tarski).[[4]](https://en.wikipedia.org/wiki/Fuzzy_logic#cite_note-4)

Fuzzy logic is based on the observation that people make decisions based on imprecise and non-numerical information. Fuzzy models or sets are mathematical means of representing vagueness and imprecise information (hence the term fuzzy). These models have the capability of recognizing, representing, manipulating, interpreting, and utilizing data and information that are vague and lack certainty.

Fuzzy logic has been applied to many fields, from [control theory](https://en.wikipedia.org/wiki/Control_theory) to [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence).

In **fuzzy logic**, the truth of any statement becomes a matter of a degree. **Fuzzy inference** is the process of formulating the mapping from a given input to an output using **fuzzy logic**. The mapping then provides a basis from which decisions can be made or patterns discerned.

<https://www.mathworks.com/help/fuzzy/types-of-fuzzy-inference-systems.html>

Fuzzy logic has two different meanings. In a narrow sense, fuzzy logic is a logical system, which is an extension of multivalued logic. However, in a wider sense fuzzy logic (FL) is almost synonymous with the theory of fuzzy sets, a theory which relates to classes of objects with unsharp boundaries in which membership is a matter of degree. In this perspective, fuzzy logic in its narrow sense is a branch of FL. Even in its more narrow definition, fuzzy logic differs both in concept and substance from traditional multivalued logical systems.

***Fuzzy Inference Systems (FIS)*** take inputs and process them based on the prespecified rules to produce the outputs. Both the inputs and outputs are real valued, whereas the internal processing is based on fuzzy rules and fuzzy arithmetic.

In terms of **inference** process there are **two** main **types of fuzzy inference system** (FIS), namely the Mamdani **type** and the TSK (Takagi, Sugeno and Kang) **type**.

In the case of Mamdani FIS the consequent membership functions are also **fuzzy** in nature.

**Mamdani** type **fuzzy** inference gives an output that is a **fuzzy** set. **Sugeno**-type inference gives an output that is either constant or a linear (weighted) mathematical expression.

.

Interpretations, evaluations and reports all depend upon custom NASIS code written in Microsoft’s general-purpose “C#” computer language. This custom code manages data retrieval from the NASIS database and computation of reports, evaluations, interpretations and fuzzy “truth”.

There is a little bit of third-party software used, the “ANTLR” software library. This library is used within the C# code to translate a user’s input (such as the definition of a report) into a data “structure” that drives the custom NASIS C# code. This data structure also drives the final report formatting, also performed by custom NASIS C# code.

<https://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html>

In Fuzzy Logic Toolbox™ software, fuzzy logic should be interpreted as FL, that is, fuzzy logic in its wide sense. The basic ideas underlying FL are explained in [Foundations of Fuzzy Logic](https://www.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html). What might be added is that the basic concept underlying FL is that of a linguistic variable, that is, a variable whose values are words rather than numbers. In effect, much of FL may be viewed as a methodology for computing with words rather than numbers. Although words are inherently less precise than numbers, their use is closer to human intuition. Furthermore, computing with words exploits the tolerance for imprecision and thereby lowers the cost of solution.

Another basic concept in FL, which plays a central role in most of its applications, is that of a fuzzy if-then rule or, simply, fuzzy rule. Although rule-based systems have a long history of use in Artificial Intelligence (AI), what is missing in such systems is a mechanism for dealing with fuzzy consequents and fuzzy antecedents. In fuzzy logic, this mechanism is provided by the calculus of fuzzy rules. The calculus of fuzzy rules serves as a basis for what might be called the Fuzzy Dependency and Command Language (FDCL). Although FDCL is not used explicitly in the toolbox, it is effectively one of its principal constituents. In most of the applications of fuzzy logic, a fuzzy logic solution is, in reality, a translation of a human solution into FDCL.

A trend that is growing in visibility relates to the use of fuzzy logic in combination with neurocomputing and genetic algorithms. More generally, fuzzy logic, neurocomputing, and genetic algorithms may be viewed as the principal constituents of what might be called soft computing. Unlike the traditional, hard computing, soft computing accommodates the imprecision of the real world. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, and low solution cost. In the future, soft computing could play an increasingly important role in the conception and design of systems whose MIQ (Machine IQ) is much higher than that of systems designed by conventional methods.

Among various combinations of methodologies in soft computing, the one that has highest visibility at this juncture is that of fuzzy logic and neurocomputing, leading to neuro-fuzzy systems. Within fuzzy logic, such systems play a particularly important role in the induction of rules from observations. An effective method developed by Dr. Roger Jang for this purpose is called ANFIS (Adaptive Neuro-Fuzzy Inference System). This method is an important component of the toolbox.

Fuzzy logic is all about the relative importance of precision: How important is it to be exactly right when a rough answer will do?

You can use Fuzzy Logic Toolbox software with MATLAB® technical computing software as a tool for solving problems with fuzzy logic. Fuzzy logic is a fascinating area of research because it does a good job of trading off between significance and precision — something that humans have been managing for a very long time.

In this sense, fuzzy logic is both old and new because, although the modern and methodical science of fuzzy logic is still young, the concepts of fuzzy logic relies on age-old skills of human reasoning.

<http://www.mentalmodeler.org/#resources>

**What is Fuzzy-Logic Cognitive Mapping?**

Fuzzy-Logic Cognitive Mapping (FCM) is a parameterized form of concept mapping where you can develop qualitative static models that are translated into semi-quantitative dynamic models. Bart Kosko originally developed FCM in 1986 as a way to structure expert knowledge using a soft systems programming approach that is "fuzzy", thought to be similar to the way that the human mind makes decisions.

FCM represents knowledge by defining three characteristics of a system:

* The components of the system
* The positive or negative relationships between the components
* The degree of influence that one component can have on another, defined using qualitative weightings (e.g. high, medium, or low influence)

The analytical mechanics of FCM are based on examining the structure and function of concept maps, using graph theory-based analyses of pairwise structural relationships between the concepts included in a model. These models can be used to examine perceptions of an environmental or social problem or to model a complex system where uncertainty is high and there is little empirical data available.

**Fuzzy Logic Modeling Tools**

**Fuzzy Logic Toolbox** – uses MatLab and Simulink – Mathworks

Fuzzy Logic Toolbox™ provides MATLAB® functions, apps, and a Simulink® block for analyzing, designing, and simulating systems based on fuzzy logic. The product guides you through the steps of designing fuzzy inference systems. Functions are provided for many common methods, including fuzzy clustering and adaptive neurofuzzy learning.

The toolbox lets you model complex system behaviors using simple logic rules, and then implement these rules in a fuzzy inference system. You can use it as a stand-alone fuzzy inference engine. Alternatively, you can use fuzzy inference blocks in Simulink and simulate the fuzzy systems within a comprehensive model of the entire dynamic system.

**Fulsome** (FUzzy Logic for SOftware MEtrics).

**Abstract:**

There has been a growing body of literature suggesting that some of the problems faced by software development project managers can be at least partially overcome by using fuzzy logic techniques. However, one issue that has been generally overlooked in this recommendation is the means by which these "software metricians" can collect data for, develop, and interpret fuzzy logic models in practice.

**Mental Modeler**

Based in Fuzzy-logic Cognitive Mapping (FCM), users can easily develop semi-quantitative models of environmental issues, social concerns or social-ecological systems in *Mental Modeler* by:

* Defining the important components of a system
* Defining the relationships between these components
* Running "what if" scenarios to determine how the system might react under a range of possible changes.
* ***Mental Modeler*** allows you to build Fuzzy-logic Cognitive Maps easily and intuitively. Once models are built, increasing or decreasing the components included in the model allows you to examine different scenarios of change. Because of their flexibility, FCM have been used in a range of scientific disciplines, from political science to economics to ecology.

<https://pdfs.semanticscholar.org/e873/3a63c5918b6aba762a0ce0df2444383d0b62.pdf>

In this paper was analyzed the following tools: FuzzyF, Fuzzy Logic ToolBox, Mathematica Fuzzy Logic, FIDE, TILShell, FuzzyTECH, RockOn Fuzzy Tool, SciLab Fuzzy Fuzzy Tool, UNFUZZY e XFuzzy. The choice of the software mentioned is due to be available for download on the Internet and its uses in projects published [2]. The positive, negative and differential points for these software’s are presented

**Open source SourceForce**

<https://sourceforge.net/projects/fuzzylogictools/>

Fuzzy Logic Tools (FLT) is a C++ framework for storage, analysis and design of fully general multiple-input multiple-output (MIMO) Takagi-Sugeno fuzzy control systems, without constraints in the order of either the inputs or the output vectors.

<https://irafm.osu.cz/en/c49_0/>

**LFLC 2000 (Linguistic Fuzzy Logic Controller)** is specialized software, which is based on deep results obtained in formal theory of fuzzy logic. It makes it possible to deduce conclusions on the basis of imprecise description of the given situation using *fuzzy IF-THEN rules*. The rules are interpreted either as fuzzy relations, or they can be taken as genuine linguistic expressions and interpreted using the original theory developed in IRAFM.Sets of linguistically interpreted fuzzy IF-THEN rules are called *linguistic descriptions*. They can be understood as specific text describing the given process, decision, or classification situation. The users may thus work only with expressions of natural language without necessity to think how they are implemented. Hence, the computer behaves as if “partner” which understands the language of human user.  
LFLC 2000 is written in C++ under Windows and it is fully object oriented system. The system is integrated with other software LFLCSim using which we can simulate simple control in closed feedback loop. LFLC2000 is also joined with MATLAB/Simulink so that simulation of wide class of systems is possible.

For theoretical background of LFLC, the following publications are recommended: detailed treatment of fuzzy logic can be found in [1]; the theory of trichotomous evaluative linguistic expressions is in detail explained in [2]; the theory of perception–based logical deduction can be found in [3], [4], [5]. For explanation of representation of fuzzy logic functions by normal forms see [6].

**The purpose of LFLC**

The main purpose of LFLC 2000 is to design, test and learn *linguistic descriptions*, i.e. sets of fuzzy IF–THEN rules. The descriptions can be further used in control, decision support and other applications. We can distinguish the following main tasks realized by LFLC:

**Design of linguistic descriptions:**LFLC makes possible to use various pre-defined evaluative linguistic expressions (e.g., *small*, *about 5*, *more or less medium* etc.) It has also means for an analysis of several properties of linguistic descriptions (sorting, detection of identical, inconsistent or redundant IF–THEN rules etc.). In addition to standard linguistic descriptions [4], it is also possible to use IF–THEN rules of Takagi–Sugeno type with consequents specified as singletons.

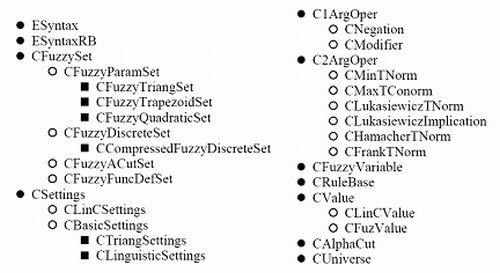
**Design and modification of user expressions:**In addition to pre-defined linguistic expressions users can also define and use their own expressions in situations when standard ones are for some reasons insufficient.

**Testing of inference over designed linguistic descriptions:**LFLC allows users to visualize the behavior of a linguistic description for various (crisp) observations. He/she can select various inference and defuzzification methods (see [4]) and look at projections with respect to individual variables or three-dimensional control surface. There is also information about IF–THEN rules fired and the 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 evaluative linguistic expression corresponding to the given value. The resulting linguistic description should be used for finding conclusions using perception-based logical deduction [4].

The second possibility is based on theoretical results published in [6], [7] and it enables to find a linguistic description interpreted by disjunctive or conjunctive normal form [6] with the prescribed accuracy of approximation of the data understood as specification of some function.Both fuzzy approximation techniques are constructed to deal with fuzzy numbers or fuzzy sets which partition the respective domain in such a way that their kernels do not overlap each other. Therefore, the learning can be designed similarly to the linguistic one after specifying the set of possible fuzzy sets for each variable. They are specified by so called *user expressions*, which are nonlinguistic fuzzy sets on respective domains. This gives a user an opportunity to make several decisions about accuracy and/or complexity on each axis. **Important data structures and algorithms**

The LFLC software system is implemented in C++ programming language with the 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.  
  
Next figure shows the hierarchy of the main classes of LFLC. There are classes for the representation of fuzzy sets (CFuzzySet and its derivatives), classes which allow assignments of fuzzy sets to linguistic expressions (CLinguisticSettings), classes for the 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 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 a class. These classes CFuzzySet, CLinguisticSettings and CRuleBase are not, however, on the same level of generality. Instances of CFuzzySet are members of CLinguisticSettings and, similarly,  
instances of CLinguisticSettings are members of CRuleBase.

----------------------------------------------------------------------------

**Fuzzy rules** are used within [fuzzy logic systems](https://en.wikipedia.org/wiki/Fuzzy_logic) to infer an output based on input variables. [Modus ponens](https://en.wikipedia.org/wiki/Modus_ponens) and [modus tollens](https://en.wikipedia.org/wiki/Modus_tollens) are the most important rules of inference.[[1]](https://en.wikipedia.org/wiki/Fuzzy_rule#cite_note-1) A modus ponens rule is in the form

Premise: *x is A*

Implication: ***IF****x is A****THEN****y is B*

Consequent: *y is B*

In crisp logic, the premise *x is A* can only be true or false. However, in a fuzzy rule, the premise *x is A* and the consequent *y is B* can be true to a degree, instead of entirely true or entirely false.[[2]](https://en.wikipedia.org/wiki/Fuzzy_rule#cite_note-:0-2) This is achieved by representing the linguistic variables *A* and *B* using [fuzzy sets](https://en.wikipedia.org/wiki/Fuzzy_set).[[2]](https://en.wikipedia.org/wiki/Fuzzy_rule#cite_note-:0-2) In a fuzzy rule, modus ponens is extended to *generalised modus ponens:.*[[2]](https://en.wikipedia.org/wiki/Fuzzy_rule#cite_note-:0-2)

Premise: *x is A*\*

Implication: ***IF****x is A****THEN****y is B*

Consequent: *y is B*\*

The key difference is that the premise *x is A* can be only partially true. As a result, the consequent *y is B* is also partially true. Truth is represented as a [real number](https://en.wikipedia.org/wiki/Real_number) between 0 and 1, where 0 is false and 1 is true.