FSTDS System: A Fuzzy-Set Manipulation System

MOTOHIDE UMANO, MASAHARU MIZUMOTO

and

KOKICHI TANAKA

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

Communicated by Lotfi Zadeh

ABSTRACT

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's STDS (Set-Theoretic Data Structure). The 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 and type-n 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, the 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. The FSTDS system, in which 52 fuzzy-set operations are available, is implemented in FORTRAN, and is currently running on a FACOM 230-45S computer.

1. INTRODUCTION

In the real world, there exist many fuzzy things which cannot or need not be precisely defined. In the past, fuzziness has been studied as vagueness, ambiguity or uncertainty. However, since 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 the mathematical theory of computation [2].

It is well known that ordinary set theory is very useful. Some systems can deal with ordinary sets—for example, STDS developed by Childs [3,4], SETL by Schwartz [5-7], and LOREL by Katayama [8].

 0020-0255/78/0014-0115/$01.75

STDS (Set-Theoretic Data Structure) provides a variety of set operations and is embedded in FORTRAN and MAD. SETL is a set-theoretically oriented language of very high level. As an example of a 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. Only basic operations are available in SETL. LOREL was developed to solve combinatorial problems (e.g., graphs, automata and 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 is 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), an extended version of Childs's STDS. The FSTDS language is considered as a fuzzy-set-theoretically oriented language which can, for example, deal with ordinary sets, ordinary relations, fuzzy sets, fuzzy relations, L–fuzzy sets, level-m fuzzy sets and type-n fuzzy sets. The system is demonstrated using a number of examples.

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

2. 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

$$\mu_F : U \rightarrow [0, 1] \tag{2.1}$$

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

$$F = \{ \mu_F(u_1)/u_1, \mu_F(u_2)/u_2, \ldots, \mu_F(u_n)/u_n \} \tag{2.2}$$

$$= \sum_i \mu_F(u_i)/u_i, \tag{2.3}$$

where u_i, $i = 1, 2, \ldots, n$, 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 $U \times V$ or from U to V is characterized by a bivariate membership function

$$\mu_R : U \times V \rightarrow [0, 1], \tag{2.4}$$

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

$$R = \{ \mu_R(u_1, v_1)/\langle u_1, v_1 \rangle, \mu_R(u_1, v_2)/\langle u_1, v_2 \rangle, \ldots, \mu_R(u_m, v_n)/\langle u_m, v_n \rangle \} \tag{2.5}$$

$$= \sum_{i,j} \mu_R(u_i, v_j)/\langle u_i, v_j \rangle, \tag{2.6}$$

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 $U \times V$.

More generally, an n-ary fuzzy relation R in $U_1 \times U_2 \times \cdots \times U_n$ is a fuzzy relation which is characterized by an n-variate membership function ranging over $U_1 \times U_2 \times \cdots \times U_n$.

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, and level-m fuzzy sets [11] and type-n fuzzy sets [12] 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,[1] the values of the 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.

---

[1] The properties of type-2 fuzzy sets are discussed in [13].

DEFINITION 3. An *L–fuzzy set* X, a *level-m fuzzy set* Y $(m=1,2,\ldots)$ and a *type-n fuzzy set* Z $(n=1,2,\ldots)$ in U are characterized by the following membership functions μ_X, μ_Y and μ_Z, respectively:[2]

$$\mu_X : U \to L, \tag{2.7}$$

$$\mu_Y : \underbrace{[0,1]\uparrow[0,1]\uparrow\cdots\uparrow[0,1]}_{m-1}{}^{U} \to [0,1], \tag{2.8}$$

$$\mu_Z : U \to \underbrace{[0,1]\uparrow[0,1]\uparrow\cdots\uparrow[0,1]}_{n}, \tag{2.9}$$

where L represents a lattice and $A^B \equiv A\uparrow B$ 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 of an ordinary fuzzy relation.

We shall now give some examples of various fuzzy sets.

EXAMPLE 1. Assume that

$$U=\{a,b,c,d\}. \tag{2.10}$$

Then, we may have a fuzzy set F in U as

$$F=\{0.1/a, 0.8/b, 0.9/d\} \tag{2.11}$$

and a fuzzy relation R in $U\times U$ as

$$R=\{0.3/\langle a,b\rangle, 0.9/\langle b,d\rangle\}. \tag{2.12}$$

---

[2] There is no associative law for exponentiation, that is,

$$U^{(V^W)} \neq (U^V)^W,$$

so μ_Y and μ_Z are defined more exactly as

$$\mu_Y : [0,1]\uparrow\big([0,1]\uparrow\cdots\uparrow\big([0,1]\uparrow([0,1]^U)\big)\cdots\big)\to[0,1],$$

$$\mu_Z : U \to \Big(\cdots\big(([0,1]^{[0,1]})^{[0,1]}\big)\cdots\Big)^{[0,1]},$$

using parentheses.

EXAMPLE 2. For U defined in (2.10), we have

$$X = \{\langle 0.1, 0.9\rangle / a, \langle 0.8, 1\rangle / b, \langle 0.9, 0\rangle / c\} \tag{2.13}$$

as an L-fuzzy set[3] in U.

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

$$\mathsf{Y1} = \{0.3/a, 0.2/b, 0.9/d\}, \tag{2.14}$$

$$\mathsf{Y2} = \{0.6/a, 0.1/b\}, \tag{2.15}$$

then we would have

$$Y = \{0.6/\mathsf{Y1}, 0.1/\mathsf{Y2}\} \tag{2.16}$$

as a level-2 fuzzy set in U.

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

$$Z = \{\textbf{high}/a, \textbf{middle}/b, \textbf{low}/d\} \tag{2.17}$$

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:

$$\textbf{high} = \{1/1, 0.8/0.9, 0.4/0.8\}, \tag{2.18}$$

$$\textbf{middle} = \{1/0.5, 0.5/0.6, 0.5/0.4\} \tag{2.19}$$

and

$$\textbf{low} = \{1/0, 0.8/0.1, 0.4/0.2\}. \tag{2.20}$$

---

[3] In this example, L is a lattice $[0,1]\times[0,1]$ ordered by

$$\langle a_1, b_1\rangle \leqslant \langle a_2, b_2\rangle \quad \Leftrightarrow \quad a_1 \leqslant a_2 \text{ and } b_1 \leqslant b_2,$$

where $a_1, a_2, b_1, b_2 \in [0,1]$. In this paper, L-fuzzy sets include only the case that L is

$$\underbrace{[0,1]\times[0,1]\times\cdots\times[0,1]}_{n},$$

with the ordered relation

$$\langle a_1, a_2, \ldots, a_n\rangle \leqslant \langle b_1, b_2, \ldots, b_n\rangle \quad \Leftrightarrow \quad a_i \leqslant b_i \quad \text{for all } i\ (i = 1, 2, \ldots, n),$$

where a_i and b_i $(i = 1, 2, \ldots, n)$ are elements of $[0,1]$.

3. A FUZZY-SET-THEORETIC DATA STRUCTURE

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

FSTDS is a data structure for representing fuzzy sets so that they can be manipulated conveniently and efficiently by fuzzy-set operators. FSTDS is composed of eight areas as follows (see Figs. 1–4):

(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 the fuzzy-set area (FSA). Fuzzy-set operators will be discussed in Sec. 5. In what follows, we shall discuss each area of FSTDS in detail.

(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[4] 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 a number n ($n \geqslant 0$) of grade/element pairs in one fuzzy set,[5] a grade part which contains n pointers of grades, and an element part which has n pointers of elements. For an ordinary fuzzy set consisting of n grade/element pairs, the element part

[4] By the term "data cell," we mean a basic element of each area regardless of the number of memory locations. Thus one data cell may occupy only one or several locations in a real storage structure.

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

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.

(3) *Grade area* (*GA*). This area is a collection of the numbers in the interval [0, 1] which are the values of membership functions.

(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 ($n \geqslant 1$, the length of the tuple) and n pointers which may refer to the grade area (GA) for an ordinary L–fuzzy set, to the fuzzy-set area (FSA) for an L–type-n fuzzy set[6] or again to the grade-tuple area (GTA) for a higher-order L–fuzzy set.[6]

(5) *Element area* (*EA*). The element area is a collection of element names, that is, all names of elements in the universes 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, the element names .123, +0.123 and +00.123000 are equal to each other, 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 ($n \geqslant 1$, 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 higher-order fuzzy relation.

(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

[6] A more generalized fuzzy set is defined in Definition 4.

the storage representation 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. However, this area is necessary, since it facilitates implementation of FSTDS system and gives added flexibility to it.

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

EXAMPLE 3. The fuzzy set F and the fuzzy relation R defined by (2.11) and (2.12), 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 (2.13), (2.16) and (2.17), respectively, in Example 2 are represented by FSTDS in Fig. 2, Fig. 3 and Fig. 4, respectively.

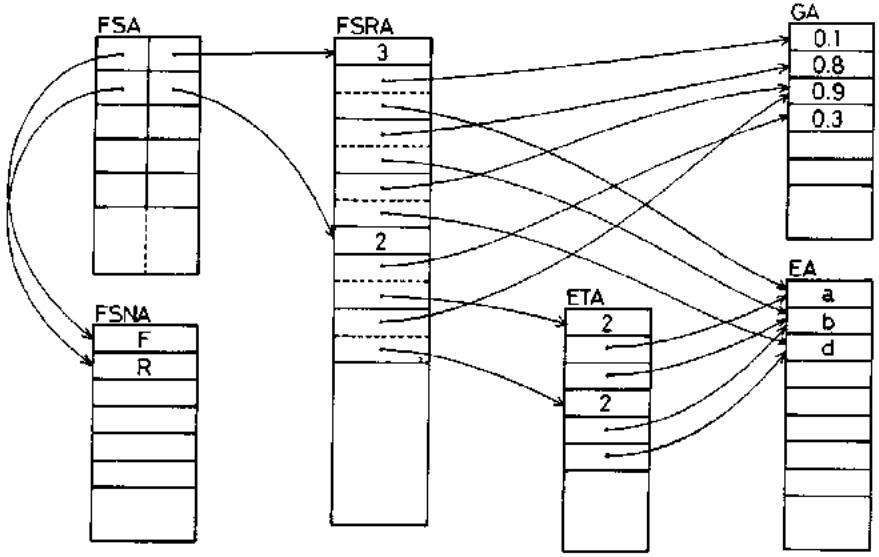

Fig. 1. The representation of a fuzzy set F and a fuzzy relation R by FSTDS.

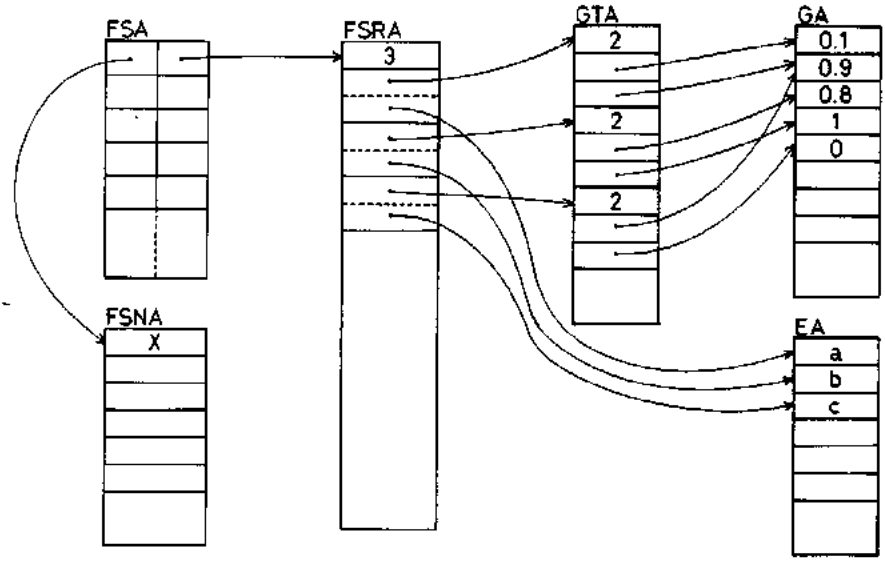

Fig. 2. The representation of an L–fuzzy set X by FSTDS.

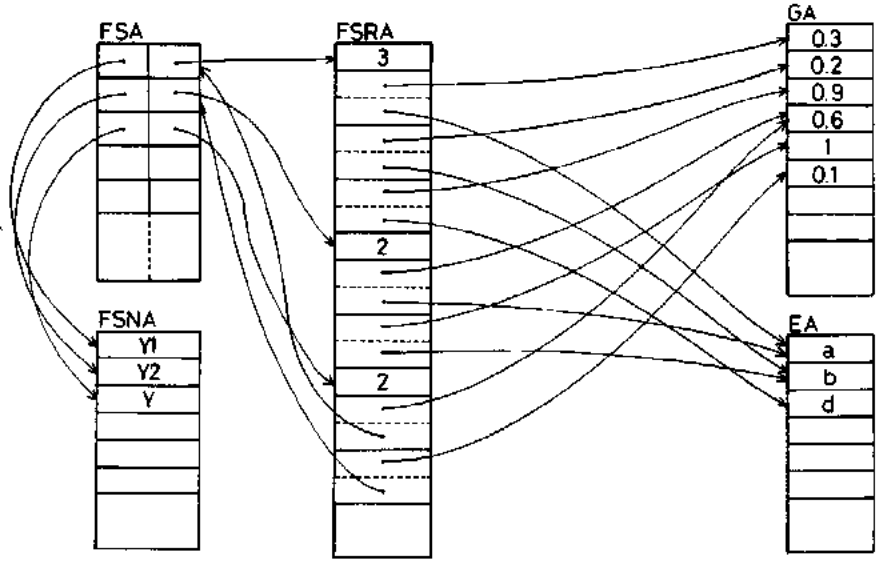

Fig. 3. The representation of a level-2 fuzzy set Y by FSTDS.

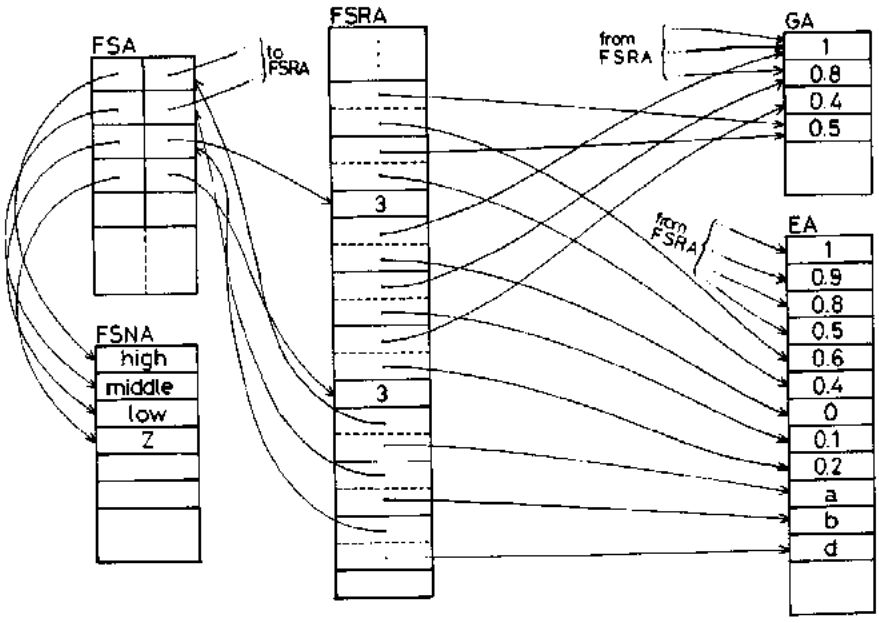

Fig. 4. The representation of a type-2 fuzzy set Z by FSTDS.

As illustrated in Figs. 1–4, the same grade values in the grade area (GA) and grade-tuple area (GTA) and the same element values in the element area (EA) and element-tuple area (ETA) are shared in FSTDS, respectively. But the common subset is not shared as in STDS. 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." In a generalized fuzzy set, various kinds of fuzzy sets defined in the previous section are combined into, e.g., an L-type-3 fuzzy set, a level-5 type-2 fuzzy relation, or an L-level-7 type-5 fuzzy relation.

DEFINITION 4. A *generalized fuzzy set* W in a universe of discourse U is defined recursively as follows:

(1) An ordinary set U is a generalized fuzzy set in U.
(2) The interval $[0,1]$ is a generalized fuzzy set in $[0,1]$.
(3) An ordinary fuzzy set in U is a generalized fuzzy set in U.
(4) If G and $G_1, G_2, \ldots, G_n$ are generalized fuzzy sets in $[0,1]$, then a fuzzy set

A which is characterized by a membership function

$$\mu_A : G \rightarrow G_1 \times G_2 \times \cdots \times G_n \tag{3.1}$$

is a generalized fuzzy set in [0, 1].

(5) If G, G_1 and G_2 are generalized fuzzy sets in [0, 1], then a fuzzy set A_1 characterized by

$$\mu_{A_1} : G_1^G \rightarrow G_2 \tag{3.2}$$

and a fuzzy set A_2 characterized by

$$\mu_{A_2} : G \rightarrow G_1^{G_2} \tag{3.3}$$

are generalized fuzzy sets in [0, 1].

(6) If E is a generalized fuzzy set in U and $G_1, G_2, \ldots, G_n$ are generalized fuzzy sets in [0, 1], then a fuzzy set A characterized by a membership function as

$$\mu_A : E \rightarrow G_1 \times G_2 \times \cdots \times G_n \tag{3.4}$$

is a generalized fuzzy set in U.

(7) If E is a generalized fuzzy set in U and G_1 and G_2 are generalized fuzzy sets in [0, 1], then a fuzzy set A_1 characterized by

$$\mu_{A_1} : G_1^E \rightarrow G_2 \tag{3.5}$$

and a fuzzy set A_2 characterized by

$$\mu_{A_2} : E \rightarrow G_1^{G_2} \tag{3.6}$$

are generalized fuzzy sets in U.

(8) All generalized fuzzy sets in U or [0, 1] are generated by applying the above rules.

It should be noted that an L–fuzzy set, a level-m fuzzy set and a type-n fuzzy set as defined in Definition 3 can be viewed as special cases of generalized fuzzy sets. For example, an L–fuzzy set in U is generated by applying rule (6) on the condition that $E = U$ and $G_1 = G_2 = \cdots = G_n = [0, 1]$. A level-2 fuzzy set in U and a type-2 fuzzy set in U are generated by applying rule (7) on the condition that $E = U$ and $G_1 = G_2 = [0, 1]$.

It should also be noted that more generalized fuzzy sets may be defined by replacing the interval [0, 1] by an arbitrary membership space such as a lattice or semiring. Furthermore, a generalized fuzzy relation can be defined using E in $U_1 \times U_2 \times \cdots \times U_n$ rather than in U.

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 (2.14), (2.15), (2.18), (2.19) and (2.20), respectively, then we may have

$$W = \{\langle \textbf{high, middle}, 1\rangle/\langle \text{Y1}, a\rangle, \langle \textbf{middle, low}, 0\rangle/\langle \text{Y1}, b\rangle, \langle \textbf{low, high}, 0.6\rangle/\langle \text{Y2}, c\rangle\} \tag{3.7}$$

as a generalized fuzzy set in $U \times U$ or, more exactly, as an L–level-2 type-2 fuzzy relation[7] in $U \times U$.

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

$$V = \{0.1/a, \langle \textbf{low}, 1\rangle/\text{Y1}, \langle \textbf{high, low}, 1\rangle/\langle \textbf{middle, high}, 3.56\rangle\}, \tag{3.8}$$

where Y1, **high**, **middle** and **low** are expressed by, say, (2.14), (2.18), (2.19) and (2.20), respectively. Thus, it is not necessary 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 natural for the data cells of each area to occupy contiguous array components. As for the representation of a whole area, we have choices depending on 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.

[7] It is not so easy to give a name specifying exactly the structure of a generalized fuzzy set. In this paper we have no intention of defining more exactly how to specify it.

One integer array is initially divided into many blocks of the same size.[8] 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 high locality of memory references in searching one specific area, so we may attain 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 fuzzy 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.

4. FUZZY-SET-THEORETIC DATA STRUCTURE SYSTEM

In this section, we shall describe the FSTDS language (FSTDSL for short), which is a programming language which enables a user to make use of FSTDS, and the FSTDS system, which interprets and executes a program written in FSTDSL. The 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; for example,

$$\mathrm{opr}(\mathrm{opd}_1, \mathrm{opd}_2, \ldots, \mathrm{opd}_n); \tag{4.1}$$

where opr is a fuzzy-set operator name and opd_i, $i = 1, 2, \ldots, n$, are its operands. The number of operands is dependent on a fuzzy-set operator (see Table 1). Each opd_i may be either a fuzzy set or a grade/element pair. Since any depth of nested operations is feasible, the opd_i may be again of the form of (4.1) instead of fuzzy sets.

---

[8] We can specify the size of blocks in the head of the FSTDSL program. The default size is 100 now.

TABLE 1
Fuzzy-Set Operators Available in the FSTDS System

Fuzzy-set operators	#opd	Remarks
SET($u_1,u_2,\ldots,u_n$)[a]	$n \geq 0$	Construct ordinary set
FSET($\mu_1/u_1,\mu_2/u_2,\ldots,\mu_n/u_n$)[a]	$n \geq 0$	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(χ)	1	Operate on all fuzzy sets over the domain of the operand set χ
INTERSECTIONA(χ)	1	
PRODA(χ)	1	
ASUMA(χ)	1	
ADIFA(χ)	1	
BSUMA(χ)	1	
BDIFA(χ)	1	
COMPOSE($R_1,R_2,\ldots,R_n$)	$n \geq 2$	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
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($\mu/u,X$)	2	Is μ/u an element of X?
CUT($\mu_1/\mu_2,X$)	2	$\mu_1 \wedge$(μ_2-level set of X)
SOP($\mu/n,X$)[b]	2	Scalar operation of μ and X
EXP($\mu/x,X$)	2	$X^x \wedge \mu$
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	Number of elements of X
MAXG(X)	1	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 $X_1,X_2,\ldots,X_n$

TABLE I
Fuzzy-Set Operators Available in the FSTDS System

Fuzzy-set operators	#opd	Remarks
PRINTB($X_1, X_2, \ldots, X_n$)	$n \geq 1$	Print $X_1, X_2, \ldots, X_n$ in Boolean type
PRINTS($X_1, X_2, \ldots, X_n$)	$n \geq 1$	Print $X_1, X_2, \ldots, X_n$ in set type
PRINTN($X_1, X_2, \ldots, X_n$)	$n \geq 1$	Print $X_1, X_2, \ldots, X_n$ with names
PRINTC(character string)	1	Print character string
DUMP($\alpha_1, \alpha_2, \ldots, \alpha_n$)	$n \geq 1$	Dump areas in FSTDS
SNAP(α)	1	Print 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)[c]	1 or 0	Evaluate X and halt

Definitions of Subscripted Symbols

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 the PARA operator

[a] For SET and FSET operators, SET( ) and FSET( ), i.e., $n = 0$, mean the empty set.
[b] For SOP operator, n represents: 1, maximum; 2, minimum; 3, product; 4, algebraic sum; 5, absolute difference; 6, bounded sum; 7, bounded difference.
[c] For END operator, END can be used for END( ).

In addition to the expression, we may have a *statement* in which a finite number of occurrences of ⟨set name⟩ : = are followed by the expression (4.1) that is,

$$v_1 := v_2 := \cdots := v_m := \text{opr}(\text{opd}_1, \text{opd}_2, \ldots, \text{opd}_n); \tag{4.2}$$

where the opr and opd_i, $i = 1, 2, \ldots, n$, are the same as in (4.1); v_j, $j = 1, 2, \ldots, m$, are set names; and the symbol := means the assignment operation. The entire statement (4.2) means that the value of the expression is computed and assigned to all set names $v_1, v_2, \ldots, v_m$.

The FSTDS system also computes the grades of fuzzy sets for 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 the FSTDS system are just like built-in functions in other programming languages.

The FSTDS system has 52 fuzzy-set operations currently available. (See Table 1.) We shall describe fuzzy-set operators in detail in the next section. To complete this section, we present some simple examples to illustrate 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. The FSTDS system interprets 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 sets and type-n fuzzy sets in FSTDSL.

For the L-fuzzy set defined by (2.13) 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 cannot 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 (2.16) 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 defined by (2.17) 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 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 by (3.7) and the fuzzy set V by (3.8) 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 now we can define W and V:

```
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[9] G 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:=FSET(⟨V, A⟩);
```

[9] A fuzzy graph is discussed in [14] by Rosenfeld.

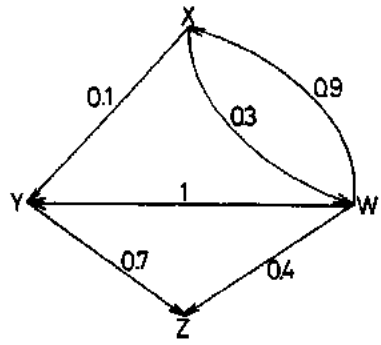

Fig. 5. A fuzzy directed graph G.

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.

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

$$(X \cup Y) \circ R = (X \circ R) \cup (Y \circ R), \tag{4.3}$$

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

Suppose that X and Y are defined by

$$X = \{1/a, 0.9/b, 0.3/c\} \tag{4.4}$$

and

$$Y = \{0.1/a, 0.7/b, 0.9/c\}, \tag{4.5}$$

and R is defined in terms of the relation matrix

$$R = \begin{matrix} \\ a \\ b \\ c \end{matrix} \begin{matrix} \begin{matrix} a & b & c \end{matrix} \\ \begin{bmatrix} 1 & 0.8 & 0 \\ 0.7 & 1 & 0.2 \\ 0 & 0.5 & 0.1 \end{bmatrix} \end{matrix}. \tag{4.6}$$

Then we can write the program as follows:

```
1  X:=FSET(1/A, 0.9/B, 0.3/C);
2  Y:=FSET(0.1/A, 0.7/B, 0.9/C);
```

```
3  R:=FSET(1/⟨A,A⟩, 0.8/⟨A,B⟩, 0.7/⟨B,A⟩, 1/⟨B,B⟩,
4            0.2/⟨B,C⟩, 0.5/⟨C,B⟩, 0.1/⟨C,C⟩);
5  PRINT(ASSIGN(Z, UNION(X, Y)));
6  PRINT(IMAGE(R, Z));
7  V:=IMAGE(R,X);  W:=IMAGE(R,Y);
8  PRINT(UNION(V,W));
9  END;
```

and the execution results of above program are

FSET(1/A, 0.9/B, 0.9/C); $X \cup Y$,
FSET(1/A, 0.9/B, 0.2/C); $(X \cup Y) \circ R$,
FSET(1/A, 0.9/B, 0.2/C); $(X \circ R) \cup (Y \circ R)$.

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);
```

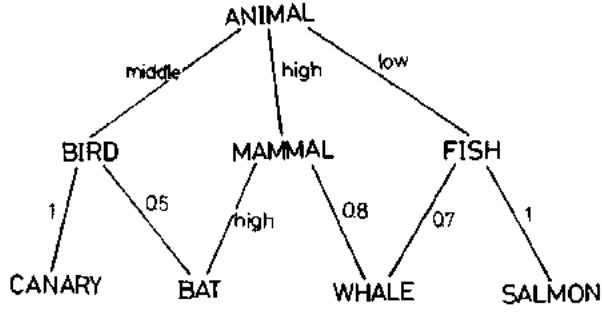

Fig. 6. An example of fuzzy knowledge.

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 for the 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 Examples 6–11, FSTDSL has a simple syntax. But it seems 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);                    (4.7)
```

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 an ordinary fuzzy set). This is evident because when the statement (4.7) is interpreted by the 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 convention is that it is inconvenient to put a character string representing an element name in quotation marks[10] (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 although 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 number of operands and the types[11] of operands, and the operands are interpreted in the predefined order.

---

[10]Quotation marks are used for another purpose, that is, the separate symbols (e.g., /, (,), =) are rendered inoperative by the use of the quotation marks.

[11]The meaning of "type" in this context is different from that in "type-n fuzzy set". Here we mean 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, a type-n fuzzy set etc.

For example, given the statement (4.7), X becomes a fuzzy-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 occurred 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, the program goes into an error state.

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 necessary to output spaces.

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.

The FSTDS system consists of a simple interpreter, a collection of fuzzy-set operations and a data structure (FSTDS).

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

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

EXAMPLE 12. The program in FSTDSL and FORTRAN given by

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

results in

```
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., in the first column on cards) to differentiate an FSTDSL statement from a FORTRAN statement, and put one or two exclamation marks (! or !!) before a FORTRAN integer or real variable, respectively, to indicate 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, compute the complement[12] of LARGE (i.e., NOT_LARGE) and output them. In more detail, we first make an ordinary set U and a fuzzy set LARGE empty[13] (line 3), and then compute a grade value G of the element J, and add the G/J pair to LARGE and the J to U for J=1,2,...,5 (from line 4 to 8). Next we output the set U and the fuzzy set LARGE together with fuzzy-set names (line 9), compute the absolute difference of LARGE from U (i.e., the complement of LARGE) and output it with the fuzzy-set name (line 10).

A program written in FSTDSL/FORTRAN is translated into the FORTRAN program by the 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 the 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 statements. In FSTDSL, for example, the expression SET(X) represents an ordinary set $\{X\}$, while SET(!!X) represents $\{3.14\}$ if X=3.14.

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

---

[12] The complement of a fuzzy set F in a universe of discourse U is defined by

$$\neg F = \sum_i 1 - \mu_F(u_i)/u_i,$$

where u_i are the elements of U.

[13] EMPTY is predefined in the FSTDS system as the empty fuzzy set. Other such predefined fuzzy sets are TRUE and FALSE, which represent FSET(1/1) and FSET(1/0), respectively.

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. Passing them from FSTDSL to FORTRAN is not so easy, because FORTRAN cannot deal with even ordinary sets. So we have set up 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 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 an FSTDS translator for those languages.

5. FUZZY-SET OPERATORS IN THE FSTDS SYSTEM

At present, the FSTDS system provides 52 fuzzy-set operators for use in problem solving. Table 1 lists all the fuzzy-set operators currently available. The definitions of these operations on fuzzy sets and fuzzy relations are briefly presented in the Appendix. For more detailed discussions, see [1, 2, 10–13, 15, 17–19].

The fuzzy-set operators in the FSTDS system may be classified into the following eight categories:

(1) Set-construction operators
(2) Assignment operators
(3) Operators on fuzzy sets of the same type
(4) Operators on fuzzy relations
(5) Relational operators
(6) Other operators on fuzzy sets
(7) Output operators
(8) Operators to debug and control the output format

We shall describe each category with examples in the following.

(1) *Set construction operators.* In this category there are two operators, SET and FSET, which construct an ordinary set and a fuzzy set, respectively. The operator SET takes some elements as operands, while FSET takes some grade/element pairs.

For ordinary sets, duplicate elements are united. This is also the case for fuzzy sets, the grade value being defined as the maximum of the value of the grades of these elements. Note that the grade/element pair whose grade value is zero is omitted from the constructed fuzzy set for the operator FSET. Note also that a fuzzy relation can be defined using the same operator FSET, as shown in Examples 6, 8, 9 and 10, and other various kinds of fuzzy sets can be defined using a sequence of the operators SET and FSET,[14] as shown in Examples 7, 8, 9 and 11.

EXAMPLE 13. As examples of operators SET and FSET, we have

```
1  SET(A, B, C);

2  SET(A, C, B, A);

3  FSET(0.1/A, 0.2/B, 0.8/A, 0/D);

4  FSET();
```

Note that statement 1 is equivalent to 2, that 3 is equivalent to

```
FSET(0.8/A, 0.2/B);
```

and that statement 4 represents the empty set, which is equivalent to SET() or the predefined empty set EMPTY.

(2) *Assignment operators.* The assignment operators are used for assigning a fuzzy set to a fuzzy-set name. We have := and ASSIGN as the assignment operators.

The operator := is used at the top level of nested operations and can be used for multiple assignment as in (4.2). Note that the operator := is the only infix operator at present. The operator ASSIGN is introduced in order to assign a fuzzy set at an arbitrary depth of nested operations. This operator assigns the second operand to the first one (e.g., ASSIGN(X, Y) is equivalent to X:=Y). The value of the operator ASSIGN is the assigned fuzzy set X of ASSIGN(X, Y).

These assignment operations do not change the pointer, but reproduce the fuzzy-set representation in the fuzzy-set representation area (FSRA) in FSTDS. So a fuzzy set can be updated or redefined independently.

EXAMPLE 14.

```
1  ASSIGN(X, FSET(0.1/A, 0.2/B));

2  Y1:=Y2:=ASSIGN(Y3, X);

3  PRINT(UNION(ASSIGN(Z, Y1), SET(C)));
```

[14] At present the nested operators of SET and FSET are not feasible.

(3) *Operators on fuzzy sets of the same type.* The operators in this category operate on more than one fuzzy set of the same type. There are seven operators, namely, UNION, INTERSECTION, PROD (product), ASUM (algebraic sum), ADIF (absolute difference), BSUM (bounded sum) and BDIF (bounded difference).

These operators have as value the fuzzy set obtained by a variety of operations on grade values for the same elements of operand fuzzy sets, that is, if the number of operands is two, we have as value the fuzzy set expressed as

$$\sum_i \mu_{A_1}(u_i) * \mu_{A_2}(u_i)/u_i, \tag{5.1}$$

where A_1 and A_2 are operand fuzzy sets and $*$ is a binary operation on two grade values for the above operators (e.g., the maximum for UNION, the minimum for INTERSECTION and so on). The grade value for the default element of an operand fuzzy set is considered as zero, that is, non-membership in it.

The number of operands is equal to or greater than two. But if it is greater than two, the first two operands are operated on, and then its result and the next operand are operated on, and similarly for the remaining operands. Note that different type fuzzy sets could be operated on by these operators, but the results would be empty.

EXAMPLE 15. If the fuzzy sets X1 and X2 are expressed as

```
X1:=FSET(0.1/A, 0.2/B, 0.3/C);

X2:=FSET(       0.8/B, 0.9/C, 0.5/D);
```

then the statements

```
PRINT( UNION(X1, X2) );

PRINT( INTERSECTION(X1, X2) );

PRINT( PROD(X1, X2) );

PRINT( ASUM(X1, X2) );

PRINT( ADIF(X1, X2) );

PRINT( BSUM(X1, X2) );

PRINT( BDIF(X1, X2) );
```

result in the output

FSET(0.1/A, 0.8/B, 0.9/C, 0.5/D);

FSET(0.2/B, 0.3/C);

FSET(0.16/B, 0.27/C);

FSET(0.1/A, 0.84/B, 0.93/C, 0.5/D);

FSET(0.1/A, 0.6/B, 0.6/C, 0.5/D);

FSET(0.1/A, 1/B, 1/C, 0.5/D);

FSET(0.1/A);

In addition to the above operators, the FSTDS system provides operators whose names have a character A at the end of the above operator names (e.g., UNIONA, INTERSECTIONA and so on). These operators have one operand of the set of fuzzy sets and operate on fuzzy sets over the domain of the operand set. For example, assume that

$$\mathcal{A} = \{A_1, A_2, \ldots, A_n\}, \tag{5.2}$$

where A_i, $i = 1, 2, \ldots, n$, are fuzzy sets; then UNIONA($\mathcal{A}$) means

$$\bigcup_{A_i \in \mathcal{A}} A_i \tag{5.3}$$

or

$$A_1 \cup A_2 \cup \cdots \cup A_n. \tag{5.4}$$

These operators are introduced to deal conveniently with the set of fuzzy sets as the operand, but a fuzzy set of fuzzy sets (i.e., a level 2 fuzzy set) may occur as an operand of these operators. In such a case, if $\mathcal{A}$ is expressed as

$$\mathcal{A} = \{\mu_1/A_1, \mu_2/A_2, \ldots, \mu_n/A_n\} \tag{5.5}$$

where μ_i and A_i are grade values and fuzzy sets, respectively, then, for example, UNIONA($\mathcal{A}$) is defined as

$$\bigcup_{A_i \in \mathcal{A}} \mu_i \wedge A_i$$

where $\mu_i \wedge A_i$ is a scalar operation as defined by (A.22) in the Appendix.

EXAMPLE 16. Let the fuzzy sets X1, X2 and X3, a set XA of these fuzzy sets, and a level-2 fuzzy set XB be given as follows:

```
X1:=FSET(0.1/A, 0.2/B, 0.3/C);

X2:=FSET(       0.3/B, 0.9/C, 0.5/D);

X3:=FSET(0.7/A,        1.0/C, 0.9/D);

XA:=SET(X1, X2, X3);

XB:=FSET(0.1/X1, 0.6/X2, 1/X3);
```

Then the execution results of

```
PRINT( UNIONA(XA), UNIONA(XB) );

PRINT( INTERSECTIONA(XA), INTERSECTIONA(XB) );

PRINT( PRODA(XA), PRODA(XB) );

PRINT( ASUMA(XA), ASUMA(XB) );

PRINT( ADIFA(XA), ADIFA(XB) );

PRINT( BSUMA(XA), BSUMA(XB) );

PRINT( BDIFA(XA), BDIFA(XB) );
```

are

```
FSET(0.7/A, 0.3/B, 1/C, 0.9/D);

FSET(0.7/A, 0.3/B, 1/C, 0.9/D);

FSET(0.3/C);

FSET(0.1/C);

FSET(0.27/C);

FSET(0.06/C);

FSET(0.73/A, 0.44/B, 1/C, 0.95/D);
```

```
FSET(0.73/A, 0.37/B, 1/C, 0.95/D);

FSET(0.6/A, 0.1/B, 0.4/C, 0.4/D);

FSET(0.6/A, 0.2/B, 0.5/C, 0.4/D);

FSET(0.8/A, 0.5/B, 1/C, 1/D);

FSET(0.8/A, 0.4/B, 1/C, 1/D);

EMPTY;

EMPTY;
```

(4) *Operators on fuzzy relations.* A fuzzy relation can be defined as a fuzzy set of n-tuples, using a set construction operator FSET. The union, intersection etc. for fuzzy relations can be obtained by the operators on fuzzy sets of the same type (UNION, INTERSECTION etc.), respectively. But the operators in this category involve components of an n-tuple element in fuzzy relations.

We have in this category nine fuzzy-set operators, namely, COMPOSE (composition of fuzzy relations), CONVERSE (converse relation), IMAGE (image of a fuzzy set under a fuzzy relation), CIMAGE (converse image of a fuzzy set under a fuzzy relation), DOMAIN (domain of a fuzzy relation), RANGE (range of a fuzzy relation), CP (Cartesian product of fuzzy sets), RS (restriction of a fuzzy relation to a fuzzy set) and RELATION (fuzzy relation deduced from a level-m fuzzy set).

EXAMPLE 17. If we have

```
R1:=FSET(1/⟨A,2⟩, 0.6/⟨A,3⟩, 0.3/⟨B,2⟩, 0.1/⟨C,1⟩, 0.8/⟨C,3⟩);

R2:=FSET(0.1/⟨1,G⟩, 0.8/⟨1,I⟩, 0.2/⟨2,G⟩, 1/⟨3,H⟩);

XA:=FSET(0.1/A, 0.9/B, 0.5/C);

XB:=FSET(0.8/1, 0.2/2, 1.0/3);
```

then the execution results of

```
PRINT( COMPOSE(R1,R2), COMPOSE(R2,R1) );

PRINT( CONVERSE(R1), CONVERSE(R2) );

PRINT( IMAGE(R1,XA), CIMAGE(R1,XB) );

PRINT( DOMAIN(R1), RANGE(R1) );

PRINT( CP(XA,XB), RS(R1,XA) );
```

are

FSET(0.2/⟨A,G⟩, 0.6/⟨A,H⟩, 0.2/⟨B,G⟩, 0.1/⟨C,G⟩, 0.8/⟨C,H⟩, 0.1/⟨C,I⟩);

EMPTY;

FSET(1/⟨2,A⟩, 0.6/⟨3,A⟩, 0.3/⟨2,B⟩, 0.1/⟨1,C⟩, 0.8/⟨3,C⟩);

FSET(0.1/⟨G,1⟩, 0.8/⟨I,1⟩, 0.2/⟨G,2⟩, 1/⟨H,3⟩);

FSET(0.1/1, 0.3/2, 0.5/3);

FSET(0.6/A, 0.2/B, 0.8/C);

FSET(1/A, 0.3/B, 0.8/C);

FSET(0.1/1, 1/2, 0.8/3);

FSET(0.1/⟨A,2⟩, 0.1/⟨A,3⟩, 0.2/⟨B,2⟩, 0.5/⟨C,1⟩, 0.5/⟨C,3⟩, 0.1/⟨A,1⟩,

0.8/⟨B,1⟩, 0.9/⟨B,3⟩, 0.2/⟨C,2⟩);

FSET(0.1/⟨A,2⟩, 0.1/⟨A,3⟩, 0.3/⟨B,2⟩, 0.1/⟨C,1⟩, 0.5/⟨C,3⟩);

(5) *Relational operators.* Operators in this category correspond to operators whose values are **True** or **False** in other programming languages (e.g., .EQ., .GT. and so on in FORTRAN). As the value of an expression in FSTDSL must always be a fuzzy set, we let these operators take as value a fuzzy set in the truth-value set [0, 1]. That is, **True** and **False** correspond to FSET(1/1) and FSET(1/0), respectively. Recall that FSET(1/1) and FSET(1/0) are equivalent to SET(1) and SET(0), respectively. A user can, however, obtain readable output by using the output operator PRINTB which outputs character strings TRUE and FALSE for FSET(1/1) and FSET(1/0), respectively.

In this category, we also have the operators EQ (equality comparison), SUBSET (fuzzy-set inclusion), DISJOINT (disjointness) and ELEMENT (fuzzy-set membership).

It should be noted that the operator ELEMENT can take a value (e.g., FSET(1/0.7)) other than FSET(1/1) or FSET(1/0). For example, the value of the expression ELEMENT(0.6/A,X) is FSET(1/μ) if the grade of membership of the element A in the fuzzy set X is μ and μ is equal to or greater than 0.6; the value is FSET(1/0) if μ is less than 0.6. This will be shown in the next example.

EXAMPLE 18. If we have fuzzy sets XA, XB and XC defined by

XA:=FSET(0.1/A, 0.2/B);

XB:=FSET(0.2/B, 0.1/A, 0/C);

XC:=FSET(0.5/A, 0.9/B);

then the executed results of the statements

```
PRINT( EQ(XA, XB), EQ(XA, XC) );

PRINT( SUBSET(XA, XB), SUBSET(XC, XA), SUBSET(EMPTY, XA) );

PRINT( DISJOINT(XA, XC), DISJOINT(XB, EMPTY) );

PRINT( ELEMENT(0.8/A, XC), ELEMENT(0.3/B, XC) );
```

are

```
FSET(1/1);

FSET(1/0);

FSET(1/1);

FSET(1/0);

FSET(1/1);

FSET(1/0);

FSET(1/1);

FSET(1/0);

FSET(1/0.9);
```

But if the operator PRINT is replaced by PRINTB in the above statements, the results will be as follows:

```
TRUE;

FALSE;

TRUE;

FALSE;

TRUE;

FALSE;

TRUE;

FALSE;

FSET(1/0.9);
```

(6) *Other operators on fuzzy sets.* The operators in this category do not operate on fuzzy sets of the same type, but on only one fuzzy set or two fuzzy sets of different types.

In this category there are the operators CUT, whose value is an α-level set of a fuzzy set; SOP (scalar operations), EXP (exponentiation), DIL (dilation), CON (concentration), CINT (contrast intensification) and NORM (normalization), which operate on grade values of one fuzzy set; CD (cardinality), # (the number of elements) and MAXG (the maximum grade value) whose values are fuzzy sets in real numbers, integers and the interval [0, 1], respectively; SF (support fuzzification) and GF (grade fuzzification), which fuzzify a fuzzy set by a kernel set; and DLT (delete fuzzy sets), which deletes fuzzy sets from the FSTDS system.

EXAMPLE 19. Given a fuzzy set X as

```
X:=FSET(1/A, 0.8/B, 0.6/C, 0.4/D);
```

we have FSTDSL statements as follows:

```
PRINT( CUT(1/0.5, X), CUT(0.4/0.7, X) );

PRINT( SOP(0.6/1, X), SOP(0.5/3, X), SOP(1/5, X) );

PRINT( EXP(1/2, X), EXP(0.8/3.6, X) );

PRINT( DIL(X), CON(X), CINT(X) );

PRINT( NORM(X) );

PRINT( CD(X), #(X), MAXG(X) );
```

and their outputs:

```
FSET(1/A, 1/B, 1/C);

FSET(0.4/A, 0.4/B);

FSET(1/A, 0.8/B, 0.6/C, 0.6/D);

FSET(0.5/A, 0.4/B, 0.3/C, 0.2/D);

FSET(0.2/B, 0.4/C, 0.6/D);

FSET(1/A, 0.6399/B, 0.36/C, 0.16/D);

FSET(0.8/A, 0.4478/B, 0.1589/C, 0.0369/D);
```

```
FSET(1/A, 0.8944/B, 0.7745/C, 0.6324/D);

FSET(1/A, 0.64/B, 0.36/C, 0.16/D);

FSET(1/A, 0.92/B, 0.68/C, 0.32/D);

FSET(1/A, 0.8/B, 0.6/C, 0.4/D);

FSET(1/2.8);

FSET(1/4);

FSET(1/1);
```

EXAMPLE 20. Before we apply the fuzzification operator, we must define a special set, called a kernel set, which plays the role of a basis set of fuzzification.

In the case of support fuzzification, a kernel set is defined as a relation between the elements of a universe of discourse and kernels. Let the kernels be as follows:

```
K1977:=FSET(1/1977, 0.9/1976, 0.8/1975);

K1976:=FSET(1/1976, 0.8/1975, 0.6/1974);

K1975:=FSET(1/1975, 0.7/1974, 0.4/1973);
```

Then a kernel set K using the above kernels is defined as

```
K:=SET(⟨1977,K1977⟩,⟨1976,K1976⟩,⟨1975,K1975⟩);
```

If a fuzzy set RECENT is given as

```
RECENT:=FSET(1/1977, 0.9/1976, 0.6/1975);
```

then we have the following FSTDSL statement to fuzzify the fuzzy set RECENT by the kernel set K:

```
MORE_OR_LESS_RECENT:=SF(RECENT,K);
```

and its output is

```
FSET(1/1977, 0.9/1976, 0.8/1975, 0.54/1974, 0.24/1973);
```

In the case of grade fuzzification, the kernel set is defined as a relation of grades and kernels.

Suppose that kernels G1, G2 and G3 and a kernel set G are as follows:

```
G1:=FSET(0.8/0.3, 1/0.4, 0.7/0.5);

G2:=FSET(0.8/0.6, 1/0.7, 0.7/0.8);

G3:=FSET(0.8/0.9, 1/1);

G:=SET(⟨0.4,G1⟩,⟨0.7,G2⟩,⟨1,G3⟩);
```

and a fuzzy set to be fuzzified is

```
X:=FSET(0.7/A, 0.8/B, 0.4/C, 0.7/D, 1/E);
```

Then the executed result of the FSTDSL statement

```
PRINT( GF(X,G) );
```

is the following type-2 fuzzy set:

```
FSET(G2/A, 0.8/B, G1/C, G2/D, G3/E);
```

(7) *Output operators.* The operators in this category output ordinary sets, ordinary relations, ordinary fuzzy sets, ordinary fuzzy relations, *L*–fuzzy sets, level-*m* fuzzy sets, type-*n* fuzzy sets and more generalized fuzzy sets in various formats, and output character strings. Each output operator except PRINTC can output an arbitrary type of fuzzy set. A user need not pay attention to the type of fuzzy-set output.

A standard output operator is PRINT, and the other operators are modified a little for convenience in programming.

PRINTB (print Boolean) outputs character strings TRUE and FALSE for FSET(1/1) and FSET(1/0), respectively, and outputs fuzzy sets in a similar way to PRINT for other fuzzy sets (see Example 18).

PRINTS (print set) outputs both ordinary sets and fuzzy sets in the form of ordinary sets, that is, it outputs only elements whose grade values are 1, but does not output its grade values. Note that the fuzzy set whose output is SET(); by PRINTS is not an empty fuzzy set, but the empty fuzzy set is output as EMPTY even by PRINTS.

PRINTN (print with names) outputs in the same form as PRINT except that the names of operand fuzzy sets are also output. If the operand is the expression, *** is output as its name.

PRINTC (print characters) never outputs any fuzzy sets, but only the character string of the operand.

The value of these operators is the last operand fuzzy set, but that of PRINTC is the empty set.

EXAMPLE 21. If we have FSTDSL statements as follows:

```
PRINTC(EXAMPLES#OF#OUTPUT#OPERATORS/);

PRINT(TRUE); PRINTB(TRUE);

X:=PRINT( SET(A,B,C) );

PRINTS( SET(A,B,C) );

Y:=PRINTS( FSET(0.1/A, 0.2/C, 0.3/E) );

PRINTS( FSET() );

Z:=PRINTN( Y, UNION(X,Y) );

PRINTC(UNION'(X,Y)'=); PRINT(Z);
```

then we have the execution results as follows:

```
EXAMPLES OF OUTPUT OPERATORS

FSET(1/1);

TRUE;

FSET(1/A, 1/B, 1/C);

SET(A, B, C);

SET();

EMPTY;

Y=FSET(0.1/A, 0.2/C, 0.3/E);

***=FSET(1/A, 1/B, 1/C, 0.3/E);

UNION(X,Y)=FSET(1/A, 1/B, 1/C, 0.3/E);
```

Note that the symbol / in an operand of PRINTC denotes end of line.

Various kinds of parameters for the output format are specified by PARA operator in the next category.

(8) *Operators to debug and control the output format.* The FSTDS system provides the operators DUMP, SNAP and PARA for the debugging of FSTDSL programs.

The DUMP operator dumps some areas in FSTDS in accordance with its operands only when it is executed. The operands of DUMP are several alphabetical characters which specify the dumped areas.

The operator SNAP outputs all fuzzy sets that have been defined by the time it is executed. The output form is just like PRINTN.

The PARA operator gives various kinds of information to the FSTDS system, some of which is very useful in debugging. For example, we can specify options to dump some areas in FSTDS and output the value of an operator each time a fuzzy-set operation is completed, and to output the statement to be interpreted, as a trace facility.

These options in the PARA operator can be considered as dynamic debugging tools, as opposed to the DUMP and SNAP operators.

The PARA operator is also used for controlling the output format. The controllable output format is as follows: the first and last columns for lines; the length of integer part and decimal fraction part of real number in both grade and element parts; the length of element name and fuzzy-set name in element part; and the length of fuzzy set name in grade part. The other options select the input and output devices, rename fuzzy-set operators, output the execution time of FSTDSL statements in one line, and output the information as shown in Fig. 7 on the execution of the END operator.

We have now described the fuzzy set operators available in the FSTDS system. These operations are implemented by SUBROUTINEs in FORTRAN.

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

EXAMPLE 22. 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:

$$\textbf{very } x = \mathrm{CON}(x), \tag{5.7}$$

$$\textbf{more or less } x = \mathrm{DIL}(x), \tag{5.8}$$

$$\textbf{slightly } x = \mathrm{NORM}(x \cap \neg \mathrm{CON}(x)), \tag{5.9}$$

$$\textbf{sort of } x = \mathrm{NORM}(\neg \mathrm{CON}(\mathrm{CON}(x)) \cap \mathrm{DIL}(x)), \tag{5.10}$$

$$\textbf{pretty } x = \mathrm{NORM}(\mathrm{CINT}(x) \cap \neg(\mathrm{CINT}(\mathrm{CON}(x))), \tag{5.11}$$

```
****** NORMAL END ..... FSTDS SYSTEM ******


GARBAGE COLLECTION 1 .....  0 TIMES
GARBAGE COLLECTION 2 .....  0 TIMES
GARBAGE COLLECTION 3 .....  0 TIMES

THE NUMBER OF EXECUTED STATEMENTS .........  111
THE NUMBER OF EXECUTED SET OPERATIONS .....  306

   NO SET OPERATION          COUNT       MSEC   MSEC/COUNT
    1 SET ..................    10         40         4.00
    2 FSET .................    70        973        13.90
    3 DUMP .................     0          0
    4 SNAP .................     0          0
    5 PARA .................     1          5         5.00
    6 PRINTC ...............     7        501        71.57
    7 ELEMENT ..............     0          0
    8 SOP ..................     0          0
    9 CUT ..................     0          0
   10 EXP ..................     0          0
   11 ASSIGN ...............   101        128         1.27
   12 UNION ................    78         68         0.87
   13 UNIONA ...............     0          0
   14 INTERSECTION .........     4          9         2.25
   15 INTERSECTIONA ........     0          0
   16 PROD .................     0          0
   17 PRODA ................     0          0
   18 ASUM .................     0          0
   19 ASUMA ................     0          0
   20 ADIF .................     5         29         5.80
   21 ADIFA ................     0          0
   22 BSUM .................     2         12         6.00
   23 BSUMA ................     0          0
   24 BDIF .................     0          0
   25 BDIFA ................     0          0
   26 PRINT ................     3         61        20.33
   27 DOMAIN ...............     0          0
   28 RANGE ................     0          0
   29 CONVERSE .............     0          0
   30 COMPOSF ..............     0          0
   31 CP ...................     7        316        45.14
   32 RS ...................     0          0
   33 IMAGE ................     3         35        11.67
   34 CIMAGE ...............     0          0
   35 NORM .................     3          4         1.33
   36 CON ..................     5          9         1.80
   37 DIL ..................     2          5         2.50
   38 CINT .................     2          5         2.50
   39 EQ ...................     0          0
   40 SUBSET ...............     0          0
   41 DISJOINT .............     0          0
   42 CD ...................     0          0
   43 # ....................     0          0
   44 MAXG .................     0          0
   45 RELATION .............     0          0
   46 SF ...................     0          0
   47 GF ...................     0          0
   48 PRINTB ...............     0          0
   49 PRINTS ...............     1         11        11.00
   50 PRINTN ...............     2         94        47.00
   51 DLT ..................     0          0
```

Fig. 7. List of the count and the CPU time of fuzzy-set operators executed in Example 22.

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

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

$$\begin{aligned} P_1&: \quad x \text{ is } A. \\ P_2&: \quad x \text{ and } y \text{ are } R. \\ \hline P_3&: \quad y \text{ is } A \circ R. \end{aligned} \tag{5.12}$$

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

If P_2 is a conditional statement such as

$$P_2: \quad \text{If } x \text{ is } P \text{ then } y \text{ is } Q \text{ else } y \text{ is } S. \tag{5.13}$$

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 for conditional propositions* [18],

$$R = P \times Q \cup \neg P \times S, \tag{5.14}$$

where $\times$, $\cup$ and $\neg$ stand for the Cartesian product, the union and the complement, respectively, or the *arithmetic rule for conditional propositions* [18],

$$R = (\neg(P \times V) \oplus (U \times Q)) \cap ((P \times V) \oplus (U \times S)), \tag{5.15}$$

where $\times$ and $\neg$ are the same as in (5.14), and $\cap$, and $\oplus$ stand for the intersection and the bounded sum, respectively.

In Fig. 8 we give a program[15] in FSTDSL/FORTRAN and its printed results. First, the program 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, by approximate reasoning it infers the consequences from the following premises

$$\begin{aligned} P_1&: \quad \text{X is SMALL.} \\ P_2&: \quad \text{X and Y are } \textit{approximately equal} \text{ (AE).} \end{aligned} \tag{5.16}$$

and

$$\begin{aligned} P_1&\quad \text{X is SMALL.} \\ P_2&\quad \text{If X is } \textbf{more or less} \text{ SMALL then Y is LARGE} \\ &\quad \text{else Y is } \textbf{sort of} \text{ MIDDLE.} \end{aligned} \tag{5.17}$$

---

[15] The functions $S(u;a,b,c)$ and $PI(u;b,c)$ in Fig. 8(a) have been defined in [17] by Zadeh. The function $Z(u;a,b,c)$ is the reflection of $S(u;a,b,c)$ about the line $u=b$.

```
        FSTDS SYSTEM SOURCE LIST

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

(a)

Fig. 8. A program in FSTDSL/FORTRAN for the linguistic hedges and approximate reasoning: (a) program, (b) output.

```
      F     PRINTC(/THE#MAX|M|N#RULE:/
     *          ####P3:#Y#IS#); PRINT(YA)
      F     PRINTC(/THE#ARITHMETIC#RULE:/
     *          ####P3:#Y#IS#); PRINT(YB)
      F     END
            STOP
            END
      C
            FUNCTION S(U,A,B,C)
            IF (U.LE.A) S=0.0
            IF (A.LT.U .AND. U.LE.B) S=2.0*((U-A)/(C-A))**2
            IF (B.LT.U .AND. U.LT.C) S=1.0-2.0*((U-C)/(C-A))**2
            IF (C.LE.U) S=1.0
            RETURN
            END
      C
            FUNCTION PI(U,B,C)
            IF (U.LE.C) PI=S(U,C-B,C-B/2.0,C)
            IF (U.GT.C) PI=1.0-S(U,C,C+B/2.0,C+B)
            RETURN
            END
      C
            FUNCTION Z(U,A,B,C)
            A2=2.0*A
            Z=S(A2-U,A,A2-B,A2-C)
            RETURN
            END
```

(8a *continued*)

```
*** 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, 0.8819/2, 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)

to compare the maximin rule and the arithmetic rule for conditional propositions.

In this example only a program to compute hedge effects and to carry out approximate reasoning is given. However, we could write a program to read the propositions, say, in the form of (5.16) or (5.17), to analyze their syntax and compute the hedge effects[16] using (5.7)–(5.11), to infer the consequences by approximate reasoning using (5.12), (5.14) and (5.15), to output the consequences of the fuzzy sets, and moreover to output these consequences in linguistic form by linguistic approximation.

6. CONCLUSION

We have described the FSTDS system, in which we can write a program using the concepts of fuzzy sets and fuzzy relations. This system, in which 52 fuzzy-set operators are available, is implemented in FORTRAN, and is currently running on a FACOM 230-45S computer. The system requires 116KB, including an integer array of size 5000 for FSTDS. This is because the current version of FSTDS is an interpreter implementation, so all SUBROUTINEs of the fuzzy-set operations must always be linked. These SUBROUTINEs occupy more than half of the FSTDS system.

There is a way around this difficulty, namely, the implementation of a compiler version of FSTDS which would read FSTDSL statements and generate a sequence of fuzzy-set operations. 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 we must compute the addresses in 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 the pointers in FSTDS but search no areas, are rather quickly executed. This fact is confirmed by use of the system SUBROUTINE CLOCKM available in FORTRAN on FACOM 230-45S. Rewriting in assembly language those parts of FSTDS which access FSTDS frequently will very significantly improve efficiency.

---

[16] More generally, we can use the hedge **not** and the connectives **and** and **or**.

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

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

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

To solve a given problem, we can write a program in FSTDSL using the concepts of fuzzy sets and fuzzy relations. We can use FSTDSL 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 the FSTDS system, we are now implementing an approximate-reasoning system and a fuzzy-graph manipulation system.

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

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 may 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 $-$ denote ordinary multiplication, addition and subtraction, respectively.

(1) *Union.*

$$F_1 \cup F_2 = \sum_i \mu_{F_1}(u_i) \vee \mu_{F_2}(u_i)/u_i. \tag{A.1}$$

(2) *Intersection.*

$$F_1 \cap F_2 = \sum_i \mu_{F_1}(u_i) \wedge \mu_{F_2}(u_i)/u_i. \tag{A.2}$$

(3) *Product.*

$$F_1 \cdot F_2 = \sum_i \mu_{F_1}(u_i) \cdot \mu_{F_2}(u_i)/u_i. \tag{A.3}$$

(4) *Algebraic sum.*

$$F_1 \dot{+} F_2 = \sum_i \mu_{F_1}(u_i) + \mu_{F_2}(u_i) - \mu_{F_1}(u_i)\cdot\mu_{F_2}(u_i)/u_i. \tag{A.4}$$

(5) *Absolute difference.*

$$F_1 \dot{-} F_2 = \sum_i |\mu_{F_1}(u_i) - \mu_{F_2}(u_i)|/u_i, \tag{A.5}$$

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

(6) *Bounded sum.*

$$F_1 \oplus F_2 = \sum_i 1 \wedge (\mu_{F_1}(u_i) + \mu_{F_2}(u_i))/u_i. \tag{A.6}$$

(7) *Bounded difference.*

$$F_1 \ominus F_2 = \sum_i 0 \vee (\mu_{F_1}(u_i) - \mu_{F_2}(u_i))/u_i. \tag{A.7}$$

(8) *Composition.*

$$R_1 \circ R_2 = \sum_{i,k} \bigvee_j (\mu_{R_1}(u_i, v_j) \wedge \mu_{R_2}(v_j, w_k))/\langle u_i, w_k \rangle. \tag{A.8}$$

(9) *Converse relation.*

$$R^{-1} = \sum_{i,j} \mu_R(v_j, u_i)/\langle u_i, v_j \rangle \tag{A.9}$$

(10) *Image.*

$$R(F) = \sum_j \bigvee_i (\mu_F(u_i) \wedge \mu_R(u_i, v_j))/v_j. \tag{A.10}$$

(11) *Converse image.*

$$R^{-1}(F) = \sum_i \bigvee_j (\mu_R(u_i, v_j) \wedge \mu_F(v_j))/u_i. \tag{A.11}$$

(12) *Domain.*

$$\text{domain}(R) = \sum_i \bigvee_j \mu_R(u_i, v_j)/u_i. \tag{A.12}$$

(13) *Range.*

$$\text{range}(R)=\sum_{j}\bigvee_{i}\mu_R(u_i,v_j)/v_j. \tag{A.13}$$

(14) *Cartesian product.*

$$F_1\times F_2=\sum_{i,j}\mu_{F_1}(u_i)\wedge\mu_{F_2}(v_j)/\langle u_i,v_j\rangle. \tag{A.14}$$

For n fuzzy sets,

$$F_1\times F_2\times\cdots\times F_n=$$

$$\sum_{i_1,i_2,\ldots,i_n}\mu_{F_1}(u_{i_1})\wedge\mu_{F_2}(u_{i_2})\wedge\cdots\wedge\mu_{F_n}(u_{i_n})/\langle u_{i_1},u_{i_2},\ldots,u_{i_n}\rangle. \tag{A.15}$$

(15) *Restriction.*

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

(16) *Equality.* F_1 is equal to F_2 if and only if for all i

$$\mu_{F_1}(u_i)=\mu_{F_2}(u_i). \tag{A.17}$$

(17) *Inclusion.* F_1 is a subset of F_2 if and only if for all i

$$\mu_{F_1}(u_i)\leqslant\mu_{F_2}(u_i). \tag{A.18}$$

(18) *Disjointness.* $F_1,F_2,\ldots,F_n$ are disjoint from each other if and only if for all combinations of i and j but $i\neq j$,

$$F_i\cap F_j=\varnothing, \tag{A.19}$$

where $\varnothing$ is the empty set, and F is the empty set if and only if for all i

$$\mu_F(u_i)=0. \tag{A.20}$$

(19) *α-level set.*

$$F_\alpha=\{u|\alpha\leqslant\mu_F(u)\} \tag{A.21}$$

(20) *Scalar operations.*

$$\mu * F = \sum_i \mu * \mu_F(u_i)/u_i, \tag{A.22}$$

where $*$ denotes an arbitrary binary operation.

(21) *Exponentiation.*

$$F^x = \sum_i (\mu_F(u_i))^x / u_i, \tag{A.23}$$

where x is a real number.

(22) *Dilation.*

$$\mathrm{dil}(F) = F^{0.5} = \sum_i \sqrt{\mu_F(u_i)}\ / u_i. \tag{A.24}$$

(23) *Concentration.*

$$\mathrm{con}(F) = F^2 = \sum_i (\mu_F(u_i))^2 / u_i. \tag{A.25}$$

(24) *Contrast intensification.*

$$\mathrm{cint}(F) = \begin{cases} 2 \cdot F^2 & \text{if} \quad 0 \leqslant \mu_F(u) \leqslant 0.5, \\ U \dot{-} \left(2 \cdot (U \dot{-} F)^2\right) & \text{if} \quad 0.5 \leqslant \mu_F(u) \leqslant 1. \end{cases} \tag{A.26}$$

(25) *Normalization.* F is normal if and only if

$$\bigvee_i \mu_F(u_i) = 1; \tag{A.26}$$

and the normalization is defined by

$$\mathrm{norm}(F) = \sum_i \mu_F(u_i) \cdot \bar{\mu}^{-1} / u_i, \tag{A.27}$$

where $\bar{\mu}^{-1}$ denotes the reciprocal of the maximum grade value in F.

(26) *Cardinality.*

$$\mathrm{cd}(F) = \mu_F(u_1) + \mu_F(u_2) + \cdots + \mu_F(u_n). \tag{A.28}$$

(27) *Support fuzzification.*

$$\mathrm{SF}(F; K) = \bigcup_i \mu_F(u_i) \cdot K(u_i), \tag{A.29}$$

where K is a kernel set of kernels $K(u_1), K(u_2), \ldots, K(u_n)$.

(28) *Grade fuzzification.*

$$\mathrm{GF}(F;K) = \sum_i K(\mu_i)/u_i, \tag{A.30}$$

where K is a kernel set of kernels $K(\mu_1), K(\mu_2), \ldots, K(\mu_n)$.

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, New York, 1975.
3. D. L. Childs, Description of a set-theoretic data structure, *AFIPS 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 relation, *Information Processing* **68**, 420–430 (1968).
5. J. T. Schwartz, Optimization of very high level language—I. Value transmission and its corollaries, *Comput. Languages* **1**, 161–194 (1975).
6. J. T. Schwartz, Optimization of very high level language—II. Deducing relationship of inclusion and membership, *Comput. Languages* **1**, 197–218, (1975).
7. J. T. Schwartz, Automatic data structure choice in a language of very high level, *Comm. ACM* **18**, 722–728 (1975).
8. T. Katayama et al., Logical relation processing language LOREL-1, *J. Information Processing Society of Japan* **15**, 326–334 (1974) (in Japanese).
9. N. Wirth, The programming language Pascal, *Acta Informat.* **1**, 35–63 (1971).
10. J. A. Goguen, *L*-fuzzy sets, *J. Math. Anal. Appl.* **18**, 145–174 (1967).
11. L. A. Zadeh, Quantitative fuzzy semantics, *Information Sci.* **3**, 159–176 (1971).
12. L. A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning, *Information Sci.* **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* (L. A. Zadeh, K. Tanaka et al., Eds.), Academic, New York, 1976, pp. 77–95.
15. L. A. Zadeh, A fuzzy-set-theoretic interpretation of linguistic hedges, *J. Cybernet.* **2**, 4–34 (1972).
16. G. Lakoff, Hedges: A study in meaning criteria and the logic of fuzzy concepts, *J. Philos. Logic* **2**, 458–508 (1973).
17. L. A. Zadeh, Calculus of fuzzy restrictions, in *Fuzzy Sets and Their Applications to Cognitive and Decision Processes* (L. A. Zadeh, K. Tanaka et al., Eds.), Academic, New York, 1975, pp. 1–39.
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 be published.

Received June 1977