

XFUZZY 3.0: A Development Environment for Fuzzy Systems

F. J. Moreno Velo I. Baturone S. Sánchez Solano A. Barriga

Instituto de Microelectrónica de Sevilla
 Centro Nacional de Microelectrónica - CSIC
 Avda. Reina Mercedes, s/n. Edif. CICA, E-41012, Sevilla, Spain
 xfuzzy-team@imse.cnm.es

<http://www.imse.cnm.es/xfuzzy>

Abstract

This paper presents the new version of *Xfuzzy*, *Xfuzzy* 3.0, which is a development environment for fuzzy-inference-based systems. It is composed by many tools that cover the different stages of the fuzzy system design process, from their initial description to the final implementation. Its main features are the capability for developing complex systems, the flexibility of allowing the user to extend the set of available functions and the possibility of been executed on any platform with JRE (Java Runtime Environment) installed.

Keywords: CAD Tools, Fuzzy System Development, Fuzzy System Tuning, Fuzzy Circuits.

1. Introduction

Fuzzy systems have become an important alternative in many fields of application, thus increasing the necessity of easing their development [4][5]. In the last few years several CAD tools tailored to the fuzzy system design have been created. Nevertheless, many of these tools are dedicated to specific realizations and have constraints on the set of fuzzy operations they support, the complexity of the systems they can develop and their capability for automatic tuning, simulation or synthesis.

Our research group has been working these years on the development of general purpose CAD tools which could solve these limitations. This led us to create *Xfuzzy* 2.0 [1], an environment which integrates different tools for edition, tuning, simulation and synthesis of fuzzy systems. These tools share a formal specification language, called *XFL* [2], which deals with the definition of complex systems.

The new version of the environment attempts to dive into this line of work. For this purpose, new tools covering distinct aspects of the design process have been created and a more homogeneous graphical interface has been defined. Moreover, the specification language

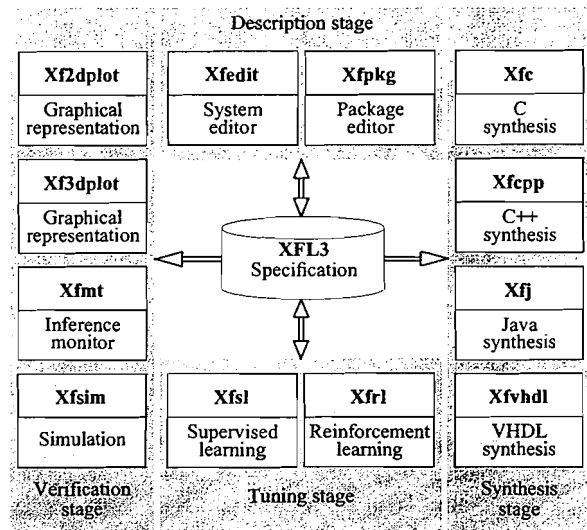


Figure 1: Design flow using *Xfuzzy* 3.0. (Shaded rectangles refer to tools under development)

has been extended. The new language, called *XFL3* [3], incorporates linguistic hedges and allows extending the set of available fuzzy functions. The environment has been completely programmed in Java, so it can be executed on any equipment with JRE (Java Runtime Environment) installed.

2. The Xfuzzy 3.0 environment

Xfuzzy 3.0 is a development environment for fuzzy systems which integrates several tools covering the different stages of the design (Fig. 1). The description stage includes graphical tools for the definition of systems. The tuning stage consists in applying learning algorithms. The verification stage is composed of tools for simulation, monitorization and graphical representations of the system behavior. Finally, the synthesis stage is divided into tools generating high-level languages descriptions for software or hardware implementations.

The nexus between all these tools is the use of a common specification language, *XFL3*, which extends the capabilities of *XFL*, the language defined in version

2.0. XFL3 is a flexible and powerful language, which allows to express very complex relations between the fuzzy variables, by means of hierarchical rule bases and user-defined fuzzy connectives, linguistic hedges, membership functions and defuzzification methods.

Every tool can be executed as an independent program. The environment integrates all of them under a graphical user interface which eases the design process (Fig. 2).

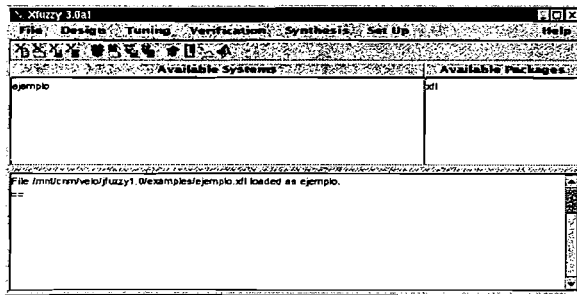


Figure 2: Main window of *Xfuzzy 3.0*.

3. Description stage

The first step in the development of a fuzzy system is to select a preliminary description of the system. This description will be refined later as a result of the tuning and verification stages.

Xfuzzy 3.0 contains two tools assisting in the description of fuzzy systems: *xfedit* and *xfpkg*. The first one is dedicated to the logical definition of the system, that is, the definition of its linguistic variables and the logical relations between them. On the other side, the *xfpkg* tool eases the description of the mathematical functions assigned to the fuzzy operators, linguistic hedges, membership functions and defuzzification methods.

The description of a fuzzy system in XFL3 language consists of several objects defining operator sets, linguistic variable types and rule bases, as well as a

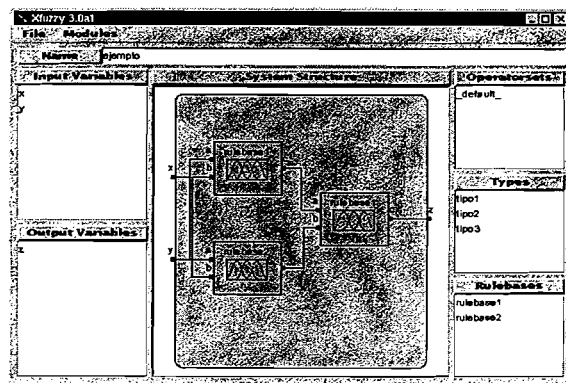


Figure 3: Main window of *xfedit*

description of its hierarchical structure. The *xfedit* tool offers a graphical interface to ease the description of systems, avoiding the need for a deep knowledge of the XFL3 language. The system hierarchy is shown on the main window of *xfedit* (Fig. 3). From this window, other edition windows for operator sets, variable types and rule bases can be displayed.

The edition of an operator set is carried out by selecting the mathematical functions assigned to the distinct logical connectives, rule implication and aggregation operators, linguistic hedges and defuzzification methods (Fig. 4).

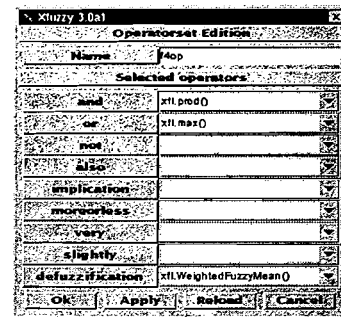


Figure 4: Operatorset edition on *xfedit*.

The definition of a linguistic variable type includes the description of its universe of discourse, of the labels applied to the variable and the membership functions assigned to those labels. The type edition window (Fig. 5) allows the introduction of the minimum/maximum values and the cardinality, which defines respectively the limits and the number of elements of the universe of discourse. Moreover, the window eases the definition of the linguistic labels and their membership functions, showing graphically the distribution of these functions on the universe of discourse and giving facilities for modifying their parameters.

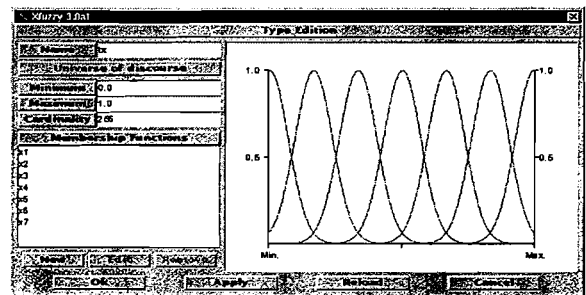


Figure 5: Type edition on *xfedit*.

The rule bases contain the expressions of the logical relations between the linguistic variables of the system. The XFL3 language can describe very complex relations between the variables by combining basic propo-

sitions (which compares the value of a variable with a linguistic label) via connectives and linguistic hedges.

The window for editing rule bases on *xfedit* (Fig. 6) presents three formats for the rule definition: matrix form, table form and free form. The matrix form constricts the rule base to two inputs and one output, and shows the rules in a compact form as a matrix. Each matrix element represents a rule like «if a is A & b is B then c is C». The table form is valid for any number of inputs and outputs, and each element of the table represents a rule like «if x0 is X0 & x1 is X1 & ... & xn is XN then z is Z». Finally, the free form allows exploiting the whole power of XFL3 to define complex relations like «if x0 is greater than X0 or not x3 is strongly equal to X3 then z is Z».

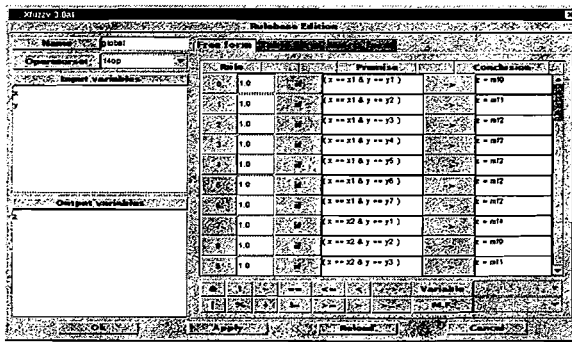


Figure 6: Rulebase edition on *xfedit*.

One of the main features of the Xfuzzy 3.0 environment is the possibility of extending the set of mathematical functions that can be assigned to the different fuzzy connectives, linguistic hedges, rule aggregation, implication operators, available membership functions, and defuzzification methods. These mathematical functions are defined in files named as *packages*. The *xfpkg* tool is dedicated to the edition of function packages, which can be used by all the environment tools (Fig. 7).

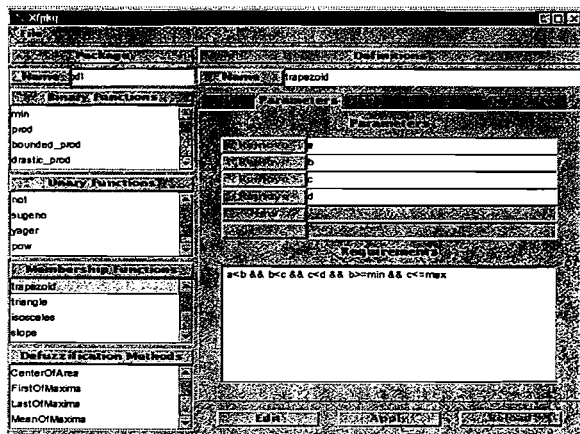


Figure 7: Main window of *xfpkg*.

The definition of a new mathematical function requires the introduction of the information needed by any tool of the environment. This includes the definition of the function parameters and their constraints, the description of the behavior in the high-level languages Java, C y C++ (needed by the software synthesis) and the description of the differential function (required by some learning algorithms of the tuning stage).

4. Tuning stage

Tuning of fuzzy system behavior usually becomes is always one of the most complex tasks of the design process. This behavior depends on the logical structure of the rule bases and on the definition of the membership functions employed for the linguistic variable types. Apart from unforeseen redesigns, the tuning process often deals with the modification of the parameters defining the membership functions. Since this should be performed simultaneously over a great number of parameters, a manual procedure happens to be an extremely complex task, making it necessary the use of automated tuning techniques.

Xfuzzy 3.0 environment currently contains one tool dedicated to the tuning stage: *xfsl*, which is based on the use of supervised learning algorithms (Fig. 8). Another tool dedicated to reinforcement learning procedures is under development. Both tools intend the system behavior to approximate a known behavior. In supervised learning, a set of desired input/output patterns is available, which defines an error function between the system behavior and the desired one. Supervised learning attempts to minimize this error function. When resorting to the use of reinforcement learning, the exact values of system outputs are unknown. Reinforcement learning is based on studying the on-line behavior of the system to discover the effect the system must produce on its operational environment.

The *xfsl* tool includes gradient descent algorithms, conjugate gradient algorithms, quasi-Newton algorithms, descent without derivatives, and stochastic

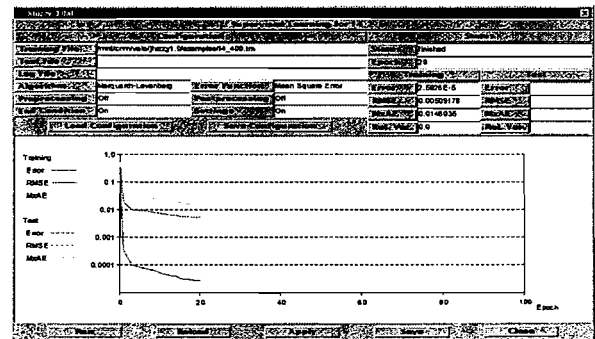


Figure 8: Main window of *xfsl*.

methods like simulated annealing. The tool admits different error functions and allows the selection of the parameters to be tuned. Besides, *xfsl* includes two methods of pre- and post-processing to simplify the fuzzy system. The first one consists on eliminating non significant rules and labels. The second one develop a clustering over the labels of the output variables.

5. Verification stage

The objective of the verification stage is to study the behavior of the system under development, detecting probable deviations on the expected behavior and identifying the source of these deviations.

Xfuzzy 3.0 environment currently contains two verification tools: *xf2dplot* and *xf3dplot*, which are dedicated to graphically represent the system behavior in two (*xf2dplot*) or three dimensions (*xf3dplot*). Both tools work in a similar way: the input and output variables to be represented must be selected while values of the non-represented variables must be assigned (Fig. 9).

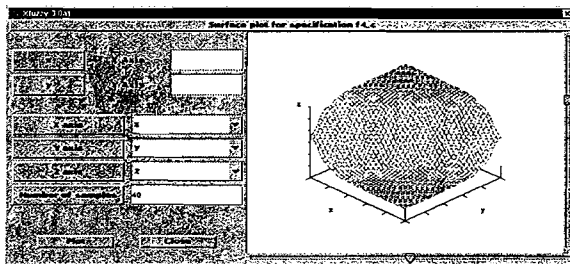


Figure 9: Main window of *xf3dplot*.

Other tools under development are *xfmt* and *xfsim*. The former attempts to monitorize the system, showing the activation degree of every linguistic label and logical rule, as well as the value of the different inner variables, for some determined input values. The *xfsim* tool is aimed at simulating the system within its actual or modeled operational environment. It allows illustrating the system evolution by means of graphical representations of user-selected variables.

6. Synthesis stage

The final stage of a fuzzy system design is to generate a system implementation, either software or hardware. Selecting the proper solution depends on the complexity of the system and the imposed requirements. When inference speed, area and power requirements are not restrictive, software implementations are preferred to offer a great flexibility and the possibility of incorporating the system into a more general software project. Xfuzzy 3.0 environment provides the user with three tools for software synthesis: *xfc*, *xfcpp*, *xfj*,

which generate descriptions of the fuzzy systems in C, C++ and Java high-level languages, respectively.

On the other side, when system requirements are restrictive, hardware implementations are more adequate. To aid the designer in this task, another tool which is currently under development is *xfvhdl*. The purpose of this tool is to generate an VHDL description of the system, able to be synthesized as an ASIC (Application Specific Integrated Circuit) or an FPGA (Field Programmable Gate Array).

7. Conclusions

The version 3.0 of Xfuzzy presented in this paper is a complete environment for the design of fuzzy systems. Its tools (some of them under development) covers efficiently the different stages of the design process. These tools share a common formal description language which allows the definition of complex systems by means of hierarchical rule bases and user-configurable membership functions, defuzzification methods, fuzzy connectives, linguistic hedges and implication and aggregation operators. The environment is distributed freely under the GNU General Public License and, since it has been programmed entirely in Java, it can be executed on any platform containing the Java Runtime Environment (JRE).

References

- [1] López, D.R., Jiménez, C.J., Baturone, I., Barriga, A., Sánchez-Solano, S.. "Xfuzzy: A Design Environment for Fuzzy Systems", Proc. 7th IEEE Int. Conf. on Fuzzy Systems (FUZZIEEE'98), pp. 1060-1065, Anchorage, May 1998.
- [2] López, D.R., Moreno, F.J., Barriga, A., Sánchez-Solano, S.. "XFL: A Language for the Definition of Fuzzy Systems", Proc. 6th IEEE Int. Conf. on Fuzzy Systems (FUZZIEEE'97), pp. 1585-1591, Barcelona, Jul., 1997
- [3] Moreno, F.J., Sánchez-Solano, S., Barriga, A., Baturone, I., López, D.R., "XFL3: Un nuevo lenguaje de especificación de sistemas difusos", Actas X Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF'2000), pp.509-514, Sevilla, 2000.
- [4] Munakata, T., Jani, Y., Fuzzy systems: an overview, Communications of the ACM, Vol. 37, N. 3, 1994.
- [5] Sugeno, M., Ed., Industrial Application of Fuzzy Control, North-Holland, pp. 19-40, 1985.