3rd Int. Conf. on Very Large Data Bases,
Tokyo, Japan, October 6-8, 1977
(To appear in ACM Trans. on Data Base Systems)

IMPLEMENTATION OF A FUZZY-SET-THEORETIC DATA STRUCTURE SYSTEM

Masaharu Mizumoto
Motohide Umano
Kokichi Tanaka

Department of Information and Computer Sciences
Faculty of Engineering Science
Osaka University
Toyonaka, Osaka 560, Japan

This paper describes an implementation of a system for fuzzy sets manipulation which is based on FSTDS (Fuzzy-Set-Theoretic Data Structure), an extended version of Childs' STDS (Set-Theoretic-Data Structure). FSTDS language is considered as a fuzzy-set-theoretically oriented language which can deal, for example, with ordinary sets, ordinary relations, fuzzy sets, fuzzy relations, L-fuzzy sets, level m fuzzy sets, type n fuzzy sets, and generalized fuzzy sets. The system consists of an interpreter, a collection of fuzzy set operations, and the data structure FSTDS for representing fuzzy sets. FSTDS is made up of eight areas, namely, Fuzzy Set Area, Fuzzy Set Representation Area, Grade Area, Grade Tuple Area, Element Area, Element Tuple Area, Fuzzy Set Name Area, and Fuzzy Set Operator Name Area.

FSTDS system, in which 52 fuzzy set operations are available, is implemented in FORTRAN, and is currently running on a FACOM 230-45S computer.

INTRODUCTION

In the real world, there exist many fuzzy things which can not or need not be precisely defined. In the past, fuzziness has been studied as vagueness, ambiguity or uncertainty. However, since L.A. Zadeh proposed the concept of fuzzy sets in 1965[1], it has been studied vigorously and applied to various fields such as automata theory, formal languages, natural languages, logic, pattern recognition, learning theory, decision making, and mathematical theory of computation (see [2]).

It is well-known that ordinary set theory is very useful to various kinds of areas. There exist many systems which can deal with ordinary sets and relations, some of which are STDS developed by Childs[3,4], SETL by Schwartz[5,6,7], and LOREL by Katayama[8]. STDS (Set-Theoretic Data Structure) embedded in FORTRAN or MAD language introduces n-ary relations as data model and provides fairly many kinds of set and relational operations, though it has a limited form of algebraic languages as to the data sublanguage. SETL is a set-theoretical language of very high level. As an example of primitive operation in SETL, x+y means the addition of two integers or reals, the union of two sets, or the concatenation of two tuples or two character strings in accordance with the types of x and y. However, only basic operations are available in SETL. LOREL was developed to solve the combinatorial problems (e.g., graphs, automata, formal languages) which have logical relations among their data. The concept of a set in LOREL, however, is not the same as that of an ordinary set. It is just that of a linear list. PASCAL[9] can deal with sets containing a small number of elements by declaring the variables which have set type.

Since fuzzy sets are considered a generalization of ordinary sets, a system which can deal with fuzzy sets will be much more useful because of the wide applicability of fuzzy set theory.

In this paper we describe an implementation of a system for fuzzy sets manipulation which is based on FSTDS (Fuzzy-Set-Theoretic Data Structure) which is an extended version of Childs' STDS. FSTDS language is considered as a fuzzy-set-theoretically oriented language which can deal with ordinary sets, ordinary (n-ary) relations, fuzzy sets, (n-ary) fuzzy relations, L-fuzzy sets, level m fuzzy sets, type n fuzzy sets and generalized fuzzy sets. The system is demonstrated using a number of examples.

FSTDS system, in which 52 fuzzy set operations are available, is implemented in FORTRAN, and is currently running on a FACOM 230-45S computer.

FUZZY SETS AND FUZZY RELATIONS

We shall make a brief summary of the concept of fuzzy sets and fuzzy relations which will be needed in later sections.

Intuitively, a fuzzy set is a class with unsharp boundaries, that is, a class in which the transition from membership to non-membership may be gradual rather than abrupt.

Definition 1. A fuzzy set F in a universe of discourse U is characterized by a membership function:

$$\mu_F : U \longrightarrow [0, 1] \qquad (1)$$

which associates with each element u of U a number $\mu_F(u)$ in the interval [0, 1] which represents the grade of membership (grade, for short) of u in fuzzy set F, with 0 and 1 denoting non-membership and full membership, respectively. In the notation of a fuzzy set F, we use

$$F = \{\mu_F(u_1)/u_1, \mu_F(u_2)/u_2, \ldots, \mu_F(u_n)/u_n\} \qquad (2)$$

$$= \sum_i \mu_F(u_i)/u_i \qquad (3)$$

where u_i, i=1,2,...,n, represent the elements of U.

A fuzzy relation is defined as a fuzzy set of the Cartesian product of some universes of discourse.

Definition 2. A fuzzy relation R (especially, a binary fuzzy relation) in UxV or U to V is characterized by a bivariate membership function as

$$\mu_R : U \times V \longrightarrow [0, 1] \qquad (4)$$

where UxV is the Cartesian product of U and V. A fuzzy relation R in UxV is expressed as

Mizumoto, [illegible]

$$R = \{\mu_R(u_1,v_1)/\langle u_1,v_1\rangle, \mu_R(u_1,v_2)/\langle u_1,v_2\rangle, \ldots, \mu_R(u_m,v_n)/\langle u_m,v_n\rangle\} = \sum_{i,j} \mu_R(u_i,v_j)/\langle u_i,v_j\rangle \quad (6)$$

where u_i, $i=1,2,\ldots,m$, and v_j, $j=1,2,\ldots,n$, represent the elements of U and V, respectively, and $\langle u_i,v_j\rangle$ stands for an ordered pair of u_i and v_j, i.e., an element of UxV.

More generally, an n-ary fuzzy relation R in $U_1xU_2x\ldots xU_n$ is a fuzzy relation which is characterized by an n-variate membership function ranging over $U_1xU_2x\ldots xU_n$.

Since Zadeh first formulated the concept of fuzzy sets and fuzzy relations, some extensions have been described, for example, L-fuzzy sets[10] by Goguen, level m fuzzy sets[11] and type n fuzzy sets[12,13] by Zadeh.

L-fuzzy sets are a generalization of the membership space from the interval [0,1] to a lattice L. The universe of discourse of a level m fuzzy set may be the set of level m-1 fuzzy sets with understanding that level 1 fuzzy sets are ordinary fuzzy sets. For type n fuzzy sets, the values of their membership functions are type n-1 fuzzy sets of the interval [0,1] rather than points of [0,1]. Type 1 fuzzy sets are equivalent to ordinary fuzzy sets.

More formally, we have the following definition.

Definition 3. An L-fuzzy set X, a level m fuzzy set Y (m=1,2,...) and a type n fuzzy set Z (n=1,2,...) in U are characterized by the following membership functions μ_X, μ_Y and μ_Z, respectively.

$$\mu_X : U \longrightarrow L \quad (7)$$

$$\mu_Y : [0,1]^{[0,1]^{\cdot^{\cdot^{m-1}}}} \longrightarrow [0,1] \quad (8)$$

$$\mu_Z : U \longrightarrow [0,1]^{[0,1]^{\cdot^{\cdot^{n}}}} \quad (9)$$

where L represents a lattice and A^B the set of all functions from B to A.

An L-fuzzy relation, a level m fuzzy relation and a type n fuzzy relation are easily defined by the same generalization to an ordinary fuzzy relation.

Now we shall have some examples of various fuzzy sets.

Example 1. Assume that

$$U = \{a, b, c, d\}. \quad (10)$$

Then we may have a fuzzy set F in U as

$$F = \{0.1/a, 0.8/b, 0.9/c\} \quad (11)$$

and a fuzzy relation R in UxU as

$$R = \{0.3/\langle a,b\rangle, 0.9/\langle b,d\rangle\}. \quad (12)$$

Example 2. For U defined in (10), we have

$$X = \{\langle 0.1,0.9\rangle/a, \langle 0.8,1\rangle/b, \langle 0.9,0\rangle/c\} \quad (13)$$

as an L-fuzzy set in U.

For the same U, if two fuzzy sets in U are expressed as, say,

$$Y1 = \{0.3/a, 0.2/b, 0.9/d\} \quad (14)$$

$$Y2 = \{0.6/a, 0.1/b\} \quad (15)$$

then we would have

$$Y = \{0.6/Y1, 0.1/Y2\} \quad (16)$$

as a level 2 fuzzy set in U.

Moreover, for the same U, we may have a type 2 fuzzy set:

$$Z = \{\underline{high}/a, \underline{middle}/b, \underline{low}/d\} \quad (17)$$

where high, middle and low are assumed to be fuzzy sets in $\{0, 0.1, 0.2, \ldots, 1\} \subseteq [0,1]$ and, for example, are expressed as follows.

$$\underline{high} = \{1/1, 0.8/0.9, 0.4/0.8\} \quad (18)$$

$$\underline{middle} = \{1/0.5, 0.5/0.6, 0.5/0.4\} \quad (19)$$

$$\underline{low} = \{1/0, 0.8/0.1, 0.4/0.2\} \quad (20)$$

Note. In (13) L is a lattice $[0,1]x[0,1]$ ordered by

$$\langle a_1,b_1\rangle \leq \langle a_2,b_2\rangle \iff a_1 \leq a_2 \text{ and } b_1 \leq b_2$$

where $a_1, a_2, b_1, b_2 \in [0,1]$. In this paper, L-fuzzy sets mean only in the case that L is $[0,1]x[0,1]x\ldots x[0,1]$ with an ordered relation such that

$$\langle a_1,a_2,\ldots,a_n\rangle \leq \langle b_1,b_2,\ldots,b_n\rangle \iff a_i \leq b_i \ (i=1,2,\ldots,n)$$

where a_i and b_i are elements of $[0,1]$.

FUZZY-SET-THEORETIC DATA STRUCTURE

In this section we shall describe "Fuzzy-Set-Theoretic Data Structure (FSTDS)" which is an extended version of "Set-Theoretic Data Structure (STDS)" by Childs[3,4].

FSTDS is a data structure representing fuzzy sets and fuzzy relations to be manipulated conveniently and efficiently by fuzzy set operators.

FSTDS is made up of eight areas as follows:

(1) Fuzzy Set Area (FSA)
(2) Fuzzy Set Representation Area (FSRA)
(3) Grade Area (GA)
(4) Grade Tuple Area (GTA)
(5) Element Area (EA)
(6) Element Tuple Area (ETA)
(7) Fuzzy Set Name Area (FSNA)
(8) Fuzzy Set Operator Name Area (FSONA)

A fuzzy set operation, which is represented by a procedure, accesses fuzzy sets through the pointers in Fuzzy Set Area (FSA). Fuzzy set operator will be presented later. In what follows, we shall discuss each area of FSTDS in detail (see Fig. 1 -4).

(1) Fuzzy Set Area (FSA)

This area is a collection of the pointers to the representations of each fuzzy set and fuzzy relation. Given a pointer in this area, it is possible to access all information associated with any fuzzy set. Most fuzzy set operations which operate on only fuzzy sets have the pointers to this area as operands. Each data cell of FSA is a pair of pointers, one to the Fuzzy Set Representation Area (FSRA), the other to the Fuzzy Set Name Area (FSNA). Pointers to FSNA are not needed for most fuzzy set operations, but they are required for some, such as the output of type n fuzzy sets.

(2) Fuzzy Set Representation Area (FSRA)

This area is a collection of fuzzy set representations. A fuzzy set representation consists of the number n ($n \geq 0$) of grade/element pairs in one fuzzy set, a grade part which contains n pointers of grades, and an element part which has n pointers of elements. For ordinary fuzzy set consisting of n grade/element pairs, the element

Mizumoto

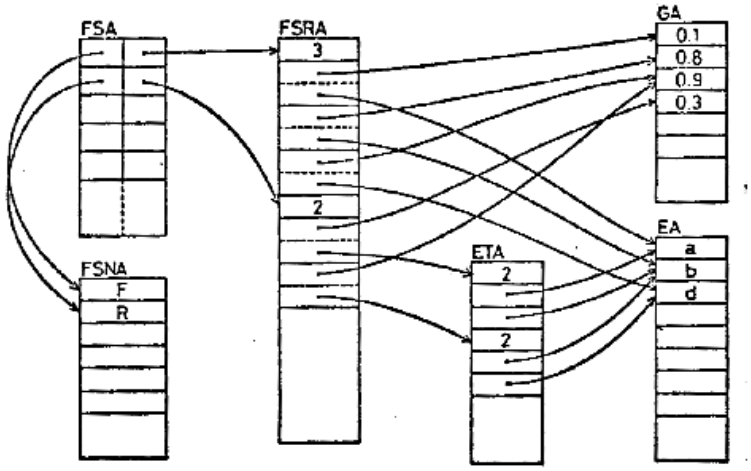

Fig. 1 The representation of a fuzzy set F (11) and a fuzzy relation R (12) by FSTDS

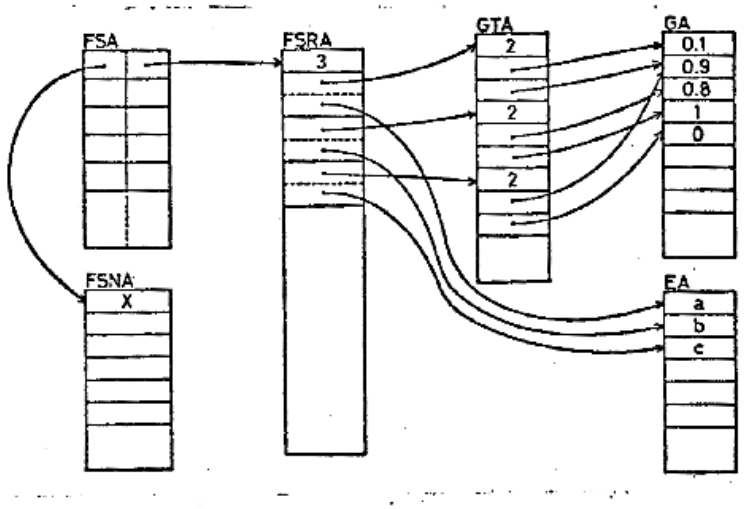

Fig. 2 The representation of an L-fuzzy set X (13) by FSTDS

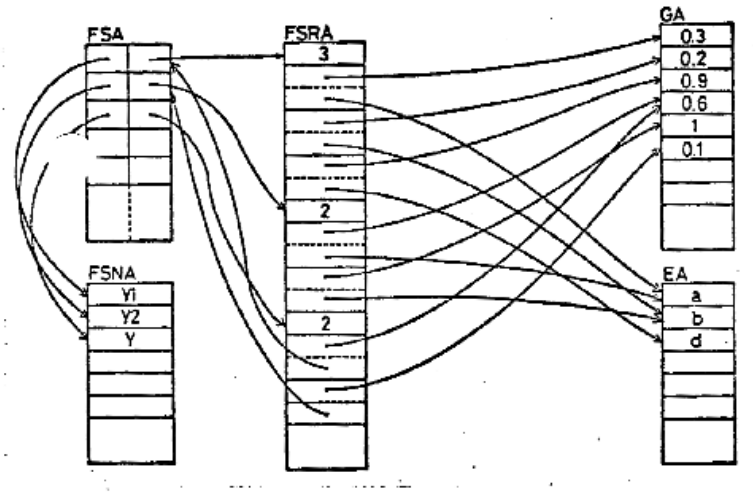

Fig. 3 The representation of a level 2 fuzzy set Y (16) by FSTDS

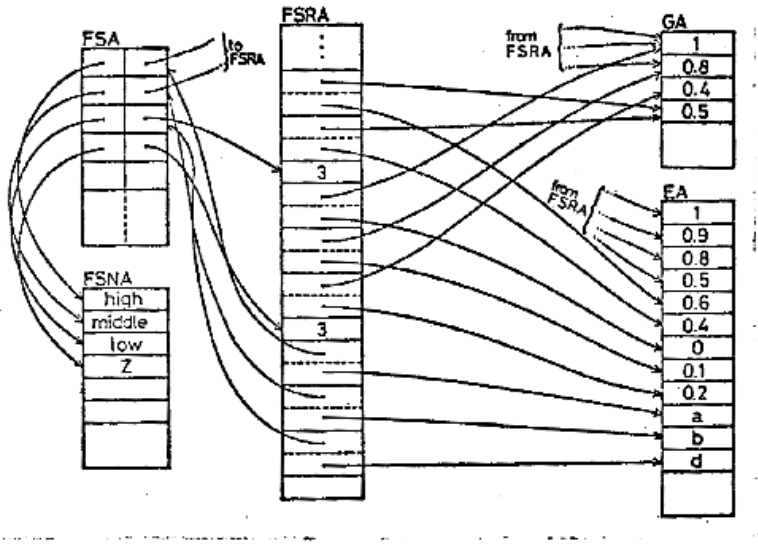

Fig. 4 The representation of a type 2 fuzzy set Z (17) by FSTDS

part contains n pointers to the Element Area (EA), and the grade part contains n pointers to the Grade Area (GA), corresponding to the element pointer of the element part. From another point of view, the grade part and the element part are considered as one grade/element part which has n grade/element pointer pairs.

The pointer of the element part can be to the Element Tuple Area (ETA) and Fuzzy Set Area (FSA) in the case of a fuzzy relation and a level m fuzzy set, respectively. On the other hand, the pointer of the grade part can be to the Grade Tuple Area (GTA) and FSA for an L-fuzzy set and a type n fuzzy set, respectively.

An ordinary set is represented as a special case of an ordinary fuzzy set, that is, all pointers of the grade part are to 1 in GA. It should be noted that any elements whose grade values are 0 are omitted from a fuzzy set representation.

Note. We shall hereafter use the term "fuzzy set" as a generic name expressing not only ordinary fuzzy set but also ordinary set, L-fuzzy set, level m fuzzy set and type n fuzzy set and occasionaly even fuzzy relation. The term "fuzzy relation" is used similarly to express various kinds of fuzzy relations.

(3) Grade Area (GA)

This area is a collection of the numbers in the interval [0,1] which are the values of membership functions (i.e., membership grades).

(4) Grade Tuple Area (GTA)

The Grade Tuple Area is a collection of n tuples of the values of membership functions for, say, an L-fuzzy set. Each n-tuple definition consists of the number n ($n \geq 1$) (the length of the tuple) and n pointers which may refer to Grade Area (GA) for an ordinary L-fuzzy set, to Fuzzy Set Area (FSA) for an L-type n fuzzy set or again to Grade Tuple Area (GTA) for a higher order L-fuzzy set.

Note. More generalized fuzzy set will be found in Example 5.

(5) Element Area (EA)

The Element Area is a collection of element names, that is, all names of elements in the universe of discourse. An element name may be a character string of an arbitrary length or a real number. In our system, if the element name can be interpreted as a number, then it is considered as a real number. Otherwise it is considered to be a character string. For example, all of the element names .123, +0.123 and +00.123000 are equal, but ..123 and ..1230 are not.

(6) Element Tuple Area (ETA)

This area is a collection of n-tuples of elements, for example, the elements of an n-ary fuzzy relation. Each n-tuple definition of this area consists of the number n ($n \geq 1$) (the length of the tuple) and n pointers which can refer to the Element Area (EA) for an ordinary fuzzy relation, to the Fuzzy Set Area (FSA) for a level m fuzzy relation or again to the Element Tuple Area (ETA) for a fuzzy relation of fuzzy relations.

Mizumoto, M

(7) Fuzzy Set Name Area (FSNA)

This area is a collection of fuzzy set names. A fuzzy set name is implicitly defined by assigning a fuzzy set to it. In other words, a character string to which a fuzzy set has been once assigned is considered as a fuzzy set name, but it is not declared explicitly as a fuzzy set name. A fuzzy set name is considered as a variable whose value is a fuzzy set rather than a number or a character string. The data cell of this area contains the storage representstion of a name as a character string and a pointer to Fuzzy Set Area (FSA).

This area is used not only in the case where fuzzy set names are needed by fuzzy set operations, e.g., output of type n fuzzy sets but also in the case where the interpreter looks for a pointer to FSA given a fuzzy set name.

(8) Fuzzy Set Operator Name Area (FSONA)

This area is a collection of fuzzy set operator names. The data cell of this area has the same structure as that of the Fuzzy Set Name Area (FSNA) except that a pointer to the Fuzzy Set Area (FSA) is in the internal code of the fuzzy set operator. This area may be omitted from FSTDS from the point of view that FSTDS is only a representation of fuzzy sets and fuzzy relations. This area, however, is necessary since it facilitates implementation of FSTDS system and gives added flexibility to i[illegible]

FSTDS consists of the above eight areas. We shall next present some examples of FSTDS.

Example 3. The fuzzy set F and the fuzzy relation R defined by (11) and (12), respectively, in Example 1 are represented by FSTDS in Fig.1.

Example 4. The L-fuzzy set X, the level 2 fuzzy set Y, and the type 2 fuzzy set Z defined by (13), (16) and (17), respectively, in Example 2 are represented by FSTDS in Fig. 2, 3 and 4, respectively.

As illustrated in Fig. 1 - Fig. 4, the same grade values in Grade Area (GA) and Grade Tuple Area (GTA) and the same element values in Element Area (EA) and Element Tuple Area (ETA) are shared in FSTDS, respectively. But the common subset is not shared as in STDS[3,4]. This is because several ordinary sets can be easily divided into disjoint ones, but it is impossible to divide fuzzy sets uniquely.

With FSTDS it is possible to represent not only ordinary fuzzy sets, L-fuzzy sets, level m fuzzy sets and type n fuzzy sets, but also more complex fuzzy sets which we call "generalized fuzzy sets". By a generalized fuzzy set, we mean the fuzzy set which is obtained by combining various kinds of fuzzy sets denoted in the previous section. For example, an L-type 3 fuzzy set, a level 5 type 2 fuzzy relation, and an L-level 7 type 5 fuzzy set are considered as generalized fuzzy sets.

Example 5. Let U be a universe of discourse expressed by

$$U = \{a, b, c, d\} .$$

If fuzzy sets Y1 and Y2 in U and high, middle and low in [0,1] are defined by (14), (15), (18), (19) and (20), respectively, then we may have

$$W = \{<high,middle,1>/<Y1,a>, <middle,low,0>/<Y1,b>, <low,high,0.6>/<Y2,c>\} \quad (21)$$

as a generalized fuzzy set in UxU, more exactly, an L-level 2 type 2 fuzzy relation in UxU.

FSTDS is general enough to represent more imaginative fuzzy sets than the ones above. We can define in FSTDS a fuzzy set V as

$$V = \{0.1/a, <low,1>/Y1, <high,low,1>/<middle,high,3.56>\}. \quad (22)$$

Thus, it is not necessary in FSTDS that all elements, or grades, within one fuzzy set have the same type.

There are several methods of mapping from FSTDS to a storage structure. In our case, as we implement it in FORTRAN because of its high portability, the storage structure means what data type in FORTRAN is suitable for FSTDS. Consequently, we have no choice but to use arrays as a storage structure. It seems to be natural that the data cell of each area occupies contiguous array components. As for the representation of a whole area, we have alternative choices concerning how many arrays are required , that is, several small arrays (i.e., an array represents one area only), or only one large array (i.e., only one array is partitioned to represent all areas). We have decided on one (integer) array rather than several. For this choice would seem to result in the greatest flexibility and the most economical use of computer memory.

One integer array is initially divided into many blocks of the same size (We can specify the size of blocks in the head of FSTDSL program. The defaulted size is 100 now). A block is occupied by one specific area. If an area demands more storage space, a new block is supplied to this area and connected to it by a pointer.

The merits of this method are that few reallocations of areas are necessary and no pointers are required for the connection of data cells. It is also a merit that since most data cells in the same area are contiguous, we can have a high locality of memory references in searching one specific area, so we may attain a high performance even in a paging environment by adjusting the size of block to that of a page.

We can use FSTDS as a data structure for representing fuzzy sets and fuzzy relations. But it is somewhat troublesome and a source of error if a user has to manage and manipulate many sorts of pointers in FSTDS. We therefore give a method by which a user can define and manipulate fuzzy sets and fuzzy relations without worrying about the pointers in FSTDS. In other words, it is possible for a user to define and manipulate fuzzy sets and relations easily with no attention to their representations in a computer, or even in FSTDS.

We shall turn our attention in the next section to this method, which may be considered as a command or a programming language for the manipulation of fuzzy sets and fuzzy relations.

FUZZY-SET-THEORETIC DATA STRUCTURE SYSTEM

In this section we shall describe FSTDS language (FSTDSL, for short) which is an expression language which enables a user to make use of FSTDS, and FSTDS system which interprets and executes a program written in FSTDSL. FSTDS system can be considered as an FSTDSL processor.

In FSTDSL, we can write an "expression" whose general form is similar to that of a function call in FORTRAN, that is,

$$opr\ (opd_1, opd_2, \ldots, opd_n); \quad (23)$$

where the opr is a fuzzy set operator name and the opd_i are its operands. The number of operands is dependent on a fuzzy set operator (see Table 1). The opd_i may be either a fuzzy set or a grade/element pair. Since any depth of nested operation is feasible, the opd_i may be again of the form of (23) instesd of a fuzzy set.

In addition to the expression, we may have a "statement" that the finite times of <set name>:= are followed by the expression (23), that is,

$$v_1:=v_2:=\ldots=v_m:=opr(opd_1, opd_2, \ldots, opd_n); \quad (24)$$

where the opr and opd_i (i=1,2,..,n) are the same as (23), and v_j (j=1,2,..,m) are set names and a symbol ":=" means the assignment operation. By an entire statement, (24) means that the value of expression is computed and assigned to all set names $v_1, v_2, \ldots, v_m$.

Nozaki, F.4

Table I Fuzzy Set Operators Available in FSTDS System

fuzzy set operators	#opd	remarks
SET($u_1,u_2,\ldots,u_n$)	$n\geq 0$	construct ordinary set
FSET($\mu_1/u_1,u_2/u_2,\ldots,u_n/u_n$)	$n\geq 0$	construct fuzzy set
ASSIGN(Y,X)	2	assign X to Y (same as Y:=X)
UNION($X_1,X_2,\ldots,X_n$)	$n\geq 2$	union of $X_1,X_2,\ldots,X_n$
INTERSECTION($X_1,X_2,\ldots,X_n$)	$n\geq 2$	intersection of $X_1,X_2,\ldots,X_n$
PROD($X_1,X_2,\ldots,X_n$)	$n\geq 2$	product of $X_1,X_2,\ldots,X_n$
ASUM($X_1,X_2,\ldots,X_n$)	$n\geq 2$	algebraic sum of $X_1,X_2,\ldots,X_n$
ADIF($X_1,X_2,\ldots,X_n$)	$n\geq 2$	algebraic difference of $X_1,X_2,\ldots,X_n$
BSUM($X_1,X_2,\ldots,X_n$)	$n\geq 2$	bounded sum of $X_1,X_2,\ldots,X_n$
BDIF($X_1,X_2,\ldots,X_n$)	$n\geq 2$	bounded difference of $X_1,X_2,\ldots,X_n$
UNIONA(X)	1	operate on all fuzzy sets over the domain of the operand set X
INTERSECTIONA(X)	1	
PRODA(X)	1	
ASUMA(X)	1	
ADIFA(X)	1	
BSUMA(X)	1	
BDIFA(X)	1	
OSE($R_1,R_2,\ldots,R_n$)	$n\geq 2$	composition of $R_1,R_2,\ldots,R_n$
CONVERSE(R)	1	converse relation of R
IMAGE(R,X)	2	image of X under R
CIMAGE(R,X)	2	converse image of X under R
DOMAIN(R)	1	domain of R
RANGE(R)	1	range of R
CP($X_1,X_2,\ldots,X_n$)	$n\geq 2$	Cartesian product of $X_1,X_2,\ldots,X_n$
RS(R,X)	2	restriction of R to X
RELATION(X)	1	translate level m fuzzy set X to fuzzy relation

fuzzy set operators	#opd	remarks
EQ(X_1,X_2)	2	Is X_1 equal to X_2?
SUBSET(X_1,X_2)	2	Is X_1 a subset of X_2?
DISJOINT($X_1,X_2,\ldots,X_n$)	$n\geq 2$	Are $X_1,X_2,\ldots,X_n$ disjoint from each other?
ELEMENT(u/u,X)	2	Is u/u an element of X?
CUT(u_1/μ_2,X)	2	$A_{\mu_2}^{\mu_1}$
SOP(u/n,X)	2	scalar operation of u and X
EXP(u/x,X)	2	$X^x_A u$
DIL(X)	1	dilation
CON(X)	1	concentration
CINT(X)	1	contrast intensification
NORM(X)	1	normalization of X
CD(X)	1	cardinality of X
#(X)	1	the number of elements of X
MAXG(X)	1	the maximum grade of X
SF(X,K)	2	support fuzzification of X by K
GF(X,K)	2	grade fuzzification of X by K
DLT($X_1,X_2,\ldots,X_n$)	$n\geq 1$	delete $X_1,X_2,\ldots,X_n$ from system
PRINT($X_1,X_2,\ldots,X_n$)	$n\geq 1$	print out $X_1,X_2,\ldots,X_n$
PRINTB($X_1,X_2,\ldots,X_n$)	$n\geq 1$	print out $X_1,X_2,\ldots,X_n$ in Boolean type
PRINTS($X_1,X_2,\ldots,X_n$)	$n\geq 1$	print out $X_1,X_2,\ldots,X_n$ in set type
PRINTN($X_1,X_2,\ldots,X_n$)	$n\geq 1$	print out $X_1,X_2,\ldots,X_n$ with names
PRINTL(character string)	1	print out character string
DUMP($\alpha_1,\alpha_2,\ldots,\alpha_n$)	$n\geq 1$	dump areas in FSTDS
SNAP(α)	1	print out all fuzzy sets
PARA($\alpha_1=\beta_1,\alpha_2=\beta_2,\ldots,\alpha_n=\beta_n$)	$n\geq 1$	specify the options
END(X)	1or0	evaluate X and halt

1. The symbols (with subscripts) in Table I represent the following meanings:

   u: an element, that is, a real number, a character string or a fuzzy set, or an n-tuple of them.
   μ: a grade, that is, a number in the interval [0,1] or a fuzzy set, or an n-tuple of them.
   X: an expression or a fuzzy set
   Y: a fuzzy set or a fuzzy set to be defined.
   R: a fuzzy relation
   χ: a set of fuzzy sets.
   n: an integer.
   x: a real number
   α: an alphabetical character
   K: a kernel set
   β: an option of PARA operator

2. For SET and FSET operators, SET() and FSET(), i.e., n=0, mean the empty set.
3. For SOP operator, the n represents:

   1: maximum 2: minimum 3: product 4: algebraic sum 5: absolute difference 6: bounded sum
   7: bounded difference

4. For END operator, END can be used for END().

As for fuzzy set operations, FSTDS system computes the grades of fuzzy sets using many kinds of fuzzy set operations.

Thus, the only things a user need do are to analyze a given problem and to define and manipulate fuzzy sets to solve it using the fuzzy set operators provided in FSTDSL. The fuzzy set operators available in FSTDS system are just like built-in functions in other programming languages.

At present FSTDS system provides 52 fuzzy set operators, and a user can solve a problem by the use of these operators. Table I lists all fuzzy set operators available now. The definitions of these operators on fuzzy sets and fuzzy relations are briefly presented in the Appendix. For more detailed discussions, see [1, 2, 10-13, 15, 17-19].

Next, we shall present some examples written in FSTDSL.

Example 6. In FSTDSL, we can write

```
U:=SET(A,B,C,D);
F:=FSET(0.1/A, 0.8/B, 0.9/D);
R:=FSET(0.3/<A,B>, 0.9/<B,D>);
```

to define the ordinary set U, the fuzzy set F and the fuzzy relation R in Example 1. FSTDS system interpretes above statements and sets up FSTDS shown in Fig.1. Note that Fig.1 does not contain the representation of U on account of limited space.

Example 7. We can define easily L-fuzzy sets, level m fuzzy sets and type n fuzzy sets in FSTDSL.

Mizumoto

For the L-fuzzy set defined by (13) in Example 2, we need only write

```
X:=FSET(<0.1,0.9>/A, <0.8,1>/B, <0.9,0>/C);
```

Level m fuzzy sets and type n fuzzy sets can not at present be written in one statement. Thus, we must define the component fuzzy sets of level m fuzzy sets and type n fuzzy sets. For example, the level 2 fuzzy set Y defined by (16) can be written in FSTDSL as follows.

```
Y1:=FSET(0.3/A, 0.2/B, 0.9/D);
Y2:=FSET(0.6/A, 0.1/B);
Y:=FSET(0.6/Y1, 0.1/Y2);
```

The type 2 fuzzy set Z in (17) can be written as

```
HIGH:=FSET(1/1, 0.8/0.9, 0.4/0.8);
MIDDLE:=FSET(1/0.5, 0.5/0.6, 0.5/0.4);
LOW:=FSET(1/0, 0.8/0.1, 0.4/0.2);
Z:=FSET(HIGH/A, MIDDLE/B, LOW/D);
```

Note that in FSTDSL we must define component fuzzy sets first. Moreover, the higher order L-fuzzy sets, the higher level fuzzy sets and the higher type fuzzy sets can be written in the same fashion.

Example 8. The generalized fuzzy set W defined in (21) and the fuzzy set V in (22) can be written in FSTDSL as follows.

First, we define component fuzzy sets:

```
Y1:=FSET(0.3/A, 0.2/B, 0.9/D);
Y2:=FSET(0.6/A, 0.1/B);
HIGH:=FSET(1/1, 0.8/0.9, 0.4/0.8);
MIDDLE:=FSET(1/0.5, 0.5/0.6, 0.5/0.4);
LOW:=FSET(1/0, 0.8/0.1, 0.4/0.2);
```

and then we can define W and V as

```
W:=FSET(<HIGH,MIDDLE,1>/<Y1,A>, <MIDDLE,LOW,0>/<Y1,B>,
        <LOW,HIGH,0.6>/<Y2,C>);
V:=FSET(0.1/A, <LOW,1>/Y1, <HIGH,LOW,1>/<MIDDLE,HIGH,3.56>);
```

Example 9. We can represent a fuzzy directed graph[14] shown in Fig.5 by FSTDSL statements as follows:

```
V:=SET(X,Y,Z,W);
A:=FSET(0.1/<X,Y>, 0.7/<Y,Z>, 0.4/<W,Z>,
    1/<W,Y>, 0.3/<X,W>, 0.9/<W,X>);
G:=SET(<V,A>);
```

where the V represents a set of vertices and the A a fuzzy set of directed arcs, that is, a set of weighted directed arcs, and thus the G represents the entire fuzzy directed graph G.

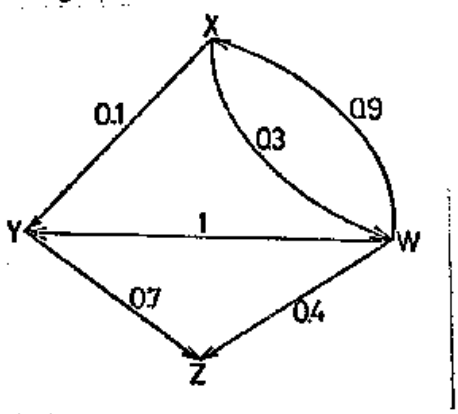

Fig.5 A fuzzy directed graph G

Example 10. Let X and Y be fuzzy sets in U and R a fuzzy relation from U to V. Then we have

$$(X \circ Y) \cup R = (X \cup R) \circ (Y \cup R) \qquad (25)$$

where U denotes the union of fuzzy sets and o the composition of fuzzy relations. But, in this case, X and Y are unary fuzzy relations (i.e., ordinary fuzzy sets), so XoR reduces to the image of X under R.

Suppose that X and Y are defined by

$$X = \{1/a, 0.9/b, 0.3/c\} \qquad (26)$$

$$Y = \{0.1/a, 0.7/b, 0.9/c\} \qquad (27)$$

and R is defined in terms of relation matrix:

$$R = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 1 & 0.8 & 0 \\ 0.7 & 1 & 0.2 \\ 0 & 0.5 & 0.1 \end{bmatrix} \end{array} \qquad (28)$$

Then, we can write the program as follows.

```
X:=FSET(1/A, 0.9/B, 0.3/C);
Y:=FSET(0.1/A, 0.7/B, 0.9/C);
R:=FSET(1/<A,A>, 0.8/<A,B>, 0.7/<B,A>, 1/<B,B>,
                  0.2/<B,C>, 0.5/<C,B>, 0.1/<C,C>);
PRINT(ASSIGN(Z,UNION(X,Y)));
PRINT(IMAGE(R,Z));
V:=IMAGE(R,X);   W:=IMAGE(R,Y);
PRINT(UNION(V,W));
END;
```

and the execution result of above program is

```
FSET(1/A, 0.9/B, 0.9/C);    ... X U Y
FSET(1/A, 0.9/B, 0.2/C);    ... (X U Y) o R
FSET(1/A, 0.9/B, 0.2/C);    ... (X U R) o (Y U R)
```

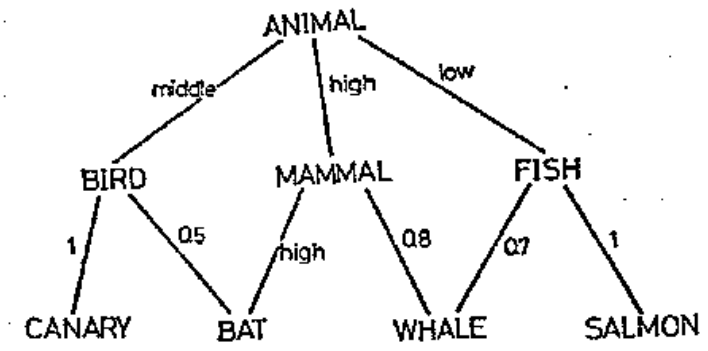

Fig.6 An example of fuzzy knowledge

Example 11. The fuzzy knowledge shown in Fig.6 is represented in FSTDSL by the following level 2 type 2 fuzzy set ANIMAL.

```
LOW:=FSET(1/0, 0.8/0.1, 0.4/0.2);
MIDDLE:=FSET(1/0.5, 0.5/0.6, 0.5/0.4);
HIGH:=FSET(1/1, 0.8/0.9, 0.4/0.8);
BIRD:=FSET(1/CANARY, 0.5/BAT);
MAMMAL:=FSET(HIGH/BAT, 0.8/WHALE);
FISH:=FSET(0.7/WHALE, 1/SALMON);
ANIMAL:=FSET(MIDDLE/BIRD, HIGH/MAMMAL, LOW/FISH);
```

The question "What does a BAT belong to ?" will be translated into FSTDSL statements as

```
ISA:=CONVERSE(RELATION(ANIMAL));
X:=IMAGE(ISA,SET(BAT));
```

where RELATION is a fuzzy set operator by which a level m fuzzy set is translated into a fuzzy relation and CONVERSE is that of a converse relation.

Then the output of above X (i.e., PRINT(X);) is

```
FSET(0.5/BIRD, HIGH/MAMMAL);
```

Thus we have the answer "A BAT belongs to BIRDs with the compatibility 0.5 and to MAMMALs with high compatibility".

As shown in Example 6 - 11, FSTDSL has a simple syntax. But it seems to be very important that a user should become familiar with many kinds of fuzzy set operators. The following should be noted.

First, if one encounters the FSTDSL statement:

X:=FSET(0.3/BIRD); (29)

one may not know exactly whether the BIRD is a fuzzy set name (i.e., X is defined as a level m fuzzy set) or an element name (i.e., X is defined as ordinary fuzzy set). This is understood from the fact that when the statement (29) is interpreted by FSTDS system, if a fuzzy set has been already assigned to the character string BIRD, then BIRD is a fuzzy set name; otherwise it is an element name.

The reason for this unusual manner is that we would not like to put a character string representing an element name in the quotation marks (e.g., FSET(0.3/'BIRD')). It is very annoying to make a program and punch it using a lot of quotation marks. In order to overcome, however, the difficulty of distinction between a fuzzy set name and an element name, the user will be recommended to ensure that fuzzy set names are different from element names, though the same names are permitted. In fact, every problem can be easily formulated in this fashion and it makes the program very easy to read and understand even in usual programming languages.

Each fuzzy set operator has a restriction on the numb of operands and the types of operands, and the operands are interpreted in the predefined order. For example, given the statement (29), X becomes a set name since X is followed by the assignment symbol := . FSET is interpreted as a fuzzy set operator name. The character string 0.3 can be interpreted as a number in the interval [0,1]. As for the character string BIRD, if BIRD has been already defined as a fuzzy set name (i.e., BIRD:=.....; has occured before), then BIRD is interpreted as a fuzzy set name, otherwise BIRD is an element name. If an unexpected number of operands occur, or if their type is incorrect, then the program goes into an error state.

Note. The term "type" in this context is different from that in "type n fuzzy set". In this context, we use the term "type" to express whether a given operand is an ordinary set, an ordinary relation, an ordinary fuzzy set, an ordinary fuzzy relation, an L-fuzzy set, a level m fuzzy set or a type n fuzzy set and so on.

Second, the omission of quotation marks causes all space symbols to be ignored in FSTDSL, so a user must use the special symbol # denoting a space if it is nec ry to output spaces (see Fig.7).

Third, an expression of FSTDSL always has as value a fuzzy set rather than a number, a truth value or a character string. So does a statement.

FSTDS system consists of a simple interpreter, a set of fuzzy set operations and a data structure FSTDS.

At the present time, the interpreter is designed to interprete one FSTDSL statement at a time and invoke a sequence of necessary fuzzy set operations. Once a fuzzy set operation is invoked, its fuzzy set operation sets up or manipulates fuzzy sets in the data structure FSTDS and returns control to the interpreter except END operation, and then the interpreter interpretes the next statement until END operation occurs.

As was shown in Example 6 - 11, FSTDSL is designed to have no labels and no control structure (e.g., IF ... THEN ... ELSE ..., GO TO ... , or WHILE ... DO ... etc). This is not only because FSTDSL processor can be implemented easily but also FSTDS system have another user interface, that is, the connection of FSTDS and FORTRAN. If a user wants to use a control structure, he may make use of that of FORTRAN. We shall next demonstrate this useful facility.

Example 12. The program in FSTDSL and FORTRAN shown as

```
      C *** EXAMPLE 12 ***     ;;;;;; FORTRAN/FSTDSL
  1   F    PARA(G=1)
  2        N=5
  3   F    LARGE=U=EMPTY
  4        DO 10 I=1,N
  5        G=FLOAT(I)/FLOAT(N)
  6   F    LARGE=UNION(LARGE, FSET(!!G/!!))
  7   F    U=UNION(U,SET(!!I))
  8     10 CONTINUE
  9   F    PRINTN(U,LARGE)
 10   F    NOT_LARGE=ADIF(U,LARGE); PRINTN(NOT_LARGE)
 11        STOP
 12        END
```

causes to the results:

```
U=FSET(1.0/1, 1.0/2, 1.0/3, 1.0/4, 1.0/5);
LARGE=FSET(0.2/1, 0.4/2, 0.6/3, 0.8/4, 1.0/5);
NOT_LARGE=FSET(0.8/1, 0.6/2, 0.4/3, 0.2/4);
```

FSTDSL can be embedded in FORTRAN as in the above program. In this case, one must put the character F at the head of an FSTDSL statement (i.e., the first column for cards) to differentiate an FSTDSL statement from a FORTRAN statement and put one exclamation mark ! and two !! followed by FORTRAN integer and real variables, respectively, which are indicating its value inside an FSTDSL statement.

The above program shows how to define a universe of discourse U (i.e., an ordinary set) and a fuzzy set LARGE, and compute the complement of LARGE (i.e., NOT_LARGE) and output them. In more detail, we first make a set U and a fuzzy set LARGE empty (line 3) and then compute a grade value G of the element I and add the G/I pair to LARGE and the I to U for I=1,2,...,5 (from line 4 to 8) and then output the set U and the fuzzy set LARGE together with fuzzy set names (line 9) and compute the absolute difference of LARGE from U (i.e., the complement of LARGE) and output it with fuzzy set name (line 10).

A program written in FSTDSL/FORTRAN is translated into the FORTRAN program by FSTDS translator, that is, an FSTDSL statement is expanded into several CALL statements in FORTRAN. Then they are compiled, linked with SUBROUTINEs in the library for the interface between FSTDSL and FORTRAN, and executed.

In order to match FSTDSL syntax with that of FORTRAN, the FSTDSL syntax is slightly modified as follows.

First, the prefix symbols ! and !! indicate FORTRAN integer and real variables, respectively, in FSTDSL statement. In FSTDSL, for example, the expression SET(X) represents an ordinary set {X}, while SET(!!X) represents {3.14} with X=3.14.

Second, the end of line (i.e., column 72 for cards) means the end of statement. We may, therefore, omit the last semicolon ; of lines. Instead of this omission, we must use column 6 for continuation.

Third, the assignment symbol := can be reduced to the FORTRAN assignment symbol =.

The data are passed from FORTRAN to FSTDSL by the use of the above ! and !! facilities. Conversely, passing them from FSTDSL to FORTRAN is not so easy because FORTRAN can not deal with even ordinary sets. So we have chosen the facilities to pass them element by element. Such facilities are now provided by CALLing specific SUBROUTINEs directly in FORTRAN. So there are no changes to FSTDSL and FORTRAN syntax. Some examples of such SUBROUTINEs are to get each grade with its type in a specified fuzzy set, to get each element with its type in a specified fuzzy set and to get the number of elements in a specified fuzzy set .

The connection of FSTDSL and FORTRAN greatly extends the capability and the applicability of FSTDSL. From an opposite point of view, this allows the provision in FORTRAN of facilities to define and manipulate fuzzy sets

Mizumoto, [illegible]

```
FSTDS SYSTEM SOURCE LIST

      C ***** LINGUISTIC HEDGES AND APPROXIMATE REASONING *****
 1    F      /,,,10/
 2           WRITE(6,200)
 3       200 FORMAT(1H1)
      C *** GET PRIMARY TERMS (SMALL, MIDDLE, LARGE) ***
 4    F      U=SMALL=MIDDLE=LARGE=EMPTY
 5           DO 10 I=1,7
 6    F      U=UNION(U,SET(!I))
 7           GSMALL=Z(FLOAT(I),4.0,2.5,1.0)
 8    F      SMALL=UNION(SMALL,FSET(!!GSMALL/!I))
 9           GMDDLE=PI(FLOAT(I),2.0,4.0)
10    F      MIDDLE=UNION(MIDDLE,FSET(!!GMDDLE/!I))
11           GLARGE=S(FLOAT(I),4.0,5.5,7.0)
12    F      LARGE=UNION(LARGE,FSET(!!GLARGE/!I))
13        10 CONTINUE
14    F      PRINTC(/**#PRIMARY#TERMS#**/)
15    F      PRINTC(U=); PRINTS(U)
16    F      PRINTN(SMALL,MIDDLE,LARGE)
      C *** COMPUTE HEDGE EFFECTS ***
17    F      VERY_LARGE=CON(LARGE)
18    F      MORE_OR_LESS_SMALL=DIL(SMALL)
19    F      SLIGHTLY_SMALL=NORM(INTERSECTION(SMALL,ADIF(U,CON(SMALL))))
20    F      SORT_OF_SMALL=NORM(INTERSECTION(
             *                              ADIF(U,CON(CON(SMALL))),DIL(SMALL)))
21    F      PRETTY_LARGE=NORM(INTERSECTION(
             *                              CINT(LARGE),ADIF(U,CINT(CON(LARGE)))))
22    F      PRINTC(//**#RESULT#OF#HEDGE#EFFECTS#**/)
23    F      PRINTN(VERY_LARGE,MORE_OR_LESS_SMALL,SLIGHTLY_SMALL,
             *                                     SORT_OF_SMALL,PRETTY_LARGE)
      C *** APPROXIMATE REASONING ***
      C          P1: X IS SMALL.
      C          P2: X AND Y ARE APPROXIMATELY EQUAL (AE).
24    F      V=U
25    F      AE=EMPTY
26           DO 20 I=1,7
27           DO 20 J=1,7
28           GAE=PI(FLOAT(I-J),3.0,0.0)
29    F      AE=UNION(AE,FSET(!!GAE/(!I,!J)))
30        20 CONTINUE
31    F      X=SMALL
32    F      Y=IMAGE(AE,X)
33    F      PRINTC(//**#APPROXIMATE#REASONING#**//
             *          ##P1:#X#IS#SMALL:/
             *          ##P2:#X#AND#Y#ARE#APPROXIMATELY#EQUAL:/
             *          ##----------------------------------/
             *          ##P3:#Y#IS#)
34    F      PRINT(Y)
      C          P1: X IS SMALL.
      C          P2: IF X IS MORE OR LESS SMALL THEN Y IS LARGE
      C                                         ELSE Y IS SORT OF SMALL
35    F      P=MORE_OR_LESS_SMALL
36    F      Q=LARGE
37    F      S=SORT_OF_SMALL
38    F      RA=UNION(CP(P,Q),CP(ADIF(U,P),S))
39    F      UXV=CP(U,V)
40    F      RB=INTERSECTION(
             *    BSUM(ADIF(UXV,CP(P,V)),CP(U,Q)), BSUM(CP(P,V),CP(U,S)) )
41    F      YA=IMAGE(RA,X)
42    F      YB=IMAGE(RB,X)
43    F      PRINTC(/**#P1:#X#IS#SMALL;/
             *          ##P2:#IF#X#IS#MORE#OR#LESS#SMALL#
             *                  THEN#Y#IS#LARGE#ELSE#Y#IS#SORT#OF#SMALL:/
             *          ##----------------------------------
             *                                    ----------------------------------/)
44    F      PRINTC(/THE#MAXIMIN#RULE:/
             *          ##P3:#Y#IS#); PRINT(YA)
45    F      PRINTC(/THE#ARITHMETIC#RULE:/
             *          ##P3:#Y#IS#); PRINT(YB)
46    F      END
47           STOP
48           END
```

(a) Program

Fig.7 A program in FSTDSL/FORTRAN for the linguistic hedges and the approximate reasoning

```
*** PRIMARY TERMS ***
U=SET(1, 2, 3, 4, 5, 6, 7);
SMALL=FSET(1/1, 0.7778/2, 0.2222/3);
MIDDLE=FSET(0.5/3, 1/4, 0.5/5);
LARGE=FSET(0.2222/5, 0.7778/6, 1/7);


*** RESULT OF HEDGE EFFECTS ***
VERY_LARGE=FSET(0.0493/5, 0.6049/6, 1/7);
MORE_OR_LESS_SMALL=FSET(1/1, 0.8319/2, 0.4713/3);
SLIGHTLY_SMALL=FSET(1/2, 0.5623/3);
SORT_OF_SMALL=FSET(1/2, 0.7432/3);
PRETTY_LARGE=FSET(0.3158/5, 1/6);


*** APPROXIMATE REASONING ***

    P1: X IS SMALL;
    P2: X AND Y ARE APPROXIMATELY EQUAL;
   ------------------------------------
    P3: Y IS FSET(1/1, 0.7778/2, 0.7778/3, 0.2222/4, 0.2222/5);

    P1: X IS SMALL;
    P2: IF X IS MORE OR LESS SMALL THEN Y IS LARGE ELSE Y IS SORT OF SMALL;
   ------------------------------------------------------------------------

THE MAXIMIN RULE:
    P3: Y IS FSET(0.2222/2, 0.2222/3, 0.2222/5, 0.7778/6, 1/7);

THE ARITHMETIC RULE:
    P3: Y IS FSET(0.2222/1, 0.2222/2, 0.2222/3, 0.2222/4, 0.3403/5, 0.7778/6, 1/7);
```

(b) Output

and fuzzy relations, and it therefore greatly extends the application area of FORTRAN. Moreover, the connection of FSTDSL and other programming languages could be easily implemented by writing down FSTDS translator for those languages.

We shall conclude this section with some examples of the applications of FSTDS system to linguistic hedges[15,16] and approximate reasoning[12,17,18].

Example 13. A linguistic hedge such as very, more or less, much, slightly etc. can be viewed as an operator which operates on the operand fuzzy set. For example, the linguistic hedges very, more or less, slightly, sort of and pretty were defined by Zadeh[15] and Lakoff[16] as follows:

$$\underline{\text{very}}\ x = \text{CON}(x) \tag{30}$$

$$\underline{\text{more or less}}\ x = \text{DIL}(x) \tag{31}$$

$$\underline{\text{slightly}}\ x = \text{NORM}(x \cap \neg\text{CON}(x)) \tag{32}$$

$$\underline{\text{sort of}}\ x = \text{NORM}(\neg\text{CON}(\text{CON}(x)) \cap \text{DIL}(x)) \tag{33}$$

$$\underline{\text{pretty}}\ x = \text{NORM}(\text{CINT}(x) \cap \neg(\text{CINT}(\text{CON}(x))) \tag{34}$$

where x stands for a fuzzy set, $\neg$ and $\cap$ the complement and the intersection, respectively, and the other operators are the same as those of FSTDS system.

As for approximate reasoning, Zadeh has proposed the compositional rule of inference which is expressed in symbols as

$$\begin{array}{ll} P_1: & x \text{ is } A \\ P_2: & x \text{ and } y \text{ are } R \\ \hline P_3: & y \text{ is } A \circ R \end{array} \tag{35}$$

where x and y are object names, A is a fuzzy set in U, R is a fuzzy relation in UxV, and $A \circ R$ is the composition of A and R.

If P_2 is a conditional statement such as

$$P_2: \text{ If } x \text{ is } P \text{ then } y \text{ is } Q \text{ else } y \text{ is } S, \tag{36}$$

where x and y are object names and P, Q and S are fuzzy sets in U, V and V, respectively, then it is translated into the relation R of x and y using either the maximin rule of conditional proposition[10] as

$$R = (P \times Q) \cup (\neg P \times S) \tag{37}$$

where x, $\cup$ and $\neg$ stand for the Cartesian product, the union and the complement, respectively, or the arithmetic rule of conditional proposition[10] as

$$R = (\neg(P \times V) \oplus (U \times Q)) \cap ((P \times V) \oplus (U \times S)) \tag{38}$$

where x and $\neg$ are the same as (37), and $\cap$ and $\oplus$ stand for the intersection and the bounded-sum, respectively.

We shall give a program in FSTDSL/FORTRAN and its printing results to Fig.7. First, it defines U, SMALL, MIDDLE and LARGE. Second, it computes very LARGE, more or less SMALL, slightly SMALL, sort of SMALL, and pretty LARGE and outputs them. Third, it infers the consequences by the approximate reasoning:

$$\begin{array}{ll} P_1: & X \text{ is SMALL.} \\ P_2: & X \text{ and } Y \text{ are approximately equal (AE).} \end{array} \tag{39}$$

and by another approximate reasoning:

$$\begin{array}{ll} P_1: & X \text{ is SMALL.} \\ P_2: & \text{If } X \text{ is more or less SMALL} \\ & \text{then } Y \text{ is LARGE else } Y \text{ is sort of SMALL.} \end{array} \tag{40}$$

to compare the maximin rule of conditional proposition (37) and the arithmetic rule of conditional proposition (38).

Note. The functions S(u;a,b,c) and PI(u;b,c) in lines 11 and 9 of Fig.7(a) are S and Π-shaped functions, respectively, defined by Zadeh[17] and the function in line 7 is a Z-shaped function which is the reflection of the S-shaped function about the line u=b. These functions are depicted as follows.

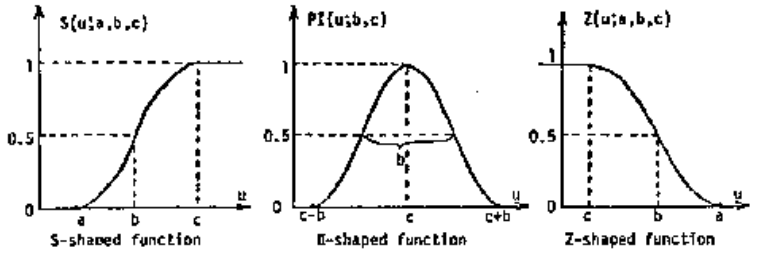

Fig.8 Standard functions available in FSTDS system

It should be noted that in this example the program to compute the hedge effects and to carry out the approximate reasoning is given, but we could write a program to read the propositions, say, in the form of (39) or (40), to analyze their syntax and compute the hedge effects using (30)-(34) and to infer the consequences by the approximate reasoning using (35), (37) and (38) and output the consequences of fuzzy sets, and moreover output the consequences in the linguistic form by the linguistic approximation.

CONCLUSION

We have described FSTDS system in which we can write a program using the concept of fuzzy sets and fuzzy relations.

FSTDS system, in which 52 fuzzy set operators are available, is implemented in FORTRAN, and is currently running on a FACOM 230-45S computer.

This system requires 116 KB including an integer array of size 5,000 for FSTDS. This is because the current version of FSTDS system is an interpreter implementation, so all SUBROUTINEs of the fuzzy set operators must always be linked. These SUBROUTINEs occupy more than half of FSTDS system.

There is a way around this difficulty, namely, the implementation of a compiler version of FSTDS system which reads FSTDSL statements and generates a sequence of fuzzy set operators. In the compiler version, the SUBROUTINEs of unused fuzzy set operations would not be linked and the memory requirements for the execution of an FSTDSL program would, therefore, be decreased.

The processing time for FSTDSL statements is strongly dependent on the number of elements of operand fuzzy sets. It takes 60-70 msec to construct a fuzzy set of several grade/element pairs and assign it to a fuzzy set name. This is because firstly we must manipulate character strings by FORTRAN and search the Grade Area (GA) or the Element Area (EA) etc. to share grades or elements, and secondly a computer must compute the addresses of memory since FSTDS is presented by an array in FORTRAN. But in comparison with the construction of fuzzy sets (i.e., the operations of FSET and SET), the other operations which manipulate only the pointers in FSTDS but search no areas are rather quickly executed. This fact is found by use of the system SUBROUTINE CLOCKM available in FORTRAN on FACOM 230-45S computer. Even rewriting in assembly language those parts of FSTDS system which access FSTDS so frequently will very significantly improve efficiency.

We can define L-fuzzy sets, level m fuzzy sets, type n fuzzy sets, and more generalized fuzzy sets in FSTDSL, but we can not manipulate all of them, since all operations defined for ordinary fuzzy sets cannot be defined for them in nature. Operation methods for some of them have been formulated by Goguen[10] and Zadeh[12], so we are now implementing their facilities.

An FSTDSL program written with only prefix operators is not so readable or understandable, so we are considering the introduction of infix operators. This is attained easily by modifying the interpreter a little.

We have used FSTDS as the representation of fuzzy sets and fuzzy relations, but the consideration of more suitable representation methods may be needed.

To solve a given program, we can write a program in FSTDSL using the concepts of fuzzy sets and fuzzy relations. We can use FSTDS system to construct a fairly large scale system, for we need not pay attention to the representation of fuzzy sets and fuzzy relations and the computations of fuzzy set operations, and we can describe complex and detailed processing in FORTRAN.

As applications of FSTDS system, we are now implementing an Approximate Reasoning System and a Fuzzy Graph Manipulation System.

FSTDS system will find various applications in the fields in which we have to deal with fuzzy information and fuzzy knowledge in nature.

AKNOWLEDGEMENTS

The authors would like to thank Associate Professor J. Toyoda of Osaka University who made several helpful suggestions and Dr. R. Ross who helped them refine their English.

REFERENCES

1. L.A.Zadeh, Fuzzy sets, Information and Control 8, 338-353 (1965).
2. L.A.Zadeh, K.S.Fu, K.Tanaka and M.Shimura (eds.), Fuzzy Sets and Their Applications to Cognitive and Decision Processes, Academic Press, New York (1975).
3. D.L.Childs, Description of a set-theoretic data structure, AFIP Conference Proceedings 33, 557-564 (1968).
4. D.L.Childs, Feasibility of a set-theoretic data structure: A general structure based on a reconstituted definition of a relation, Information Processing 68, 420-430 (1968).
5. J.T.Schwartz, Optimization of very high level language (I): Value transmission and its corollaries, Computer Languages 1, 161-194 (1975).
6. J.T.Schwartz, Optimization of very high level language (II): Deducing relationship of inclusion and membership, Computer Languages 1, 197-218 (1975).
7. J.T.Schwartz, Automatic data structure choice in a language of very high level, Communications of the ACM 18, 722-728 (1975).
8. T.Katayama et al., Logical relation processing language LOREL-1, Journal of Information Processing Society of Japan 15, 326-334 (1975) (in Japanese).
9. N.Wirth, The programming language Pascal, Acta Informatica 1, 35-63 (1971).
10. J.A.Goguen, L-fuzzy sets, Journal of Mathematical Analysis and Applications 18, 145-174 (1967).
11. L.A.Zadeh, Quantitative fuzzy semantics, Information Sciences 3, 159-176 (1971).
12. L.A.Zadeh, The concept of a linguistic variable and its application to approximate reasoning, Information Sciences 8, 199-249; 8,301-357; 9, 43-80 (1975).
13. M.Mizumoto and K.Tanaka, Some properties of fuzzy sets of type 2, Information and Control 31, 312-340 (1976).
14. A.Rosenfeld, Fuzzy graphs, in Fuzzy Sets and Their Applications to Cognitive and Decision Processes (ed. by L.A.Zadeh and K.Tanaka et al.), Academic Press, New York, 77-95 (1975).
15. L.A.Zadeh, A fuzzy-set-theoretic interpretation of linguistic hedges, Journal of Cybernetics 2, 4-34 (1972).
16. G.Lakoff, Hedges: A study in meaning criteria and the logic of fuzzy concepts, Journal of Philosophical Logic 2, 458-508 (1973).

Mizumoto, P.10

17. L.A.Zadeh, Calculus of fuzzy restrictions, in Fuzzy Sets and Their Applications to Cognitive and Decision Processes (ed. by L.A.Zadeh and K.Tanaka et al.), Academic Press, New York, 1-39 (1975).
18. L.A.Zadeh, Fuzzy logic and approximate reasoning, Synthese 30, 407-428 (1975).
19. M.Mizumoto and K.Tanaka, Fuzzy sets and their operations, Information and Control (to appear).

APPENDIX

We shall briefly present the definitions of various kinds of fuzzy set operations. Some operations are n-ary but we will define them as binary operations for simplicity.

The symbols μ, u, F and R, with or without subscripts, are used generally to denote a membership function, an element of a universe of discourse, a fuzzy set and a fuzzy relation, respectively. We use the symbols v and w with or without subscripts as elements of the other universes of discourse. The symbols $\vee$ and $\wedge$ denote the maximum and the minimum, respectively, and $\cdot$, $+$ and $-$ are the ordinary multiplication, addition and substraction, respectively.

(1) Complement:

$$\neg F = \sum_i 1 - \mu_F(u_i)/u_i \qquad (A.1)$$

(2) Union:

$$F_1 \cup F_2 = \sum_i \mu_{F_1}(u_i) \vee \mu_{F_2}(u_i)/u_i \qquad (A.2)$$

(3) Intersection:

$$F_1 \cap F_2 = \sum_i \mu_{F_1}(u_i) \wedge \mu_{F_2}(u_i)/u_i \qquad (A.3)$$

(4) Product:

$$F_1 \cdot F_2 = \sum_i \mu_{F_1}(u_i) \cdot \mu_{F_2}(u_i)/u_i \qquad (A.4)$$

(5) Algebraic Sum: (A.5)

$$F_1 \dot{+} F_2 = \sum_i \mu_{F_1}(u_i) + \mu_{F_2}(u_i) - \mu_{F_1}(u_i) \cdot \mu_{F_2}(u_i)/u_i$$

(6) Absolute Difference:

$$F_1 \dot{-} F_2 = \sum_i |\mu_{F_1}(u_i) - \mu_{F_2}(u_i)|/u_i \qquad (A.6)$$

where $|x|$ denotes the absolute value of real number x.

(7) Bounded-Sum:

$$F_1 \oplus F_2 = \sum_i 1 \wedge (\mu_{F_1}(u_i) + \mu_{F_2}(u_i))/u_i \qquad (A.7)$$

(8) Bounded-Difference:

$$F_1 \ominus F_2 = \sum_i 0 \vee (\mu_{F_1}(u_i) - \mu_{F_2}(u_i))/u_i \qquad (A.8)$$

(9) Composition: (A.9)

$$R_1 \circ R_2 = \sum_{i,k} \bigvee_j (\mu_{R_1}(u_i,v_j) \wedge \mu_{R_2}(v_j,w_k))/\langle u_i,w_k\rangle$$

(10) Converse Relation:

$$R^{-1} = \sum_{i,j} \mu_R(v_j,u_i)/\langle u_i,v_j\rangle \qquad (A.10)$$

(11) Image:

$$R(F) = \sum_j \bigvee_i (\mu_F(u_i) \wedge \mu_R(u_i,v_j))/v_j \qquad (A.11)$$

(12) Converse Image:

$$R^{-1}(F) = \sum_i \bigvee_j (\mu_R(u_i,v_j) \wedge \mu_F(v_j))/u_i \qquad (A.12)$$

(13) Domain:

$$\text{domain}(R) = \sum_i \bigvee_j \mu_R(u_i,v_j)/u_i \qquad (A.13)$$

(14) Range:

$$\text{range}(R) = \sum_j \bigvee_i \mu_R(u_i,v_j)/v_j \qquad (A.14)$$

(15) Cartesian Product:

$$F_1 \times F_2 = \sum_{i,j} \mu_{F_1}(u_i) \wedge \mu_{F_2}(v_j)/\langle u_i,v_j\rangle \qquad (A.15)$$

(16) Restriction:

$$\text{rs}(R,F) = \sum_{i,j} \mu_R(u_i,v_j) \wedge \mu_F(u_i)/\langle u_i,v_j\rangle \qquad (A.16)$$

(17) α-level Set:

$$F_\alpha = \{\, u \mid \mu_F(u) \geq \alpha \,\} \qquad (A.17)$$

(18) Scalar Operation:

$$\mu * F = \sum_i \mu * \mu_F(u_i)/u_i \qquad (A.18)$$

where $*$ denotes an arbitrary binary operation.

(19) Exponentiation:

$$F^x = \sum_i (\mu_F(u_i))^x/u_i \qquad (A.19)$$

where x is a real number.

(20) Dilation:

$$\text{dil}(F) = F^{0.5} = \sum_i \sqrt{\mu_F(u_i)}/u_i \qquad (A.20)$$

(21) Concentration:

$$\text{con}(F) = F^2 = \sum_i (\mu_F(u_i))^2/u_i \qquad (A.21)$$

(22) Contrast Intensification:

$$\text{cint}(F) = \begin{cases} 2 \cdot F^2 & \cdots\ 0 \leq \mu_F(u) \leq 0.5 \\ U \ominus (2 \cdot (U \ominus F)^2) & \cdots\ 0.5 \leq \mu_F(u) \leq 1 \end{cases} \qquad (A.22)$$

(23) Normalization:

$$\text{norm}(F) = \sum_i \mu_F(u_i) \cdot \bar{\mu}^{-1}/u_i \qquad (A.23)$$

where $\bar{\mu}^{-1}$ is the reciprocal of the maximal grade of F.

(24) Cardinality:

$$\text{cd}(F) = \mu_F(u_1) + \mu_F(u_2) + \cdots + \mu_F(u_n) \qquad (A.24)$$

(25) Support Fuzzification:

$$\text{SF}(F;K) = \bigcup_i \mu_F(u_i) \cdot K(u_i) \qquad (A.25)$$

where K is a kernel set of kernels $K(u_1), \ldots, K(u_n)$.

(26) Grade Fuzzification:

$$\text{GF}(F;K) = \sum_i K(\mu_i)/u_i \qquad (A.26)$$

where K is a kernel set of kernels $K(\mu_1), \ldots, K(\mu_n)$.

Mizumoto, P. 11