introduce the concept of lattice-valued regular expressions. This concept provides not only the necessary tool for the analysis and synthesis of fuzzy automata, but also forms a vehicle for a recursive generation of the family of fuzzy languages accepted by fuzzy automata from certain simple fuzzy languages. Since the families of fuzzy languages accepted by various models of fuzzy automata are, in general, nonenumerable, the concept of regular fuzzy expressions could provide the necessary insights into the study of the structure of such families. Since the nondeterministic fuzzy automata are not equivalent to deterministic fuzzy automata. Here, we demonstrate that the nondeterministic fuzzy automata can be represented by regular fuzzy expression in a more generalised frame-lattice-valued finite automata. We also make some observations as to the family of the fuzzy languages. It will be shown that

- The family of the fuzzy languages of DLAs is not closed under Kleene closure
- The family of the fuzzy languages accepted by LAs is not closed under complement.

This phenomenon also shows the special properties of fuzzy finite automata and points at the fact that the regular fuzzy expressions for LA are not trivial extensions of those encountered in "classical" cases. In fact, we have not found the corresponding regular fuzzy expressions as in classical case for DLA. As it is well-known, deterministic finite automata and nondeterministic finite automata have the same regular expressions, owing to the fact that deterministic finite automata and nondeterministic finite automata are equivalent. This fact does not hold for lattice-valued finite automata. Having this in mind, we give another more direct description for the regular expressions for DLA.

II. LATTICE-VALUED FINITE AUTOMATA AND THEIR LANGUAGES

**Definition:** Given a lattice L, we use A,V to represent the supremum operation and infimum operation on L, respectively, with 0, 1 being the least and the largest elements. Assume that there is a binary operation * (we call it multiplication) on L such that (L, *, e) is a monoid with identity e ∈ L. We call L a lattice-monoid if it satisfies the following two conditions

a. (i) ∀ a ∈ L, a * 0 = 0 * a = 0,
b. \( \forall a, b, c \in L, \ a \cdot (b \cdot c) = (a \cdot b) \cdot (a \cdot c) \), and \( (b \cdot c) \cdot a = (b \cdot a) \cdot (c \cdot a) \)

c. \( (i) \) a \((V, b t) = !/(a^* bt)\), and \((V, b t) a = V t(b t a)\). If the distributive law in (iii) hold only for countable set \( \{bt\} \), then \( L \) is called a countable lattice-monoid. For a lattice-monoid, since we only concerned with the multiplication and finite supremum operation \( V \), in what follows, a lattice-manoid will be denoted by \((L, 0, V)\).

d. If we talk about the subalgebra \( L_1 \) of a lattice-monoid \((L, *, V)\), it means that \( L_1 \) is a nonempty subset of \( L \) and \( L_1 \) is closed under the multiplication and finite supremum of \( L \). We are not concerned with the infimum operation in \( L_1 \).

We give some examples of lattice-monoids.

Example

a. Let \((L, A, V)\) be a distributive lattice, and let \( A = A \), then \( L \) is a lattice- monoid, and the identity of multiplication is \( I \).

b. Let \((L, 0, V)\) he a lattice-monoid, the identity is \( e \). We use \( L(n) \) to denote all \( n \times n \) matrices with values in \( L \). The multiplication, denoted as \( \cdot \), is defined as sup composition; and \( V \) is the pointwise-\( V \). That is, for two \( n \times n \) matrices, \( A = (a_{ij}) \) and \( B = (b_{ij}) \), with values in \( L \), let \( A \circ B = C = (c_{ij}) \). Let \( A \cdot V B = D = (d_{ij}), \) then \( d_{ij} = a_{ij} ; \ v b_{ij} \). Then \((L(n), 0, V)\) is also a lattice-monoid, the identity is the diagonal-matrix \( E = \text{diag}(e, e, \ldots, e) \) with \( e \) as the diagonal element. In general, the multiplication on \( L(n) \) is not commutative, even if the multiplication on \( L \) is commutative.

Lattice-valued finite automaton

Definition: A lattice-valued finite automaton (LA, for short) is a five tuple,

\[ M = (X, U, \delta, q_0, q_f) \]

Where \( X, U \) are finite nonempty sets, \( \delta \): \( X \times U \rightarrow F(X) \), and \( q_0, q_f \): \( X \rightarrow L \) are \( L \)-fuzzy sets of \( X \). The elements of \( X \) are called states, and the elements of \( U \) are called (input) symbols, respectively. \( \delta \) is called a fuzzy transition function. Let \( U' \) denote the set of all words of finite letter over \( U \) and let \( A \) denotes the empty word. Then \( U^* \) is the free monoid generated by \( U \) with the concatenation operation. For \( \delta \in U^* \), denote as \( \delta : X \times U^* \rightarrow F(X) \), in the following forms:

\[ Vy = X, if y = z, then \delta*(x, A, y) = e, otherwise \delta*(x, A, y) = 0; \]

\[ Vw \in U^*, u \in U, \delta*(x, wu, y) = Vz \in X[\delta*(xw, z) \bullet \delta(z, u, y)] \]

Lattice-valued Language

Definition: Suppose that \( M = (X, U, \delta, q_0, q_f) \) is an LA. Then the \( L \)-valued language \( F_m \in F(U^*) \) accepted by \( M \) or recognized by \( M \) is defined as follows,

\[ f_m(w) = V_x, y \in X[q_0(x) \bullet \delta*(x, w, y) \bullet q_f(y)] \]

We also denote \( f_m(w) = q_0 \bullet \delta_w \bullet q_f = V_{x, y \in X}[q_0(x) \bullet \delta*(x, w, y) \bullet q_f(y)] \)

The element \( f \in F(U^*) \) is called an \( L \)-language on \( U \), an \( L \)-language which is accepted by an LA is called an LA-language. For two LAs \( M_1 \) and \( M_2 \), we say that they are equivalent if they accept the same LA-language, that is, \( f M_1 = f M_2 \). As to classical automata theory, we have the notions of deterministic finite automata and nondeterministic finite automata. The notion of an LA is a generalization of the notion of nondeterministic automaton: instead of sets of initial and final states we have fuzzy sets of initial and final states; instead of a (bivalent) transition relation we have fuzzy transition relation.

The LA is nondeterministic in nature: there may be non- zero truth degrees that the automaton can go to more than one state (given a state and input symbol). In the following we present a deterministic counterpart of the notion of an LA.

Deterministic lattice-valued Finite Automaton

Definition: A deterministic lattice-valued finite automaton (DLA, for short) is a five tuple,

\[ M = (X, U, \delta, q_0, q_f) \]

Where \( X, U \) are finite nonempty sets, \( \delta \): \( X \times U \rightarrow F(X) \), and \( q_0, q_f \) are L-fuzzy sets of \( X \). Such that \( \delta : X \times U \rightarrow X, q_0, q_f : X \rightarrow L \)

The extension of \( \delta \) onto \( U^* \) is the same as that of classical case.

Note that our definition differs from the usual definition of a deterministic automaton only in that the initial and the final states form a fuzzy set. This is, however, makes it possible to accept words to certain truth degrees, and thus to recognize L-
language. Moreover, in the definition of a DLA, we can require that fuzzy initial state or stated in the following theorem.

Theorem: For an L-language f on U, the following three conditions are equivalent.

1. There exists a DLA M1 = (Y, U, q, yo, p1) with crisp initial state yo such that f = fM1.

2. There exists a DLA M2 = (Z, U, E, q, F) with crisp final states F such that f = fM2

Unlike the classical case, the following discussions show that in general LA and DLA are not equally powerful.

Theorem: If an L-language can he accepted by a DLA, and then it also can be accepted by an LA. The converse does not hold.

Theorem: For any LA, M, there is an equivalent DLA iff the lattice L satisfies the following conditions, for any finite subset L' of L, the subalgebra of (L, •, V) generated by L' is finite.

We further discuss some operations on the families of LA languages. Recall we call an L-language accepted by an LA the LA language. Then the family of LA languages on U, denoted LA, forms a subset of F(U*).

Operations on L-Languages

Definition: Let f, f1, f2 ∈ F(U*) be L-languages.

1. The join of f1 and f2, denoted f1 U f2, is defined as f1 U f2(w) = f1(w) V f2(w) for any w ∈ U*.

2. The meet of f1 and f2, denoted f1 · f2, is defined as f1 · f2(w) = f1(w) · f2(w) for any w ∈ U*.

3. The concatenation of f1 and f2, denoted f1f2, is defined as f1f2(w) = Vw1,w2=w[f1(w1) x f2(w2)] for any w ∈ U*.

4. The Kleene closure of f, denoted f*, is defined as f0 = I and f0 = f n-1f for n ≥ 1.

5. The reversal of f, denoted f⁻1, is defined as f⁻1(w) = f(w⁻1), Vw ∈ U*; where for w = u1u2 …..uk, w⁻1 = uk…….u1,u2

6. For a ∈ L, the scalar operations of f and a, denoted af and fa, is defined as (af)(w) = a • f (w) and (fa)(w) = f (w) • a for any w ∈ U*.

From the definition of Kleene closure of f, it is necessary to require that L is closed under countable supremum. For this reason, we assume that L is a countable lattice-monoid in the following discussion.

Theorem: The family of LA languages is closed under the operations of join, concatenation, the Kleene closure and the scalar operations.

Proof: Suppose that M1 = (X1, U, δ1, q1, qf1) and M2 = (X2, U, δ2, q12, qf2) are LAs, the L-languages they accept are fM1 and fM2, respectively. Since we may rename the states of an LA at will without changing the language accepted we may assume that the two sets X1 and X2, are disjoint.

(i) Join Operation

We first consider the join operation of LA-languages. Construct an LA, M = (X, U, δ, q1, qf) as, X = X1 U X2, q1 : X → L and q2 : X → L are defined respectively as,

q1(x) = q11, x ∊ X1, q1(x) = q12, x ∊ X2, q1(x) = q11, x ∊ X2

δ : X x U → F (X) is defined as follows,

δ(x, u, y) = δ1(x, u, y), x, y ∊ X1
δ(x, u, y) = δ2(x, u, y), x, y ∊ X2
δ(x, u, y) = 0, otherwise

(ii) Concatenation Operation.

Considering the concatenation operation. Construct an LA, M = (X, U, δ, q1, qf) as,

X= X1 U X2, q1 : X → L and q2 : X → L are defined respectively as,

q1(x) = q11, x ∊ X1, q1(x) = 0, x ∊ X1, q1(x) = 0, x ∊ X2, q1(x) = q11, x ∊ X2

δ : X x U → F (X) is defined as follows,

δ(x, u, y) = δ1(x, u, y), x, y ∊ X1
δ(x, u, y) = δ2(x, u, y), x, y ∊ X2
δ(x, u, y) = 0, otherwise

Where a = Vx1 ∊ X1[δ1(x, u, x1) • q12(x1) • q2(x1) V x2 ∊ X2

δ*(x, A, y) = e, x=y
δ*(x, A, y) = 0, x!=y
δ*(x, w, y) = δ*(x, w, y), x ∊ X1
δ*(x, w, y) = δ*(x, w, y), x ∊ X2
\[ \delta^*(x, y) = b \quad x \in X_1, y \in X_2 \]
\[ \delta^*(x, y) = 0 \quad x \in X_2, y \in X_1 \]

Where \( b = V_{w1=2} V_{x1 \in X1, y \in X2} [\delta_1(x, w1, x1) \bullet q_{ii(x1)} \bullet q_{i2(y)} \bullet \delta_2(x, w1, x1)] \)

(iii) Scalar Operation

Consider the scalar operation of a \( \in L \) and \( f = f^M \), where \( M = (X, U, \delta, q_0, q_f) \) is an LA. Let \( aM = (X, U, \delta, aq_0, q_f) \) and \( Ma = (X, U, \delta, q_0, q_f a) \) where \( aq_0 : X \to L \) is defined as \( aq_0(x) = a \circ q_0(x) \) and \( q_f a : X \to L \) is defined as \( q_f a(x) = q_f (x) \). Then \( aM \) and \( Ma \) are two LAs and it is readily verified that \( f aM = a f^M = a f \) and \( fMa = f^M a = f a \). Hence LA is closed under the scalar operations.

We also study the necessary and sufficient conditions for the LA being closed under intersection, generalized intersection and reversal operations. If we denote the set of DLA languages on \( U \), the L-languages accepted by DLAs, by DLA, then the family DLA is closed under union, concatenation, scalar, intersection, generalized intersection and reversal operations. The proofs are similar to those of LA. However, the following example illustrate that DLA is not closed under Kleene closure operation.

Example 2.3: Let \( L = ([0,1], \bullet, V) \), \( U = \{0,1\} \), and an language \( U^* \to L \) is defined as \( F(w) = \frac{1}{2}, w = 0^m, m > 0 \)
\( F(w) = \frac{1}{3}, w = 1^m, m > 0 \)
\( F(w) = 0, \) otherwise

then \( f^*(A) = 1 \), and for any \( m > 0 \), \( f^*(0^m) = 1/2, f^*(1^m) = 1/3 \).
\( f^*((01)^m) = f^*((10)^m) = 1/6^m \) and thus \( R (f^*) \) is infinite. \( f^* \) is not a DLA language. But \( f \) is a DLA language which can be accepted by a DLA

\[ M = (X, U, \delta, q_0, q_f) \]

where \( X = (x_0, x_1, x_2, x_3), q_0 = 0^1/2/3x_1+1/3x_3 \) and \( \delta (x_0, 0) = x_1, \delta (x_0, 2) = x_2, \delta (x_1, 0) = x_1, \delta (x_1, 1) = x_1, \delta (x_2, 0) = x_1, \delta (x_2, 1) = x_1, \delta (x_3, 0) = x_3, \delta (x_3, 1) = x_1 \) Then DLA language are not closed under Kleene closure. \( q : Z \times U \to F(Z) \) is defined as,

III. LATTICE-VALUED REGULAR EXPRESSIONS

Definition: Let \( U \) be a finite nonempty set. The family \( R \) of regular L-expressions over \( U \) is defined inductively as follows

a. \( 0 \in R \);

b. \( A \in R \);

c. \( u \in R \), for all \( U \in U \), (aa) \( \in R \) and (aa) \( \in R \), for all \( a \in L \) and \( a \in L \), for all \( a_1, a_2 \in R \), for all \( a_1, a_2 \in R \);

d. \( (u \circ a z) \in R \), for all \( 01, az \in R \)

e. \( (a^*) \in L \), for all \( a \in R \);

f. There are no other regular L-expressions other than those given in steps (i) to (vii).

In writing regular L-expressions we can omit many parentheses if we assume that * has higher precedence than concatenation, scalar or c, and that concatenation has higher precedence than scalar or +, and that scalar has higher precedence than +.

CONCLUSION

The lattice-valued regular expressions and their languages are introduced in this paper. Some regular operations on LA and DLA are studied in detail. We show that LA is not closed under complement, and DLA is not closed under Kleene closure, which form the essential differences between lattice-valued automata and classical automata. Even if this phenomenon, we still proved the equivalence between lattice-valued regular expressions and lattice-valued finite automata in the sense of recognizing fuzzy languages. Some regular expressions on DLA are also discussed. The next step is to study computing with words. It would be of interest to study hierarchies of automata.

REFERENCES

3. Y.M.Li, Finite automata with values in lattice-monoid and their languages,