Mathware & Soft Computing 7 (2000) 199-212

On Defining Multiple-valued Logics for Knowledge-based Systems Communication

J.A. Reyes, J. Puyol-Gruart, F. Esteva
Artificial Intelligence Research Institute (IIIA)
Spanish Scientific Research Council (CSIC)
Campus UAB. 08193 Bellaterra, Catalonia, Spain
{*reyes, puyol, esteva*}*@iiia.csic.es*

Abstract

Multiple-valued logics are useful for dealing with uncertainty and imprecision in Knowledge-Based Systems. Different problems can require different logics. Then we need mechanisms to translate the information exchanged between two problems with different logics. In this paper, we introduce the logical foundations of such logics and the communication mechanisms that preserve some deductive properties. We also describe a tool to assist users in the declaration of logics and their communication mechanisms.
Keywords: Uncertain Reasoning, Multiple-Valued logics, Knowledge-Based Systems.

1 Introduction

The management of uncertainty and imprecision in knowledge-based systems (KBS) becomes essential to model many real problems. Multiple-valued logics (MV-Logics) have been proved to be useful in knowledge-based systems [2, 4, 6, 12]. Different problems can require different MV-Logics, those more adapted to them.

This paper is related to the problem of how to make compatible the communication of information among different MV-Logics maintaining some properties of the deduction. We present the theoretical foundations of our MV-logics and their combinations, and **QMORPH** [11], a tool which makes automatic the process of defining and combining finite MV-Logics. This tool has been designed to be incorporated to the shell **Milord II** [10, 9]—a shell and a language to develop KBS—although it can be used in a more extensive framework. The basic construct of **Milord II** language is the module. A **Milord II** application is a hierarchy of modules. Each module is a complete KBS containing the declaration of submodules, facts, rules, meta-rules, and the particular MV-Logic of the module. The declaration of this *local logic* includes how to make the translation of the information provided by a module with a different MV-Logic.

This paper is organized as follows. In section 2, we introduce a parametric family of MV-logics determined by a particular algebra of truth-values and the extension of that algebra to an algebra of intervals of truth-values. In Section 3 we discuss the eficient generation of conjunction operators. Section 4 is devoted to establish the essential requirements needed to preserve several deductive properties when we communicate different logics. We give conditions for every requirement and we identify different sorts of renaming functions between logics. In section 5, we describe the main features of **QMORPH**. Finally, in section 6 we outline the conclusions of this work.

2 Defining MV-Logics

In this section we will give the definitions required to declare the family of finite MV-logics we deal within this paper and which are expressive enough to model the uncertain reasoning used in many rule-based systems (see [1, 2, 4]), including **Milord II**. See [7] for a mathematical characterization of t-norms (continuous-like or smooth) in a finite linearly ordered scale. . Each logic of this family is determined by a particular **algebra of truth-values** defined as a finite algebra

$$A_T^n = \langle A_n, \mathbf{0}, \mathbf{1}, N_n, T, I_T \rangle$$

where:

1. The ordered set of **truth-values** A_n is a chain of n elements:

   $$\mathbf{0} = a_0 \prec a_1 \prec \ldots \prec a_{n-1} = \mathbf{1}$$

   where $\mathbf{0}$ and $\mathbf{1}$ are the Boolean *False* and *True* respectively.

2. The **negation** operation N_n is the unary operation univocally determined by the order reversing and involutive properties. defined as:

   $$N_n(a_i) = a_{n-1-i}$$

3. The **conjunction** operation T is a binary operation such that satisfy the following properties $\forall a, b, c \in A_n$:

   - **T1**: $T(a, b) = T(b, a)$
   - **T2**: $T(a, T(b, c)) = T(T(a, b), c)$
   - **T3**: $T(\mathbf{0}, a) = \mathbf{0}$
   - **T4**: $T(\mathbf{1}, a) = a$
   - **T5**: If $a \leq b$ then $T(a, c) \leq T(b, c)$ for all c

4. The **implication** operation I_T is defined by residuation with respect to T, i.e.

   $$I_T(a, b) = Max\{c \in A_n | T(a, c) \leq b\}$$

$T(x,y)$	**false**	**unlikely**	**maybe**	**likely**	**true**
false	*false*	*false*	*false*	*false*	*false*
unlikely	*false*	*unlikely*	*unlikely*	*unlikely*	*unlikely*
maybe	*false*	*unlikely*	*maybe*	*maybe*	*maybe*
likely	*false*	*unlikely*	*maybe*	*maybe*	*likely*
true	*false*	*unlikely*	*maybe*	*likely*	*true*

Table 1: T conjunction operation on A_5.

In Table 1 we can see a possible table for T on the set of five truth-values $A_5 = \{false, unlikely, maybe, likely, true\}$. Notice that an algebra of truth-values is univocally defined by the set of truth-values A_n and the operation T, since I_T and N_n are univocally determined by T and n respectively.

The language is composed of:

- **Well-formed Formulas (wff):** obtained in the usual way from a denumerable set of propositional symbols and the connectives $\neg$, $\wedge$ and $\rightarrow$.
- **Sentences:** pairs (p, V) of a wff p and an interval of truth-values V.

The semantic interpretation is given by:

- **Models:** defined by valuations ρ from the first component of sentences to A_n provided that:
  - $\rho(\neg p) = N(\rho(p))$
  - $\rho(p_1 \wedge p_2) = T(\rho(p_1), \rho(p_2))$
  - $\rho(p \rightarrow q) = I_T(\rho(p), \rho(q))$
- **Satisfaction Relation:** between models and sentences defined by:

$$M_\rho \models (p, V) \text{ if, and only if } \rho(p) \in V$$

where M_ρ stands for the model defined by a valuation ρ.

A MV-Logic Calculus is defined by a set of axioms and rules of inference that have to be sound with respect to the satisfaction relation given above (see the particular calculus of **Milord II** in [8]). The most common rule of inference on rule-based systems is *modus ponens*. The corresponding Modus Ponens operator MP_T between values of the algebra of truth-values is defined by:

$$MP_T(a,b) = \begin{cases} \emptyset & \text{if } a \text{ and } b \text{ are inconsistent} \\ [a, \mathbf{1}] & \text{if } b = \mathbf{1} \\ T(a,b) & \text{otherwise} \end{cases}$$

which gives the set of solutions of the equation $I_T(a,c) = b$, and where $a, b \in A_n$ are said to be inconsistent if there is no solution c to this equation.

Given an algebra of truth-values we are interested in the extension to the algebra of intervals of truth-values (see [3]). We have some motivations: the modus ponens operator above requires the use of intervals to chain rules, we want to deal with imprecision, and—as we will see in the next section—intervals are needed to make possible mappings between different logics.

Given an algebra of truth-values $A_T^n = \langle A_n, \mathbf{0}, \mathbf{1}, N_n, T, I_T \rangle$, we will consider the set of intervals of A_n as $Int(A_n) = \{[a,b] | a,b \in A_n\}$, being $[a,b] = \{x \in A_n | a \preceq x \preceq b\}$. The extension of the algebra operators for dealing appropriately with intervals are defined by: $N_n^*([a,b]) = [N_n(b), N_n(a)]$, $T^*([a,b],[c,d]) = [T(a,c), T(b,d)]$, $MP_T^*([a,b],[c,\mathbf{1}]) = [T(a,c),\mathbf{1}]$. Notice that $N_n^*([a,b])$ is the minimum interval of $Int(A_n)$ such that for all $x \in [a,b]$, $N_n(x)$ belongs for sure, and the same is valid for T^* and MP_T^*. Coherently with the definitions the order between intervals is extended to $\prec^*$. Then $[a_i,a_j] \prec^* [a_k,a_l]$ if $a_j \prec a_k$, or $a_j = a_k$ and $[a_i,a_j] \neq [a_k,a_l]$.

3 Conjunction Generation

To declare an algebra of truth-values it is only necessary to determine the linguistic terms more adequate for the concerning problem, and define a conjunction operator necessary to combine and propagate uncertainty when making inference. Next we discuss how the conjunction operator has been generated and how this process has been automated.

Throughout this work we represent the table of the conjunction operator as a matrix M, where each element is a variable. If the algebra has n elements, then M is a $n \times n$ matrix which elements are noted by $V_{i,j} = T(a_i, a_j)$, where $0 \leq i,j \leq n-1$.

Properties **T1-T5** act as constraints over the set of possible solutions. Satisfaction of each property or constraint, causes the following guidelines which influence the conjunction search generation problem:

- Commutativity allows us to consider only the set of variables $V = \{V_{i,j}, i \geq j\}$.
- Existence of absorbent and neutral elements implies to fix the values for variables $V_{0,j}$ and $V_{n-1,j}$, where $0 \leq j \leq n-1$.
- Monotonicity requires $max(V_{i-1,j}, V_{i,j-1}) \leq V_{i,j} \leq max(V_{i+1,j}, V_{i,j+1})$.
- Associativity test is expensive in time and memory, but it can be reduced taking into account the commutative property and considering that $T(a_i,a_j) \leq min(a_i,a_j)$, each submatrix M_k (a matrix $k \times k$ with elements $V_{i,j}$ where $0 \leq i,j \leq k$) is closed with respect to the conjunction operator. In this way the matrix will be associative if all its sub-matrices are so. This property let us check associativity in an incremental way every time a value is assigned to a variable $V_{i,j}$.

Applying these constraints, the number of operators generated is exponential with respect to n. It is significantly high for a certain number of linguistic terms,

n	**A**	**B**	**C**			**D**		
			$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$
3	2	1	2	2	2	1	1	1
4	6	2	5	6	6	2	2	2
5	22	6	13	21	22	5	6	6
6	94	22	38	78	93	13	21	22
7	451	94	118	306	422	38	78	93
8	2386	451	395	1274	2002	118	306	422
9	–	2386	1404	–	–	395	1274	2002

Table 2: Matrices per number of terms. **A: T1-T5**, **B: T1-T6**, **C: T1-T5** and **T7**, **D: T1-T7**

in particular, for $n > 5$ we obtain more than 22 matrices (as we can see in Table 2, column A). In fact we are interested in fewer possibilities, because finally we will only choose one operator. Then we can reduce this number by fixing several values of the matrix or by introducing new constraints. Two types of well-known desirable—although optional —properties are discussed in the literature (see [5]).

It seems reasonable that the conjunction of two truth-values different than **0** should not be **0**. This is achieved by the property:

> **T6 Strictness**: $T(a_i, a_j) \neq 0$, for all $i, j \neq 0$

In the infinite case we require some continuity. Similarly, in the finite case the conjunction of two values should be close to the conjunction of different, but close values. This property is defined as follows:

> **T7 α-Smoothness**: given $\alpha \in \mathbb{N}$, T is said to satisfy α-smoothness property if: $T(a_i, a_j) = a_k$ and $T(a_{i-1}, a_j) = a_p$, imply $k - p \leq \alpha$

Satisfaction of these news properties causes the following influences in the conjunction search generation problem: Strictness implies that the generation of strict matrices of dimension n is equivalent to generate non-strict matrices of dimension $n - 1$; α-Smoothness reduces the set of possible values for a variable. It takes into account column and row adjacency.

The definition of a conjunction operation is given by stating its properties. There are the mandatory properties (**T1-T5**), and additional ones (**T6,T7**) in order to decrease the number of possible solutions (see the experimentals results in Table 2). As an additional option, we consider that some of the variables of the conjunction matrix can be fixed.

3.1 Generation as a Constrain Satisfaction Problem

The problem of finding the T operators satisfying a set of properties can be formulated as a classic Constraint Satisfaction Problem (CSP):

- Initially, we have a set of variables with fixed values. This set is composed by the values concerning the variables: $V_{0,j} = a_0$, $V_{n-1,j} = a_j$, with $0 \leq j \leq n-1$ (**T3,T4**), and the ones fixed by the user.

- We only take into account the set of variables $V = \{V_{i,j} | i \geq j$ with $i, j \neq 0$ and $i, j \neq n-1\}$ (**T1**).

- The initial domain for these considered variables is $D_{i,j} = \{a_0, \ldots, a_i\}$ (**T5**).

- This initial domain can be modified according to strictness and α-smoothness properties (**T6,T7**). If strictness is applied, then the a_0 value is rejected. If α-smoothness is applied, the domain length will be up to $\alpha + 1$ elements, keeping the last $\alpha + 1$ elements of the set if this number is greater.

We solved this search generation problem with a depth-first algorithm applying backtracking and look-ahead. Backtracking needs the existence of stacks. In this case, we use one to keep matrices generated and other one to keep which values remains unassigned to each variable. The generated solution results are shown in Table 2.

Now we know how to define a logic and how to solve and automate the conjuntion generation. In the next section we will see how to communicate different logics and how to preserve certain deductive properties in this process.

4 Combining Logics

As we have seen in the previous sections, we can declare different logics varying the set of truth-values (linguistic terms) and the conjunction operator.

When two different logics need to exchange information, it is necessary some mechanism of translation of the linguistic terms to make the communication between those logics compatible, preserving the deductive properties of each one. As usual, when information is transmitted, we loose some precision. This loss of information can be represented by mapping truth-values of one logic to intervals of truth-values of the other one. In the following we analyze different requirements for this mapping [1].

Let $(L, \vdash)$ and $(L', \vdash')$ be two logics, L and L' standing for the languages, and $\vdash$ and $\vdash'$ for the entailment relations defined on L and L' respectively. To establish a correspondence between both logics, a mapping $H : L \to L'$ is needed relating their languages. Next, we will analyze some natural requirements for the mapping H with respect to the entailment systems $\vdash$ and $\vdash'$ proposing that at least one of the following three requirements should be fulfilled by the mapping H in order to ensure a consistent communication. Henceforth Γ and e will denote a set of sentences and a sentence of L respectively.

A map H is said to be *forward conservative* when,

RQ.1. If $\Gamma \vdash e$, then $H(\Gamma) \vdash' H(e)$

With this requirement we ensure that for every sentence e, deducible from a set of sentences Γ, its corresponding sentence, $H(e)$, will also be deducible from the corresponding sentences of $H(\Gamma)$.

A map H is said to be *backward conservative* when,

RQ.2. If $H(\Gamma) \vdash' H(e)$, then $\Gamma \vdash e$

This is the inverse requirement of **RQ.1.** So, in this case if a fact is not deducible from Γ, then its corresponding fact from $H(\Gamma)$ will not be deducible either. Nevertheless $\Gamma \vdash e$ does not imply $H(\Gamma) \vdash' H(e)$.

A map between entailment systems allows us to preserve inference in a strict sense. In particular, when a map is *conservative* (*forward* or *backward*), one entailment system is an extension of the other one. Conditions **RQ.1** and **RQ.2**, which are very strong, can be weakened in the uncertainty reasoning framework. That means that we can consider not to deduce always the same sentences, but sometimes only coherent sentences.

Formally, this can be expressed by the third and last requirement:

RQ.3. If $H(\Gamma) \vdash' e'$, then there exists e such that $\Gamma \vdash e$ and $H(e) \vdash' e'$

This requirement assures that every sentence deducible from $H(\Gamma)$ must be in agreement with those deduced from Γ. This requirement is slightly different from **RQ.2**, in the sense that it is not necessary to be e' an exact translation of a deducible sentence e from Γ, but only something deducible from such a translation. In the framework of logics for uncertainty management, e' can be interpreted as a *weaker* form of e, i.e. a sentence expressing more uncertainty than e. We call it a *weak conservative* map.

4.1 Algebra Morphisms

Now we consider the problem of finding inference preserving correspondences between two algebras $A_T^n = \langle A_n, \mathbf{0}, \mathbf{1}, N_n, T, I_T \rangle$ and $B_{T'}^m = \langle B_m, \mathbf{0}, \mathbf{1}, N'_m, T', I_{T'} \rangle$. We are interested in mapping the entailment system $(L_A, \vdash_A)$ into the entailment system $(L_B, \vdash_B)$, by means of *renaming functions* between the corresponding sets of linguistic terms. We consider those mappings from L_A to L_B that just involve translations of truth-values, i.e. any mapping $H : L_A \to L_B$ is defined as $H(e, V) = (e, h(V))$, where the *renaming function* h translates subsets of values of A_n into subsets of values of B_m, and e (a well-formed formula) remains invariant.

We are interested in *renaming functions* h such that H satisfy inference preserving properties **RQ.1**, **RQ.2** and **RQ.3**.

In [1] it is proved first that such function h exists if h maps truth-values of A_n into intervals of truth-values of B_m, that is $h : A_n \to Int(B_m)$ (it is not always possible to find mapping if we restrict h to be $h : A_n \to B_m$) and second it is proved which are the necessary and sufficient conditions for h (summarized in Table 3) in order H satisfy the above inference preserving properties.

Sometimes there exists *renaming functions* h satisfying the required conditions such that $h(a_i) = [b_j, b_j]$, i.e., mapping elements of A_n into *elements* (intervals of

RQ.1	$\Leftrightarrow$	$h(T(V_1, V_2)) \supseteq T'^*(h(V_1), h(V_2))$ $h(N_n(V)) = N_m^*(h(V))$ $h(V_1 \cap V_2) = h(V_1) \cap h(V_2)$	
RQ.2	$\Leftarrow$	$h(T(V_1, V_2)) \subseteq T'^*(h(V_1), h(V_2))$ $h(N_n(V)) = N_m^*(h(V))$ $h(V_1) \supseteq h(V_2) \Rightarrow V_1 \supseteq V_2$	$\Downarrow$
RQ.3	$\Leftrightarrow$	$h(T(V_1, V_2)) \subseteq T'^*(h(V_1), h(V_2))$ $h(N_n(V)) = N_m(h(V))$	

Table 3: Requirements conditions.

type $[b_j, b_j]$) of B_m. In such a case if h satisfy one of the requirements, h is always a *morphism* of the algebras of truth-values. As a particular case if h satisfy the requirements **RQ.1** or **RQ.2**, then h is a *mono-morphism*, an embeding of A_n into B_m. Finally if h maps elements of A_n into proper intervals of B_m, then if it satisfies **RQ.3**, h is said to be a *quasi-morphism*.

We are mainly interested in *mono-morphisms* because they embed A_n into B_m and because they are order preserving mappings (it is equivalent to a communication without any loose of information). After *mono-morphisms*, we are interested in *morphisms*, which accomplish the algebra operations (it is a transmission of information fulfilling the required properties). Finally, because of the strong conditions for *morphisms* and *mono-morphisms*, it is not always possible to find these kind of renaming functions, *quasi-morphisms* can be useful thanks to the additional freedom of map truth-values of an algebra into intervals of the other (we allow certain loose of information).

4.2 Renaming Algorithm

For a given set of truth-values A_n, there exists only one negation operator N_n, and it is defined by $N_n(a_i) = a_{n-i-1}$. Then, we can make a partition of the set A_n into three subsets:

- the set of negative elements $\mathfrak{N}_\mathfrak{n} = \{x | x \prec N(x)\}$
- the set of fixed elements $\mathfrak{F}_\mathfrak{n} = \{x | x = N(x)\}$
- the set of positive elements $\mathfrak{P}_\mathfrak{n} = \{x | x \succ N(x)\}$

being the subsets: $\mathfrak{F}_\mathfrak{n} = \{a_k\}$, $\mathfrak{N}_\mathfrak{n} = \{a_i | i < k\}$, $\mathfrak{P}_\mathfrak{n} = \{a_i | i > k\}$, if $n = 2k + 1$, and $\mathfrak{F}_\mathfrak{n} = \emptyset$, $\mathfrak{N}_\mathfrak{n} = \{a_i | i < k\}$ and $\mathfrak{P}_\mathfrak{n} = \{a_i | i \geq k\}$, if $n = 2k$.

In the same way, we can do the same with a set B_m, obtaining $\mathfrak{N}_\mathfrak{m}^*$, $\mathfrak{F}_\mathfrak{m}^*$, and $\mathfrak{P}_\mathfrak{m}^*$ for the case of working with intervals. Then, the renaming algorithm is as follows:

1. Initialization: Obtain the subsets $\mathfrak{N}_\mathfrak{n}$, $\mathfrak{F}_\mathfrak{n}$, $\mathfrak{N}_\mathfrak{m}^*$, and $\mathfrak{F}_\mathfrak{m}^*$.
2. Maps Generation: Generate all the maps $h_1 : \mathfrak{N}_\mathfrak{n} \cup \mathfrak{F}_\mathfrak{n} \rightarrow \mathfrak{N}_\mathfrak{m}^* \cup \mathfrak{F}_\mathfrak{m}^*$ such that:

$T'(x,y)$	**impos**	**fewp**	**slightp**	**possib**	**quitep**	**veryp**	**sure**
impos	*impos*	*impos*	*impos*	*impos*	*impos*	*impos*	*impos*
fewp	*impos*	*fewp*	*fewp*	*fewp*	*fewp*	*fewp*	*fewp*
slightp	*impos*	*fewp*	*slightp*	*slightp*	*slightp*	*slghtip*	*slightp*
possib	*impos*	*fewp*	*slightp*	*possib*	*possib*	*possib*	*possib*
quitep	*impos*	*fewp*	*slightp*	*possib*	*quitep*	*quitep*	*quitep*
veryp	*impos*	*fewp*	*slightp*	*possib*	*quitep*	*veryp*	*veryp*
sure	*impos*	*fewp*	*slightp*	*possib*	*quitep*	*veryp*	*sure*

Table 4: T' conjunction operation on B_7.

(a) $h_1(0) = 0$.

(b) $h_1(\mathfrak{F}_\mathfrak{n}) \in \mathfrak{F}^*_\mathfrak{m}$.

(c) $x \preceq y$ implies $h_1(x) \not\succ h_1(y)$, where $x, y \in A_n$ and $h_1(x), h_1(y) \in Int(B_m)$.

3. Map extension: Extend each mapping h_1 with respect to the negation operation defining the morphism h as:

$$h(x) = \begin{cases} h_1(x) & x \in \mathfrak{N}_\mathfrak{n} \cup \mathfrak{F}_\mathfrak{n} \\ N^*_m(h_1(N_n(x))) & x \in \mathfrak{P}_\mathfrak{n} \end{cases}$$

4. Conjunction checking: Check which ones are compatible with the conjunction operators T and T'.

5. Renaming checking: Finally check which mappings h are *mono-morphisms*, *morphisms* or *quasi-morphisms*.

As an example, consider two logics declared with the sets of truth-values A_5 and B_7, and with the conjunction operations T and T' (defined in Table 1 and Table 4 respectively).

$A_5 = \{false, unlikely, maybe, likely, true\}$
$B_7 = \{impos, fewp, slightp, possib, quitep, veryp, sure\}$

First, we obtain the initial subsets $\mathfrak{N}_5$, $\mathfrak{F}_5$, $\mathfrak{N}^*_7$ and $\mathfrak{F}^*_7$:

$\mathfrak{N}_5 = \{false, unlikely\}$
$\mathfrak{F}_5 = \{maybe\}$
$\mathfrak{N}^*_7 = \{impos, fewp, slightp, possib, [impos, fewp], [impos, slightp], [impos, possib], [fewp, slightp], [fewp, possib], [slightp, possib]\}$
$\mathfrak{F}^*_7 = \{possib, [fewp, veryp], [slightp, quitep], [impos, sure]\}$

And following the above algorithm schema, we can find several mappings between both logics. There are here two mappings that are examples of those that

hold the last requirement **RQ.3**:

$$\begin{cases} false \to impos \\ unlikely \to [impos, fewp] \\ maybe \to [fewp, veryp] \\ likely \to [veryp, sure] \\ true \to sure \end{cases} \qquad \begin{cases} false \to impos \\ unlikely \to [fewp, slip] \\ maybe \to possible \\ likely \to [quitep, veryp] \\ true \to sure \end{cases}$$

When ending the generation process, the list of inference preserving mappings is presented to the user in this way:

1. First, we offer the existent mono-morphisms (one-to-one *morphisms*).
2. Next, we show which *morphisms* are between both algebras, if any.
3. Finally, we display the list of generated *quasi-morphisms*.

Due to the strong conditions they must fit, it is not always possible to find *morphisms* and *mono-morphisms*. In addition, it is very possible that the renaming generation produces a large list of *quasi-morphisms*, so the possibility of giving an ordered list of *quasi-morphisms* to aid users in their final selection may be considered.

For the purpose of producing an ordered list of *quasi-morphisms*, we consider a *weight* as a relation among the number of terms which have an atomic image (that is, an interval with the form $[a_i, a_i]$) and the length of those producing intervals (number of truth-values included in this interval).

Given two chains A_n and B_m, we will generate mappings $h : A_n \to Int(B_m)$. In order to obtain an evaluation for every map, we use the following empirical function:

$$\psi = \frac{\sum_{i=1}^{n} L_i}{k}$$

for $1 \leq L_i \leq m, 1 \leq k \leq n$, where L_i is the length of the interval $h(a_i)$ and k is the number of points a_i which have an atomic image[1]. Then, for every mapping we obtain a value ψ, such that:

$$1 \leq \psi \leq \frac{(n-2)m + 2}{2}$$

This method produces an ordered list of *quasi-morphisms* that simplify the selection of a specific one by the user.

User is also able to establish the behavior that can take the generated maps defining a partial map between both algebras. This definition must satisfy the set of requirements suggested previously in order to be a negation *morphism*. That is, our partial map must accomplish the next points:

1. It must be a non-decreasing function.

[1] Note that in the case of morphisms, all the mappings produce the same evaluation.

2. If our map involves the truth-value a_0, its image must be the truth-value b_0. For instance, in the above example any map must fit $h(false) = impos$. Therefore, to be a negation *morphism*, it is the same for the case of truth-value a_{n-1}, so it is required that $h(true) = sure$.

3. If our map implies the fixed element of the first chain A_n, we must give as its image an element belonging to the set of fixed elements $\mathfrak{F}^*_{\mathfrak{m}}$. When we have defined our partial map (or total, or none), a list of renaming functions fulfilling these defined characteristics will be generated.

Now we know how to declare a MV-logic and how to establish inference preserving communication between two of these logics. In the following section we will present the interactive implemented tool to define and combine MV-logics.

5 The QMORPH Tool

QMORPH [11] is a tool that allows users to define and combine MV-logics. Two different interfaces have been developed: a graphical interface for the UNIX operative system, running under the X-Windows environment and developed on Sun machines using *Tcl/Tk* packages and a generic text mode interface performed in Common Lisp for the Macintosh system. In the present, this tool is attached to **Milord II**, as an tool to assist experts when developing modular applications.

In this section, we illustrate the use of **QMORPH** throughout an example in which we try to find existing morphisms between two MV-logics.

We consider the same above example. Let us suppose we need to establish a communication from $A^5_T = \langle A_5, \mathbf{0}, \mathbf{1}, N_5, T, I_T \rangle$ (Origin Module or O-Module) to $B^7_{T'} = \langle B_7, \mathbf{0}, \mathbf{1}, N_7, T', I_{T'} \rangle$ (Image Module or I-Module).

Figure 1: Conjunction generation interfaces.

First, the user has to define the main characteristics of both logics. It is mandatory to declare the set of truth-values of both logics and the T operation of the Origin Module for avoiding a computational explosion.

This is made by entering first how many linguistic terms compose the logics (5 and 7) and optionally the concrete names for the terms—otherwise they will be generated automatically. In this example we consider the terms A_5 and B_7 of the previous examples.

Secondly, the user has to declare the conjunction operator that the O-Module uses to combine and propagate the linguistic terms when making inference, in this case T. The user can choose this function depending on the meaning he wants to give to the conjunction operation. There are some matrix values that remain fixed (by properties: $T(0, a_i) = 0$ and $T(1, a_i) = a_i$), so these values will be presented automatically avoiding any changes. There are additional possibilities: if the user specify the conjunction completely, then the system will check whether that conjunction operator holds the mandatory properties or not; it is also possible to give a partial matrix with some pre-fixed values; and finally, there is the possibility of not giving any value, so all the conjunction operators would be generated. To restrict the number of possible solutions, user can apply strictness and smoothness constraints over the process generation. In our example (see Figure 1), the user fixs T_5(unlikely, unlikely) = unlikely.

If many matrices have been generated, user must choose one of them. If user consider that there are too many matrices, it is possible to come back to redefine the matrix and the restrictions over it. To choose among the set of generated matrices, results are presented in the Matrices Reproducer (Figure 1) where we can display all the generated matrices and select the one that fits better.

Figure 2: Renaming functions generation interfaces.

Once the logic of the A^5_T (O-Module) has been determined, we must define the logic concerning to $B^7_{T'}$ (I-Module). The procedure for defining an I-Module conjunction operator is exactly the same that in the case of the O-Module, but it is not

necessary to choose one of them. Then for every matrix solution, all the renaming mappings between both defined algebras will be generated. It is the possible to give a partial map between both algebras. If user desire a particular truth-value from one logic to be translated to another truth-value (or to an interval), the values of the partial map will be checked to be correct (see Figure 2). It is permitted to choose a determinate *criteria* to be satisfied during renaming generation. There are three *criteria*: A, B and C, corresponding to the requirements **RQ.1**, **RQ.2** and **RQ.3**, respectively.

6 Conclusions

In this paper we have defined which are the theoretical foundations that allow to deal with uncertainty by means of multiple-valued logics. We have settled how to declare a suitable MV-logic from a parametric family of algebra of truth-values, as well as which requirements are necessary when we need to establish a communication between two of these logics.

We have presented the developed algorithms to aid users in the generation of a specific conjunction operation (implemented as a classic CSP) and in the search of an inference preserving mapping.

The performed tool has been designed with the aim of automating these problems, as well as assisting users and offering different alternative possibilities. Beyond its particular use within **Milord II**, this tool can be deployed by any other system using MV-logics.

References

[1] J. Agustí, F. Esteva, P. Garcia, L. Godo, R. López de Mántaras, and C. Sierra. Local multi-valued logics in modular expert systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 6(3):303–321, 1994.

[2] P.P. Bonissone, S.S. Gans, and K.S. Decker. Rum: A layered architecture for reasoning with uncertaintyg. In *IJCAI'87*, pages 891–898, 1987.

[3] F. Esteva, P. Garcia-Calves, and L. Godo. Enriched interval bilattices: An approach to deal with uncertainty and imprecision. *Uncertainty, Fuzzyness and Knowledge–Based Systems*, 1994.

[4] L. Godo, R. López de Mántaras, C. Sierra, and A. Verdaguer. Milord: The architecture and management of linguistically expressed uncertainty. *International Journal of Intelligent Systems*, 4:471–501, 1989.

[5] Lluís Godo and Carles Sierra. A new approach to connective generation in the framework of expert systems using fuzzy logic. In *Proceedings of the 18th International Symposium on Multiple-Valued Logics*, pages 157–162, 1988.

[6] Ramon López de Mántaras. *Approximate Reasoning Models.* Ellis Horwood Series in Artificial Intelligence, 1990.

[7] G. Mayor and J. Torrens. On a class of operators for expert systems. *International Journal of Intelligent Systems*, 8:771–778, 1993.

[8] J. Puyol-Gruart, L. Godo, and C. Sierra. Specialisation calculus and communication. *International Journal of Approximate Reasoning (IJAR)*, 18(1/2):107–130, 1998.

[9] J. Puyol-Gruart and C. Sierra. Milord II: a language description. *Mathware and Soft Computing*, 4(3):299–338, 1997.

[10] Josep Puyol-Gruart. *MILORD II: A Language for Knowledge-Based Systems*, volume 1 of *Monografies del IIIA*. IIIA–CSIC, 1996. ISBN: 84–00–07499–8.

[11] J.A. Reyes. QMORF: a tool to define and combine local logics in Milord II. Master's thesis, Universitat Autònoma de Barcelona, 1997.

[12] R. Turner. *Logics for Artificial Intelligence*. Ellis Horwood Series in Artificial Intelligence, 1984.