A CAD Approach to Simplify Fuzzy System Descriptions

Iluminada Baturone, Francisco J. Moreno-Velo, and Andrés Gersnoviez

***Abstract*—Simplification is an important step in the design of a fuzzy system since the membership functions that represent the fuzzy sets as well as the 'if-then' rules that relate them usually contain redundant information. This paper presents a CAD tool which provides the user with a wide set of algorithms to automate simplification process. It allows reducing the number of membership functions and rules described initially as well as increasing its expressiveness and linguistic interpretability. Since the tool is included within the design environment Xfuzzy 3, the simplified system can be also verified, tuned and synthesized automatically. Several examples are included to illustrate the efficiency of the simplification facilities provided.**

I. Introduction

THE original way of describing a fuzzy system is to translate heuristic and uncertain knowledge expressed linguistically. Another way, which was developed later, is to extract the fuzzy system description from a set of numerical data that specify its behavior. In both cases (and their possible combinations) the membership functions that represent the fuzzy sets as well as the 'if-then' rules that relate them can be usually simplified to obtain a similar system with less and simpler membership functions and rules. The simplification process is very interesting not only to ease the hardware or software implementation of the fuzzy system but also to ensure and/or increase its linguistic meaning.

Several approaches have been proposed in the literature to address this simplification process. One of them is the use of linguistic hedges, whose interest is pointed out by many authors, from early works like [1] to recent ones like [2]. They allow reducing the number of membership functions (many of them can be obtained as modifications of other ones) as well as the rules (some of them can be combined into a more generic one). In addition, the resulting fuzzy system is clearly more transparent and interpretable.

Other authors focus the simplification process on the membership functions [3]. Functions which are similar are iteratively searched to be replaced by the same function. The number of membership functions is reduced not only by this merging process but also by eliminating those functions which are too similar to the universal set. A by-product is that the number of rules can be also reduced because several rules may become redundant.

Another approach is to simplify the rules directly. This is done in [4]-[5] by selecting the most significant rules of the rule base, and in [6] by applying minimization algorithms inspired in the Boolean design. This simplification usually reduces in turn the number of membership functions because some of them become unused or are merged.

Depending on the problem to be solved, one of the above commented approaches can be more efficient than the others when designing the fuzzy system. Even a successive application of several approaches could provide better results. This is why our work has been to include all of them together with some methods developed by the authors into a CAD tool, named *xfsp*, that allows automating the simplification process. Since simplification is one of the design steps of a fuzzy system, this CAD tool has been integrated within the design environment Xfuzzy 3 developed at the Instituto de Microelectrónica de Sevilla, which contains other CAD tools to cover the other design steps (description, verification, tuning, identification, and synthesis) [7]. To the best of our knowledge, no other fuzzy software provides so many facilities to design a fuzzy system and, in particular, to simplify its description.

This paper is organized as follows. Section II explains briefly how the simplification process can be applied within the design methodology of a fuzzy system with the aid of Xfuzzy 3. Section III describes the simplification tool *xfsp*, showing the methods it provides to simplify automatically either membership functions or rule bases. Two examples are included in Section IV to illustrate the efficiency of this tool: one of them describes the simplification of a fuzzy system obtained from a set of numerical data while the other concerns with the simplification of a system described from translating heuristic knowledge. Finally, conclusions are given in Section V.

II. Design Methodology with Xfuzzy 3

The design methodology that can be followed with the aid of Xfuzzy 3 is shown in Fig. 1. The aim of the first stage (*description*) is to describe the whole fuzzy system. This is done by specifying: (a) the membership functions employed to describe the fuzzy sets, (b) the rule bases and mathematical modules involved, (c) the fuzzy operators (used to perform antecedent connection, implication, defuzzification, etc.), and (d) the structure of the system (its inputs and outputs and how the rule bases are

This work was supported in part by the Spanish CICYT Projects DPI2005-02293 and TEC2005-04359. The third author is supported by the Spanish Minister of Education under the program F.P.U. for PhD. students.

I. Baturone and A. Gersnoviez are with the Instituto de Microelectrónica de Sevilla (IMSE-CNM) and the Dept. de Electrónica y Electromagnetismo, Univ. de Sevilla, Seville, SPAIN (phone: +34-955-056-666; fax: +34-955-056-686; e-mail: {lumi, andres}@imse.cnm.es).

F. J. Moreno Velo is with the DIESIA, Esc. Politécnica Superior, Univ. de Huelva, Huelva, SPAIN (e-mail: francisco.moreno@diesia.uhu.es).

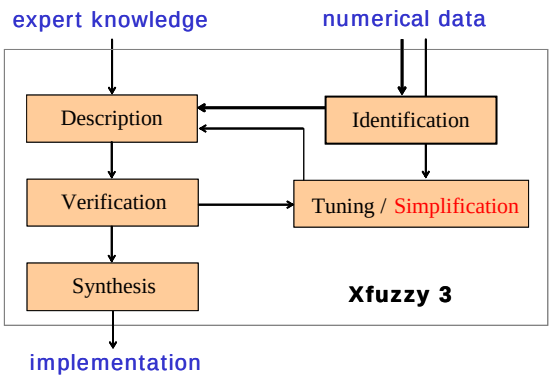

Fig. 1. Design flow aided by Xfuzzy 3.

interconnected).

The formal specification language XFL3 has been defined for the Xfuzzy 3 environment to ease this description process. It allows connecting several antecedents in the rules by any kind of conjunctive and disjunctive connectives, input variables can be related with fuzzy sets by any kind of linguistic hedges, and linguistic hedges can be applied even to some connected antecedents (Table I and Fig. 2). In addition, XFL3 allows the assignment of weights to the rules so that the confidence of the expert on some rules or the relative importance between rules can be also expressed [8]. Rules can be described in a free form by exploiting all these facilities provided by XFL3. An example is the following:

‘if(x1 < E & (x2 != C | x3 >= B)) -> y = D’ (1)

Other format to describe the rules is the tabular form in which the antecedents are connected only by conjunctive operator and no linguistic hedge is employed. For example:

TABLE I
EXAMPLE OF FUZZY PROPOSITIONS EXPRESSED BY XFL3

Basic propositions	**Description**
variable == fuzzy set	equal to
variable >= fuzzy set	equal or greater than (Fig. 2a)
variable <= fuzzy set	equal or smaller than (Fig. 2b)
variable > fuzzy set	greater than (Fig. 2c)
variable < fuzzy set	smaller than (Fig. 2d)
variable != fuzzy set	not equal to (Fig. 2e)
variable %= fuzzy set	slightly equal to (Fig. 2f)
variable ~= fuzzy set	more or less equal to (Fig. 2g)
variable += fuzzy set	strongly equal to (Fig. 2h)
Complex propositions	**Description**
proposition & proposition	and operator
proposition | proposition	or operator
!proposition	not operator
% proposition	slightly operator
~proposition	moreorless operator
+proposition	strongly operator

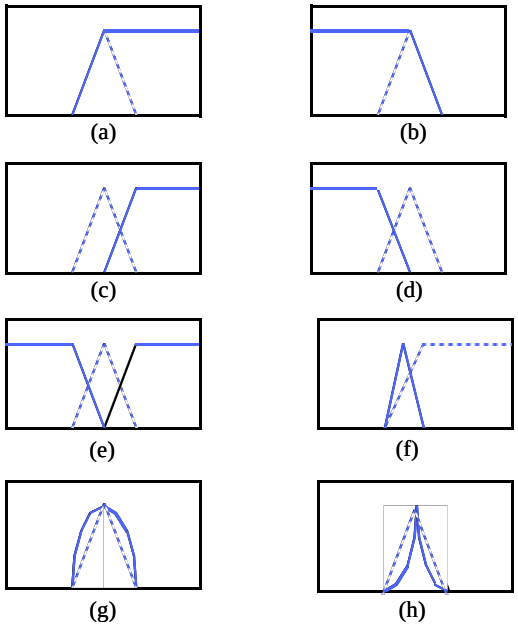

Fig. 2. Illustrating linguistic hedges on membership functions.

‘if(x1 == A & x2 == B & x3 == C) -> y = D’ (2)

The free format is more expressive than the tabular one and contains fewer rules because a rule of the free format usually summarizes several rules of the tabular format. In the above examples, if the fuzzy sets covering the variables x1, x2 and x3 are {A, B, C, D, E}, the rule in (1) would summarize 32 rules in the tabular form, being one of them the rule in (2). The free format is usually employed when the rule base is described from translating expert knowledge expressed linguistically. In the other side, the rules extracted from numerical data are usually expressed in the tabular format.

The Xfuzzy identification tool extracts the rules in the tabular format [9]. It can currently apply five algorithms based on clustering and four algorithms based on grid techniques. The first one organizes the data into clusters and uses them to create the rules and membership functions simultaneously [1], [10]. An advantage of these techniques is that the number of rules extracted is usually low. As a drawback, each rule has its own fuzzy sets which may be similar and complicates their linguistic meaning. As an example, Fig. 3 shows the membership functions obtained by applying Fuzzy C-Means clustering when extracting 11 fuzzy rules from a set of numerical data corresponding to the function $(1+\sin(2\pi x)\cos(2\pi y))/2$. In the other side, the identification algorithms based on grid techniques generate a partition or grid of the input and output spaces prior to creating the rule base [11]. They extract more rules but with the advantage of being more interpretable.

Once the whole system has been described, its behavior should be tested at the verification stage. Xfuzzy 3 contains

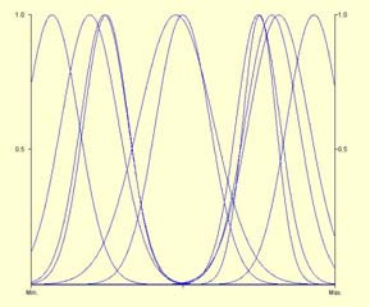

Fig. 3. Membership functions obtained for an input variable with a clustering-based identification algorithm.

three tools to facilitate this verification process. One of them allows to show two- and three-dimensional graphics with the input/output behavior of the fuzzy system. Another tool allows monitoring the values of the internal and global variables of the system and the activation degrees of the rules of the different modules. The last tool simulates the behavior of the fuzzy system working in line with a model of an external system (a plant in the case of a controller). The user can employ these tools to modify the fuzzy system description manually. An automatic way to modify the parameters of the membership functions is to apply supervised learning techniques with the tuning tool of Xfuzzy 3. As happens in the identification process, it is usual that after the tuning process several membership functions become similar, thus complicating the linguistic meaning of the system. As an example, Fig. 4 shows the 49 membership functions learnt for the output variable of a fuzzy system with 49 rules when trying to approximate the function $(1+\sin(2\pi x)\cos(2\pi y))/2$.

After verification and (if applied) tuning stages, the simplification process usually performs a relevant action. Concerning membership functions, it is interesting: (a) to use a minimum number of relevant functions which could be modified by linguistic hedges (as shown in Fig. 2), (b) to merge similar functions resulting from identification and/or tuning stages (as illustrated in Fig. 3 and 4), and (c) to purge those functions which are not used (as a consequence of previous simplifications or a non careful manual description). Concerning rules, it is interesting: (a) to use a compact free format instead of a expanded tabular one (as

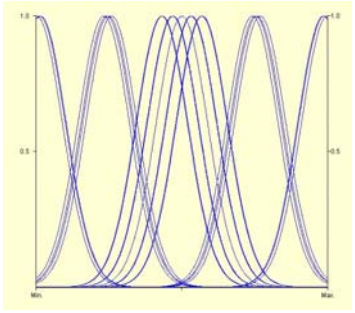

Fig. 4. Membership functions obtained for an output variable with a supervised learning algorithm.

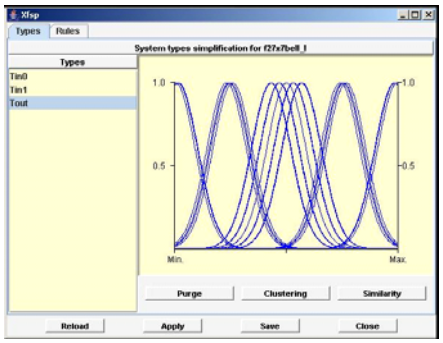

Fig. 5. Main window of the tool *xfsp* for membership function simplification.

illustrated with expressions (1) and (2)) and (b) to use a minimum number of relevant rules by purging those which are not significant. All these simplification processes can be applied automatically by the simplification tool of Xfuzzy 3, as described in the following section.

III. The Simplification Tool Xfsp

The tool *xfsp* of Xfuzzy 3 allows applying simplification algorithms to either the variable membership functions or the rule bases of a fuzzy system.

A. Simplification of membership functions

If the membership functions ("types" in the nomenclature of Xfuzzy 3) are selected in the main window of *xfsp*, the variables defined for the fuzzy system under design appear at the left part of the window. The membership functions describing the selected variable appear at the right part together with three buttons corresponding to the three simplification processes which can be applied to them: purge mechanism, clustering and similarity-based merging method (Fig. 5).

The purge mechanism looks for those membership functions which are not used in any rule base and eliminates them. This may happen not only as a consequence of previous simplification processes but also when the fuzzy system has been defined from translating heuristic knowledge.

The clustering method looks for a reduced number of clusters (membership function prototypes) into which several original functions are grouped. It follows the Hard C-Means algorithm, that is, the clusters found are crisp because each original membership function will be replaced by only one of the resulting prototypes. The clusters are made on the space formed by the parameters that define the membership functions and weights can be applied to them. For example, Gaussian functions are defined by their centers and widths, but if the weight of the widths is zero, only the centers are considered in the clustering process. The optimal number of membership function prototypes can be found automatically by applying validity indexes. Dunn Separation Index, Davies-Bouldin Index, and Generalized Dunn Indexes can be applied with the tool *xfsp* [12]. Considering

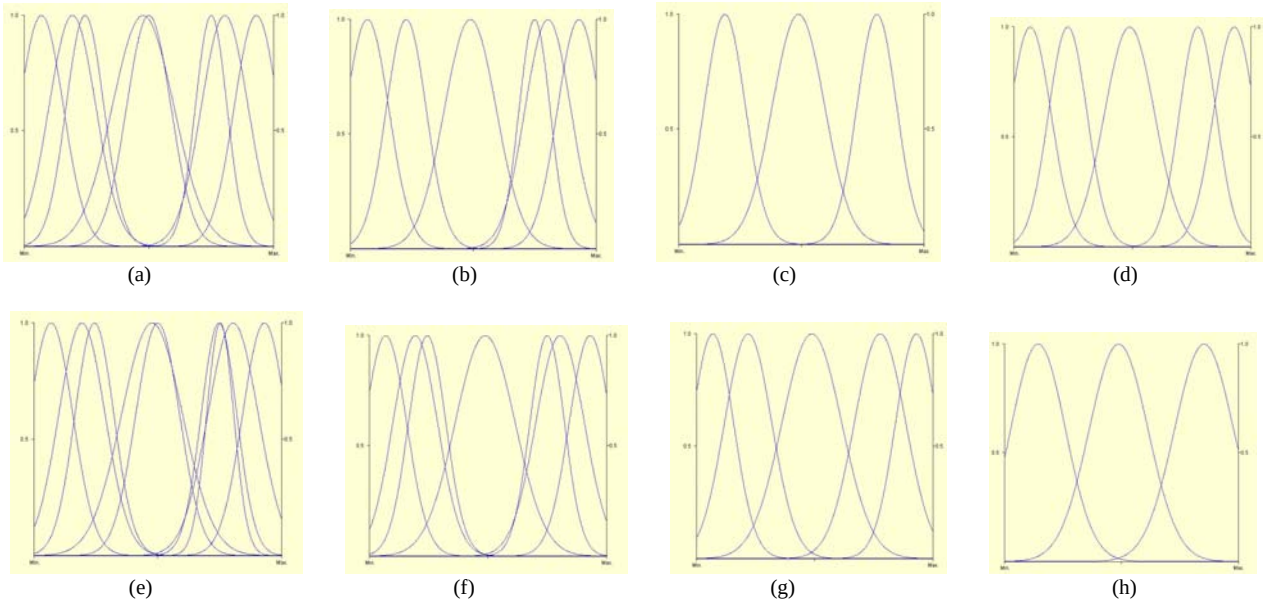

Fig. 6 Simplification results on the functions in Fig. 3: clustering with (a) the Separation Index, (b) Davies-Bouldin Index, (c) Generalized Dunn Index, and (d) manual index; merging with a similarity factor over (e) 0.8, (f) 0.7, (g) 0.5, and (h) 0.2.

the 11 functions illustrated in Fig. 3, a clustering process that applies the above indexes results, respectively, in the 8, 6, and 3 functions shown in Fig. 6a-c. Such direct validity indexes are very useful for clustering problems in which visual inspection is not possible. In this problem, however, the tool *xfsp* provides the user with the graphical display of the original and grouped functions. Hence, *xfsp* also offers the option of fixing the number of clusters after the visual inspection of the original functions. This is usually the best option for giving more freedom to the user. For example, if the user decides to cluster the 11 functions of Fig. 3 into 5 clusters, the result is that in Fig. 6d.

The other way to simplify membership functions with *xfsp* is to apply a similarity-based merging process. This process looks for the pair of most similar functions iteratively and replaces them by a unique function if the similarity degree is over a threshold defined by the user. The process finishes when no more functions can be merged. The lower the threshold the more functions is merged as illustrated in Fig. 6 e-h. The similarity measure used by *xfsp* is the one defined by Dubois and Prade as follows [13]:

$$S(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\sum_{j=1}^{m} \left[\mu_A(x_j) \wedge \mu_B(x_j)\right]}{\sum_{j=1}^{m} \left[\mu_A(x_j) \vee \mu_B(x_j)\right]} \tag{3}$$

where A and B are two fuzzy sets described by the membership functions μ_A and μ_B and defined on a discrete universe of discourse formed by *m* x_j points.

If the functions μ_A and μ_B to be merged are defined by 4 parameters (like a trapezoidal function) as follows:

$$\mu_A(x;a_1,a_2,a_3,a_4) = \begin{cases} 0, & x \le a_1 \quad o \quad x \ge a_4 \\ 1, & a_2 \le x \le a_3 \\ \alpha, & \alpha \in (0,1) \quad \text{otherwise} \end{cases} \tag{4}$$

xfsp replaces them by another membership function $\mu_C(x; c_1, c_2, c_3, c_4)$ where:

$$\begin{aligned} c_1 &= \min(a_1, b_1) \\ c_2 &= (a_2 + b_2)/2 \\ c_3 &= (a_3 + b_3)/2 \\ c_4 &= \max(a_4, b_4) \end{aligned} \tag{5}$$

The merging of triangular and rectangular functions is similar. If the membership functions are Gaussians, as follows:

$$\mu_A(x;a_1,a_2) = \exp\left[(a_1 - x)^2 / a_2\right] \tag{6}$$

xfsp replaces them by another Gaussian function $\mu_C(x; c_1, c_2)$ where:

$$\begin{aligned} c_1 &= (a_1 + b_1)/2 \\ c_2 &= |a_2 - b_2|/6 + \max(a_2, b_2) \end{aligned} \tag{7}$$

For other types of membership functions or when the functions to merge are different *xfsp* does not decide by default the resulting function but asks the user to define it through a graphical interface.

The results of applying similarity-based simplification are similar to those of applying clustering, as can be seen in Fig. 6d and g, and Fig. 6c and h. Advantages of using similarity-based method are that functions of different types (a triangle

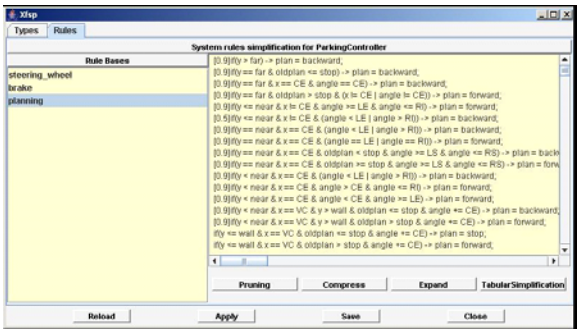

Fig. 7. Main window of the tool *xfsp* for rule base simplification.

with a Gaussian, for instance) can be merged (which is not possible with clustering) and that the use of a threshold value can be more intuitive for the user. As a drawback, its computational cost is higher because the universe of discourse has to be swept, although this cost is significant only for very complex systems.

B. Simplification of rule bases

If the rules are selected in the main window of *xfsp*, the rule bases defined for the fuzzy system under design appear at the left part of the window. The set of rules describing the selected rule base appear at the right part together with four buttons corresponding to the four processes which can be applied to them: pruning, compression, and expansion methods and tabular simplification (Fig. 7).

The compression method simply merges all the rules sharing the same consequent by connecting their antecedents disjunctively. For example, given the following 5 rules:

```
if(x1 == A & x2 == B) -> y = D;
if(x1 == A & x2 == C) -> y = A;
if(x1 == B & x2 == A) -> y = A;                    (8)
if(x1 == C & x2 == B) -> y = A;
if(x1 == B & x2 == C) -> y = D;
```

the result of compression is the following 2 rules:

```
if((x1 == A & x2 == B) | (x1 == B & x2 == C))  -> y = D;
if((x1 == A & x2 == C) | (x1 == B & x2 == A)
  | (x1 == C & x2 == B))  -> y = A;                (9)
```

In the other side, the expansion method implements the complementary process to compression (applied to (9), it would return (8)).

Compression (and also expansion) method may help the user to better understand or visualize the rule base but it does not really perform a simplification. Simplification can be truly carried out by the pruning method and/or the tabular simplification.

The pruning process is usually a preprocessing method applied prior to any simplification. Given a set of input data set representative of the application problem, this process evaluates the activation degree of the rules and can eliminate: (a) the *n* worst rules, or (b) all the rules except for the *n* best rules, or (c) all the rules whose activation degree is below a threshold, where the parameter *n* or the threshold are established by the user. Pruning allows reducing the number of rules by selecting the most significant ones to the application problem.

Another simplification method available at *xfsp* is a tabular simplification of the rules based on an extension of the Quine-McCluskey algorithm of Boolean design. Quine-McCluskey method performs an ordered lineal search to find all the combinations of logically adjacent minterms of the n-variable function to simplify. It begins with a list of the n-variable minterms to later obtain successively lists with (n-1)-, (n-2)-, etc., variable implicants until no more implicants can be formed, thus obtaining the so-named prime implicants of the function. The last step is to select the minimum number of prime implicants which cover all the minterms [14].

Since fuzzy systems are an extension of Boolean systems, we have extended the Quine-McCluskey method in the following way (let us consider the two-input rule base shown in Fig. 8 to illustrate the procedure). Firstly, the consequents of a fuzzy rule base are not bi-valued but can take several values (N, NS, Z, PS, and P in our example). Hence, tabular simplification is applied to every consequent (although, for the case of *r* consequents, *r*-1 simplifications could be done by using the condition else for the *r*-th consequent). Secondly, the input variables are neither bi-valued but are usually related with several fuzzy sets (MP, P, M, G, and MG, in our example). Instead of working with binary codes, we assign ordered natural numbers to the fuzzy sets of each input variable (for example, 1 to MP, 2 to P, 3 to M, 4 to G, and 5 to MG). In this way, the antecedents related with the same consequent can be combined if their addition differs in the unity and they share the same membership functions for each input variable except for one which should be consecutive. This is illustrated in Fig. 9. At the most left column (first list), the 13 rules (in tabular form) associated with the Z consequent are ordered into 7 groups (those whose antecedents sum 2, 3, 4, 5, 6, 7, and 8). The second list shows how pairs of adjacent rules of the first list can be formed. The third one, how groups of three rules can be formed from the groups of the second list. Finally, the fifth list shows the biggest groups formed by 5 adjacent

x1 \ x2	(1) MP	(2) P	(3) M	(4) G	(5) MG
(1) MP	Z	Z	Z	Z	Z
(2) P	PS	PS	Z	NS	NS
(3) M	Z	Z	Z	Z	Z
(4) G	N	N	Z	P	P
(5) MG	N	N	Z	P	P

Fig. 8. Rule base to illustrate tabular simplification.

x_1	x_2		x_1	x_2		x_1	x_2		x_1	x_2		x_1	x_2
1	1	√	1	1,2	√	1	1,2,3	√	1	1,2,3,4	√	1	1,2,3,4,5
1	2	√	1	2,3	√	1	2,3,4	√	1	2,3,4,5	√	3	1,2,3,4,5
1	3	√	1	3,4	√	1	3,4,5	√	3	1,2,3,4	√	1,2,3,4,5	3
3	1	√	1,2	3	√	1,2,3	3	√	1,2,3,4	3	√		
1	4	√	1	4,5	√	2,3,4	3	√	2,3,4,5	3	√		
2	3	√	2,3	3	√	3	2,3,4	√	3	2,3,4,5	√		
3	2	√	3	2,3	√	3	3,4,5	√					
1	5	√	3	3,4	√	3,4,5	3	√					
3	3	√	3,4	3	√								
3	4	√	3	4,5	√								
4	3	√	4,5	3	√								
3	5	√											
5	3	√											

Fig. 9. Minimization table illustrating tabular simplification.

rules, which are shown within ellipses in Fig. 8.

In order to find the simplest rule associated with a consequent (the best prime implicants of the minimization table) the following procedure is carried out: (1) Initialize to zero the set of covered minterms and select the most right list (the last list) of the minimization table. (2) From the selected list, choose the implicant which covers the largest number of non covered minterms to form the simplest rule, remove it from the list and go to step 3. If no implicant is found then go to step 4. (3) Elliminate those implicants (already selected to form the simplest rule) which are now covered by the new included implicant. Then go to step 2. (4) Select from the minimization table the list immediately before to the latest analyzed and go to step 2. The procedure finishes when all the minterms associated with the consequent are covered. In the example of Fig. 8 and 9, it can be seen how the three prime implicants of the last list cover all the minterms.

The last action performed by the tabular simplification is to use the linguistic hedges available at XFL3 to express the resulting rule in a simple and expressive way. In particular, the linguistic hedges >= and <= are used. For example, for the implicant x1(1), x2(1, 2, 3, 4, 5), the antecedent part is expressed as follows:

```
if (x1==MP & (x2 >=MP | x2 <=MG))                    (10)
```

which is simpler than:

```
if(x1==MP & (x2==MP | x2==P | x2==M | x2==G |
  x2 ==MG))                                          (11)
```

In addition, the following items are considered to simplify further the rule expressions:

(a) If all the membership functions of a variable are covered by the prime implicant which appear in the rule, that variable is eliminated of the antecedent part because its value does not matter. For example, expression in (10) is further simplified to ‘if(x1==MP)’ because x2 can take any value.

(b) If all the membership functions of a variable are covered by the prime implicant except for one, the ‘!=’ operator is used with that membership function. For example, for the implicant x1(1), x2(1, 2, 4, 5), the antecedent part is expressed as ‘if (x1==MP & x2 !=M)’.

(c) If the lowest (highest) limit of a grouping is the first (last) membership function, such condition is eliminated from the antecedent part. For example, for the implicant x1(4, 5), x2 (1, 2), the associated antecedent part is expressed as ‘if(x1>=G & x2<=P)’.

Hence, the 25 rules (in tabular form) shown in Fig. 8 are simplified into the following 5 rules (in free form):

```
if(x1==MP | x1==M | x2==M) -> y = Z;
if(x1==P & x2<=P) -> y = PS;
if(x1==P & x2>=G) -> y = NS;
if(x1>=G & x2<=P) -> y = N;
if(x1>=G & x2>=G) -> y = P;                          (12)
```

This example also illustrates how the rule base simplifications usually result in membership functions simplifications. In this case, the membership functions used for the variable x1 are 4 (MP, P, M, G) instead of 5 and those for x2 are 3 (P, M, G) instead of 5

IV. Application Examples

Two examples have been selected to illustrate the advantages of using the simplification tool *xfsp* when designing a fuzzy system. The first one deals with a rule base obtained from a set of numerical data while the second one considers a rule base obtained from translating heuristic knowledge.

A. Rule base obtained from numerical data

Let us consider the approximation of the function $(1+\sin(2\pi x)\cos(2\pi y))/2$ with a fuzzy system. Given a set of numerical data corresponding to that function, a grid-based algorithm (Wang-Mendel’s) of the identification tool of Xfuzzy 3 has been employed to generate a fuzzy system with 7 Gaussian functions for each input (Fig. 10a) and 9 singletons to represent the output variable. This means 49

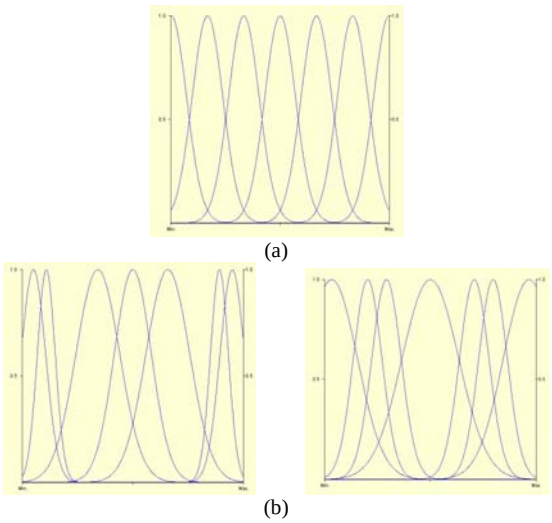

Fig. 10. Membership functions of the input variables: (a) in the initial system, and (b) in the learnt system.

rules in tabular form (49 t-norms and 38 s-norms). The approximation error obtained with this system, measured in terms of root-mean-square-error (RMSE) is 9.2%. Applying supervised learning (Marquardt-Levenberg's algorithm) with the tuning tool of Xfuzzy 3, the RMSE is reduced to 0.8%, and the membership functions of the two inputs are changed as illustrated in Fig. 10b.

The system has been then simplified in several ways. Firstly, clustering has been applied to the consequents and their number has been reduced from 9 to 5. The rule base is shown in Fig. 11. Tabular simplification has been then applied to obtain 5 expressive rules. They are transformed into 17 rules with no disjunctive operator in their antecedent parts by applying the expand method. The resulting rule base contains now 14 t-norms and 16 s-norms, which means a complexity reduction of more than 65%. In addition, the groups found by tabular simplification allows merging two pairs of membership functions of one of the input variables, thus reducing their number from 7 to 5.

An important consideration is that applying supervised learning algorithm to the simplified system, the RMSE is

x1 \ x2	(1)	(2)	(3)	(4)	(5)	(6)	(7)
(1)	c	c	c	c	c	c	c
(2)	e	d	b	a	b	d	e
(3)	e	d	b	a	b	d	e
(4)	c	c	c	c	c	c	c
(5)	a	b	d	e	d	b	a
(6)	a	b	d	e	d	b	a
(7)	c	c	c	c	c	c	c

Fig. 11. Rules identified and tuned for the approximation problem.

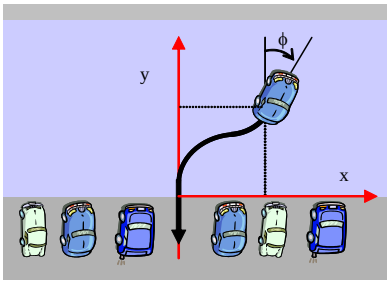

Fig. 12. Diagonal parking problem.

0.4%. Hence, the simplified system is also better in terms of approximation error.

B. Rule base obtained from heuristic knowledge

Let us consider the diagonal parking problem (Fig. 12), which in several works reported in the literature has been addressed with fuzzy controllers that translate the heuristic knowledge of a driver. In our case, we have considered the realistic situation that our car can drive forward or backward to finish correctly at the parking place. Hence, one of the rule bases of the fuzzy controller is in charge of deciding the driving direction. The first description of this rule base, obtained from our heuristic knowledge, contained 17 rules, such as the following:

"If the y coordinate is near the cars of the parking place and the x coordinate is not approximately zero and the car orientation ϕ is greater than approximately –90º and smaller than approximately 90º, the driving direction should be forward to avoid collision with the parked cars".

Table II shows the above rule and other five ones related to the situation '*y* near' expressed with the XFL3 language. The universes of discourse of the input variables *x*, ϕ, *y*, and *oldv* (the current driving direction of the car) are covered initially by 3, 7, 5, and 5 membership functions, respectively. In the other side, the output variable can take three singleton or constant values: stop (0 m/s), forward (1 m/s) or backward (-1 m/s).

Some of the initial and heuristic rules may not be adequate for the control objective. For instance, simulation results obtained within Xfuzzy 3 revealed that when the car was driving forward, it stopped too far before changing the

TABLE II
HEURISTIC RULES EXPRESSED WITH XFL3

```
1) if(y == near & x != CE & phi >= LE & phi <= RI)
   -> pv = forward;
2) if(y == near & x != CE & (phi < LE | phi > RI))
   -> pv = backward;
3) if(y == near & x == CE & (phi < LE | phi > RI))
   -> pv = backward;
4) if(y == near & x == CE & (phi == LE | phi == RI))
   -> pv = forward;
5) if(y == near & x == CE & oldv < stop & phi >= LS & phi <= RS)
   -> pv = backward;
6) if(y == near & x == CE & oldv >= stop & phi >= LS & phi <= RS)
   -> pv = forward;
```

x \ ϕ	LB	LE	LS	CE	RS	RI	RB
LE	bw	fw	fw	fw	fw	fw	bw
CE	bw	fw	bw	bw	bw	fw	bw
RI	bw	fw	fw	fw	fw	fw	bw

Fig. 13. Rules derived from those in Table I.

driving direction to backward. The monitoring tool of Xfuzzy 3 showed that the rule 5 and, mainly, rule 6 in Table II were responsible of that behavior. Hence, rule 6 was eliminated as well as the antecedent 'oldv<stop' in rule 5. The resulting five rules cover all the possible situations in x and ϕ as shown in Fig. 13 (y==near). If the tabular simplification method of the tool *xfsp* is now applied, the groupings depicted with gray lines in Fig. 13 are formed, thus resulting the two rules shown in Table III. If this simplification process is applied to the whole rule base the initial 17 rules obtained from our heuristic knowledge are reduced to 14 by using the verification tools of Xfuzzy, and then, these 14 rules are reduced to 8 by using the tool *xfsp*. In addition, the number of membership functions for the y variable are reduced from 5 to 4 (one of the zones considered in the y direction was redundant) and the membership functions relevant for the x direction is 1 instead of 3.

ACKNOWLEDGMENT

Authors are acknowledged to Jorge Agudo Praena who programmed the first version of the tool *xfsp*.

V. CONCLUSION

The CAD tool presented in this paper allows automating the simplification of fuzzy system descriptions to obtain simpler and more linguistically interpretable systems. This is possible thanks to including purging, clustering and similarity-based algorithms reported in the literature that eliminate and/or merge similar membership functions that appear often after applying tuning and identification steps. The tool also includes a pruning algorithm to reduce the number of rules depending on their activation degree and a tabular simplification algorithm developed by the authors which is inspired by the Quine-McCluskey method. The latter algorithm translates typical rule bases described in tabular form, which usually contain many rules, into simple rule bases whose rules exploit the use of linguistic hedges and, hence, are more expressive. To the best of our knowledge no other fuzzy software offers so many facilities.

Since the developed CAD tool is integrated within the design environment Xfuzzy 3 developed at the Instituto de Microelectrónica de Sevilla, the system to simplify can be described from heuristic knowledge or identified from numerical data by using the expressive language XFL3; and can be verified, tuned and synthesized with the other CAD tools of Xfuzzy.

TABLE III
RESULT AFTER SIMPLIFYING THE RULES IN TABLE II

1) if(y==near & ((x==CE & phi>= LS & phi <= RS) | phi<LE | phi>RI))
   -> pv=backward;

2) if(y==near & (x!=CE & (phi>=LE & phi<=RI) | phi==LE | phi==RI)
   -> pv= forward;

REFERENCES

[1] M. Sugeno, T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling", *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, 1993.

[2] P. Carmona, J. L. Castro, J. M. Zurita, "FRIwE: fuzzy rule identification with exceptions", *IEEE Trans. on Fuzzy Systems*, vol. 12, no. 1, pp. 140-151, Feb. 2004.

[3] M. Setnes, R. Babuska, U. Kaymak, H. R. van Nauta Lemke, "Similarity measures in fuzzy rule base simplification", *IEEE Trans. on Systems, Man and Cybernetics, part B*, vol. 28, no. 3, pp. 376-386, June 1998.

[4] D. Nauck, "Data Analysis with Neuro-Fuzzy Methods", PhD. Dissertation, Univ. of Magdeburg, Faculty of Computer Science, Germany, 2000.

[5] R. Senhadji, S. Sánchez-Solano, A. Barriga, I. Baturone, F. J. Moreno-Velo, "NORFREA: An Algorithm for non-redundant fuzzy rule extraction", Proc. IEEE SMC'2002, vol. 1, pp. 604-608, Tunisia, Oct. 2002.

[6] R. Rovatti, R. Guerrieri, G. Baccarani, "An enhanced two-level Boolean synthesis methodology for fuzzy rules minimization", *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 3, pp. 288-299, Aug. 1995.

[7] F. J. Moreno-Velo, I. Baturone, F. J. Barrio, S. Sánchez Solano, A. Barriga, "A Design Methodology for Complex Fuzzy Systems", Proc. 3rd European Symp. on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems, EUNITE'2003, Oulu, July 2003.

[8] F. J. Moreno-Velo, S. Sánchez-Solano, A. Barriga, I. Baturone, D. R. López, "XFL3: A New Fuzzy System Specification Language", *Mathware & Soft Computing*, pp. 239-: 253 , December 2001.

[9] I. Baturone, F. J. Moreno-Velo, A. Gersnoviez, "Identifying fuzzy systems from numerical data with Xfuzzy", Proc. 4th Conference of the European Society for Fuzzy Logic and Technology, EUSFLAT'2005, pp. 1257-1262, Barcelona , Sept. 2005

[10] F. Klawonn, R. Kruse, "Constructing a Fuzzy Controller from Data", Fuzzy Sets and Systems, 85, pp. 177-193, 1997.

[11] L. Wang, J.M. Mendel, "Generation Rules by Learning from Examples". Proc. Int. Symp. on Intelligent Control, pp. 263-268, 1991.

[12] E. H. Ruspini, P. P. Bonissone, W. Pedrycz, Eds., *Handbook of Fuzzy Computation*, Institute of Physics Pub., 1998

[13] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, New York Academic, 1980.

[14] E. J. McCluskey Jr., *Introduction to the Theory of Switching Circuits*, McGraw-Hill Book Co., New York, 1965.