Information Sciences 178 (2008) 4152–4165

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Computing behavior of finite fuzzy machines – Algorithm and its application to reduction and minimization

K. Peeva *, Zl. Zahariev

Sofia 1000, P.O. Box 384, Faculty of Applied Mathematics and Informatics, Technical University of Sofia, Bulgaria

ARTICLE INFO

Article history:
Received 11 September 2007
Received in revised form 30 June 2008
Accepted 3 July 2008

Keywords:
Finite fuzzy machine
Behavior
Equivalence
Reduction
Minimization

ABSTRACT

We define finite fuzzy machines and investigate their behavior. Algorithm and software are proposed for computing behavior, for establishing equivalence and redundancy of states and for solving reduction and minimization problems. Computational complexity of the algorithm is discussed. Testing examples are supplied. The results are valid for finite max–min, min–max and max–product fuzzy machines.

1. Introduction

Fuzzy machines are proposed by Santos [29,30] and first studied by him [29–32] and by Santos and Wee [33]. During the period 1968–2000 few references were published, see the monographs [7] by Klir and Yuan, [17] by Mordeson and Malik, [21] by Peeva and Kyosev.

Recently the interest in this subject has highly increased. Many authors contributed to algebraic approach and introduced variety of fuzzy machines on suitable algebraic structures (lattices [20,34] or lattice-ordered monoids [8–12,14,16], as well as formal power series [27], semirings [18], categorical approach [13], homomorphisms [2], etc.) with extension for fuzzy languages [2,8,15,16,26].

As soon as the concept of fuzzy sequential machine has been introduced, Santos set equivalence, reduction and minimization problems for finite max–min fuzzy machines [31] and for finite max–product fuzzy machines [32]. In order to investigate them two types of algebras, called max–min algebra [31] and max–product algebra [32] have been developed. The role played by these algebras in the theory of max–min and max–product fuzzy machines is the same as that played by linear algebra in the theory of stochastic machines. But linear algebra, max–min algebra and max–product algebra are completely unrelated [5,17].

Without doubt, the problems of equivalence, reduction and minimization for fuzzy machines are among the main and most interesting in this field. Published papers concern theory and some algorithms: for equivalence of states and equivalence of machines [10,18,31,34]; for reduction [2,9–11,18,31,32] and minimization [3,9,11,12,15,16,27,31]. None of them pretends to present software for obtaining behavior matrix, for establishing equivalence of states, for finding redundant states, for finding reduced and minimal forms of a fuzzy machine – still open problems for finite fuzzy machines. The main

* Corresponding author.
E-mail addresses: kgp@tu-sofia.bg (K. Peeva), zlatko@tu-sofia.bg (Zl. Zahariev).

0020-0255/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2008.07.009

obstacle is that fuzzy algebras like max–min algebra, min–max algebra and max–product algebra are different from the traditional linear algebra. Obviously developing software for solving problems for finite fuzzy machines requires first to develop theory and software for fuzzy relational calculus [21–24] and then to implement it in software for finite fuzzy machines.

In this paper based on the theory developed in [20,21], and on the software for fuzzy relational calculus from [21–24] we define finite fuzzy machine over fuzzy algebra, propose algorithms and develop software for computing behavior, for establishing equivalence of states, for reduction and minimization of finite fuzzy max–min, min–max and max–product machines. In Section 2 we introduce basic notions from fuzzy relational calculus that are necessary for exposition. Section 3 presents behavior of finite fuzzy machine as well as illustrative examples for its computing. In Section 4, we give algorithm how to extract and compute a finite behavior matrix (when possible) from the complete behavior matrix. Section 5 illustrates implementation of the software for solving equivalence, reduction and minimization problems. Section 6 briefly describes software for computing behavior matrix of finite fuzzy machine and its implementation for equivalence, reduction and minimization. Examples are included in Appendix.

2. Basic notions

2.1. Fuzzy algebra

Partial order relation on a partially ordered set (poset) P is denoted by the symbol $\leqslant$. By the *greatest element* of the poset P we mean an element $b \in P$ such that $x \leqslant b$ for all $x \in P$. The *least element* of P is defined dually.

Set $\mathbb{I}_* = \langle [0,1], \vee, \wedge, \odot, 0, 1\rangle$, where $[0,1]$ is the real unit interval, $\odot$ is the usual product between real numbers and for each $a, b \in [0,1]$ the operations $\vee$, $\wedge$ are respectively defined by

$$a \vee b = \max\{a, b\}, \quad a \wedge b = \min\{a, b\}.$$

The algebraic structure $\mathbb{I}_* = \langle [0,1], \vee, \wedge, \odot, 0, 1\rangle$ is called *fuzzy algebra*.

$\mathbb{I}_*$ is a complete lattice with universal bounds 0 and 1.

In $\mathbb{I}_*$ the operations α, ε and $\diamondsuit$ are very often used. For each $a, b \in [0,1]$ they are defined as follows:

$$a\alpha b = \begin{cases} 1 & \text{if } a \leqslant b, \\ b & a > b, \end{cases} \quad a\varepsilon b = \begin{cases} b & \text{if } a < b, \\ 0 & a \geqslant b, \end{cases} \quad a \diamondsuit b = \begin{cases} 1, & \text{if } a \leqslant b, \\ \frac{b}{a}, & \text{if } a > b. \end{cases}$$

Remark. $\mathbb{I}_*$ is an example of the so called BL-algebra introduced in [6]. Operation α is also called Gödel implication and is denoted as $\rightarrow_G$ in Gödel algebra $\langle [0,1], \vee, \wedge, \odot, \rightarrow_G, 0, 1\rangle$ [5,25], operation $\diamondsuit$ is denoted as $\rightarrow_p$ in Product (or Goguen) algebra $\langle [0,1], \vee, \wedge, \odot, \rightarrow_P, 0, 1\rangle$, see [25].

2.2. Matrix products

A matrix $A = (a_{ij})_{m\times n}$ with $a_{ij} \in [0,1]$ for each $i, j, 1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, $m, n \in \mathbb{N}$ ($\mathbb{N}$ is the set of natural numbers), is called *membership matrix* [7]. In what follows we write 'matrix' instead of 'membership matrix'.

Two matrices $A = (a_{ij})_{m\times p}$ and $B = (b_{ij})_{p\times n}$ are called *conformable*, if the number of columns in A coincides with the number of rows in B.

Several matrix products with conformable matrices may be defined on $\mathbb{I}_*$.

Definition 1. Let the matrices $A = (a_{ij})_{m\times p}$ and $B = (b_{ij})_{p\times n}$ be given.

The matrix $C = (c_{ij})_{m\times n}$, where

(i) $C = A \bullet B$, is called max–min product of A and B if

$$c_{ij} = \bigvee_{k=1}^{p} (a_{ik} \wedge b_{kj}) \quad \text{when } 1 \leqslant i \leqslant m, \ \ 1 \leqslant j \leqslant n.$$

(ii) $C = A \circ B$, is called min–max product of A and B if

$$c_{ij} = \bigwedge_{k=1}^{p} (a_{ik} \vee b_{kj}) \quad \text{when } 1 \leqslant i \leqslant m, \ \ 1 \leqslant j \leqslant n.$$

(iii) $C = A \,\alpha\, B$, is called min $-\alpha$ product of A and B if

$$c_{ij} = \bigwedge_{k=1}^{p} (a_{ik} \alpha b_{kj}) \quad \text{when } 1 \leqslant i \leqslant m, \ \ 1 \leqslant j \leqslant n.$$

(iv) $C = A \,\varepsilon\, B$, is called max $-\varepsilon$ product of A and B if

$$c_{ij} = \bigvee_{k=1}^{p} (a_{ik} \varepsilon b_{kj}) \quad \text{when } 1 \leqslant i \leqslant m, \ \ 1 \leqslant j \leqslant n.$$

(v) $C = A \odot B$, is called max $-\odot$ product of A and B if

$$c_{ij} = \bigvee_{k=1}^{p} (a_{ik} \cdot b_{kj}) \quad \text{when } 1 \leqslant i \leqslant m, \ \ 1 \leqslant j \leqslant n.$$

(vi) $C = A \diamondsuit B$, is called min $-\diamondsuit$ product of A and B if

$$c_{ij} = \bigwedge_{k=1}^{p} (a_{ik} \diamondsuit b_{kj}) \quad \text{when } 1 \leqslant i \leqslant m, \ \ 1 \leqslant j \leqslant n.$$

Stipulation. For simplicity and compactness of exposition we use the sign $*$ instead of $\bullet$, $\circ$ and $\odot$ and write $A * B$ when the results are valid for any of the products $A \bullet B$, $A \circ B$, $A \odot B$, introduced in Definition 1.

2.3. Direct and inverse problems

If the matrices $A = (a_{ij})_{m\times p}$ and $B = (b_{ij})_{p\times n}$ are given, computing their product according to Definition 1 is called *direct problem resolution.*

If $A = (a_{ij})_{m\times p}$ and $C = (c_{ij})_{m\times n}$ are given, computing the unknown matrix $B = (b_{ij})_{p\times n}$ such that $A * B = C$ is called *inverse problem resolution.*

Theorem 1. *Let $A = (a_{ij})_{m\times p}$ and $C = (c_{ij})_{m\times n}$ be given matrices and let $\mathbb{B}_*$ denote the set of all matrices such that $A * B = C$. Then*

(i) *[28] For $A \bullet B = C$ with max–min composition, $\mathbb{B}_\bullet \neq \emptyset$ iff $A^t \ \alpha \ C \in \mathbb{B}_\bullet$.*
(ii) *[28] For $A \circ B = C$ with min–max composition, $\mathbb{B}_\circ \neq \emptyset$ iff $A^t \ \varepsilon \ C \in \mathbb{B}_\circ$.*
(iii) *[4] For $A \odot B = C$ with max $-\odot$ composition, $\mathbb{B}_\odot \neq \emptyset$ iff $A^t \diamondsuit C \in \mathbb{B}_\odot$.*

We implement Theorem 1 and software from [21–24] in a new algorithm and software for computing behavior of finite fuzzy machine.

3. Finite fuzzy machines

In this section, we define finite fuzzy machine $\mathscr{A}$ over fuzzy algebra $\mathbb{I}_*$ and describe its complete behavior by suitable matrix $T_{\mathscr{A}}$. It is shown that computing behavior matrix for finite fuzzy machine with the software from [21,22] is tremendous, which motivated the authors to develop new software for the same purpose.

For a finite set C we denote by $|C|$ its cardinality.

Definition 2 [19]. A *finite fuzzy machine* (FFM) over the fuzzy algebra $\mathbb{I}_*$ is a quadruple

$$\mathscr{A} = (X, Q, Y, \mathscr{M}),$$

where

(i) X, Q, Y are nonempty finite sets of input letters, states and output letters, respectively.
(ii) $\mathscr{M}$ is the set of transition-output matrices of $\mathscr{A}$, that determines its stepwise behavior. Each matrix $M(x \mid y) = (m_{qq'}(x \mid y)) \in \mathscr{M}$ is a square matrix of order $|Q|$ and $x \in X, y \in Y, q, q\prime \in Q, m_{qq'}(x \mid y) \in [0, 1]$.

In Definition 2 (ii) we mark by $(x \mid y)$ the pair $(x, y) \in X \times Y$ to emphasize that $x \in X$ is the input letter when $y \in Y$ is the output response letter. The same stipulation is used in next exposition for input–output pair of words of the same length.

We regard $m_{qq'}(x \mid y)$ as the degree of membership for the FFM to enter state $q' \in Q$ and produce output $y \in Y$ if the present state is $q \in Q$ and the input is $x \in X$.

3.1. Extended input–output behavior of FFM

While $\mathscr{M}$ is the set of transition-output matrices that describes operating of $\mathscr{A}$ for exactly one step, in this subsection we will be interested in matrices that describe operating of $\mathscr{A}$ for words, i.e. for more than one consecutive steps.

The *free monoid of the words* over the finite set X is denoted by X^* with the *empty word* e as the identity element. If $X \neq \emptyset$ then X^* is countably infinite. The *length of the word* u is denoted by $|u|$. By definition $|e| = 0$. Obviously $|u| \in \mathbb{N}$ for each $u \neq e$, $\mathbb{N} = \{1, 2, \ldots\}$ stands for the set of natural numbers.

For $u \in X^*$ and $v \in Y^*$, if $|u| = |v|$ we write $(u \mid v) \in (X \mid Y)^*$ to distinguish it from the case $(u, v) \in X^* \times Y^*$. We denote by $(X \mid Y)^*$ the set of all input–output pairs of words of the same length:

$$(X \mid Y)^* = \{(u \mid v) | u \in X^*, \ v \in Y^*, \ |u| = |v|\}.$$

Definition 3. Let $\mathscr{A} = (X, Q, Y, \mathscr{M})$ be FFM over the fuzzy algebra $\mathbb{I}_*$.

For any $(u \mid v) \in (X \mid Y)^*$ the *extended input–output behavior* of $\mathscr{A}$ upon the law of composition $*$ is determined by the square matrix $M(u \mid v)$ of order $|Q|$:

$$M(u|v) = \begin{cases} M(x_1|y_1) * \cdots * M(x_k|y_k), \\ \quad \text{if } (u \mid v) = (x_1 \ldots x_k \mid y_1 \ldots y_k), \quad k \geqslant 1, \\ U, \quad \text{if } (u \mid v) = (e \mid e), \end{cases} \tag{1}$$

where $M(x_1|y_1) * \cdots * M(x_k|y_k)$, for $* = \max - \min$, min–max or $\max - \odot$ matrix product (see Definition 1) and $U = (\delta_{ij})$ is the square matrix of order $|Q|$ with elements δ_{ij} determined as follows:

- if the composition is max–min or $\max - \odot$,

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

- if the composition is min–max,

$$\delta_{ij} = \begin{cases} 0, & \text{if } i = j, \\ 1, & \text{if } i \neq j. \end{cases}$$

We regard each element $m_{qq'}(u \mid v)$ in $M(u \mid v)$ as degree of membership that $\mathscr{A}$ will enter state $q\prime \in Q$ and produce output word $v \in Y^*$ under the input word $u \in X^*$ beginning at state $q \in Q$, after $|u| = |v|$ consecutive steps.

Definition 3 describes the extended input–output behavior of several types of finite fuzzy machines:

- Finite max–min fuzzy machines, when $* = \bullet$, first introduced in [29,30];
- Finite min–max fuzzy machines, when $* = \circ$, first introduced in [30,33];
- Finite max–product fuzzy machines, when $* = \odot$, first introduced in [32].

Example 1 in Appendix illustrates the main notions from this subsection.

3.2. Complete input–output behavior

When we consider FFM as a 'black box' we are not interested in the next state $q\prime$. This is the essence of the input–output behavior of $\mathscr{A}$, determined by column-matrices $T(u \mid v)_{|Q|\times 1}$ as follows:

$$T(u \mid v)_{|Q|\times 1} = (t_q(u \mid v)) = \begin{cases} M(u \mid v) * E, & \text{if } (u \mid v) \neq (e \mid e); \\ E, & \text{if } (u \mid v) = (e \mid e), \end{cases} \tag{2}$$

where E is the $|Q| \times 1$ column-matrix with all elements equal to 1 if the composition is max–min or $\max - \odot$ and with all elements equal to 0 if the composition is min–max.

Each element $t_q(u \mid v)$ of $T(u \mid v)$ in (2) determines the operation of $\mathscr{A}$ under the input word u beginning at state q and producing the output word v after $|u| = |v|$ consecutive steps.

For instance, if the FFM is max–min or max–product, the element

$$t_q(u \mid v) = \bigvee_{q' \in Q} (m_{qq'}(u \mid v))$$

gives the way of achieving maximal degree of membership under the input word u, beginning at state q and producing the output word v.

We denote by $T_{\mathscr{A}}$ the *complete behavior matrix* of $\mathscr{A}$. It is semi-infinite matrix with $n = |Q|$ rows and with columns $T(u \mid v)$, $(u \mid v) \in (X \mid Y)^*$, computed by (2) and ordered according to the lexicographical order in $(X \mid Y)^*$, see Table 1 (if $X \neq \emptyset$ and $Y \neq \emptyset$ then X^* and Y^* are countably infinite and $(X \mid Y)^*$ is lexicographically ordered).

Let us mention that $t_q(e \mid e) = 1$ for $i = 1, \ldots, n$ if FFM is max–min or max–product, $t_q(e \mid e) = 0$ for $i = 1, \ldots, n$ if FFM is $\min - \max$.

Table 1
$T_{\mathscr{A}}$ – initial fragment

	$T(e \mid e)$	$T(x_1 \mid y_1)$	$\cdots$	$T(u \mid v)$	$\cdots$
q_1	$t_{q_1}(e \mid e)$	$t_{q_1}(x_1 \mid y_1)$	$\cdots$	$t_{q_1}(u \mid v)$	$\cdots$
q_2	$t_{q_2}(e \mid e)$	$t_{q_2}(x_1 \mid y_1)$	$\cdots$	$t_{q_2}(u \mid v)$	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
q_n	$t_{q_n}(e \mid e)$	$t_{q_n}(x_1 \mid y_1)$	$\cdots$	$t_{q_n}(u \mid v)$	$\cdots$
length l	$l = 0$	$l = 1$	$\cdots$	$l = \|u\|$	$\cdots$

Examples 2 and 4 in Appendix illustrate computing initial fragment from $T_{\mathscr{A}}$ for words of length $\leqslant 2$ for max–min and max–product FFM.

4. Behavior matrix of finite fuzzy machine

For any finite fuzzy machine $\mathscr{A}$ its complete input–output behavior matrix $T_{\mathscr{A}}$ is semi-infinite – it has finite number of rows (equal to the number of states in Q) and infinite number of columns. Since $T_{\mathscr{A}}$ is semi-infinite, one can not solve traditional problems – equivalence of states, reduction of states, minimization of states, because all of them explore $T_{\mathscr{A}}$.

In this section, we propose algorithm for extracting a finite matrix $B_{\mathscr{A}}$ (called behavior matrix) from the complete behavior matrix $T_{\mathscr{A}}$. $B_{\mathscr{A}}$ captures all the properties of $T_{\mathscr{A}}$ and also provides solving equivalence, reduction and minimization problems for FFM.

In order to explain how to compute $B_{\mathscr{A}}$ we first provide supplementary information – what is linear combination, how to establish that a vector is a linear combination of a set of vectors and how we implement this for obtaining $B_{\mathscr{A}}$.

4.1. Linear combination

Let $A(1) = (a_{i1})_{n\times 1}, A(2) = (a_{i2})_{n\times 1}, \ldots, A(k) = (a_{ik})_{n\times 1}$ be column-vectors.

Definition 4. We say that a column-vector $C_{n\times 1}$ is:

- max–min *linear combination* of the vectors $A(i)_{n\times 1}$ with coefficients $x_i \in [0,1]$, $1 \leqslant i \leqslant k$, if
  $$C = [A(1) \wedge x_1] \vee \cdots \vee [A(k) \wedge x_k];$$
- min–max *linear combination* of the vectors $A(i)_{n\times 1}$ with coefficients $x_i \in [0,1]$, $1 \leqslant i \leqslant k$, if
  $$C = [A(1) \vee x_1] \wedge \cdots \wedge [A(k) \vee x_k];$$
- $\max-\odot$ *linear combination* of the vectors $A(i)_{n\times 1}$ with coefficients $x_i \in [0,1]$, $1 \leqslant i \leqslant k$, if
  $$C = [A(1) \cdot x_1] \vee \cdots \vee [A(k) \cdot x_k].$$

Checking whether $C_{n\times 1}$ is a linear combination of $A(1), \ldots, A(k)$ requires to solve the system

$$A * X = C$$

for the unknown X, if $A_{n\times k}$ has as columns $A(1) = (a_{i1})_{n\times 1}$, $A(2) = (a_{i2})_{n\times 1}$, …, $A(k) = (a_{ik})_{n\times 1}$.

In principle, when the system is consistent (inconsistent, respectively) the right-hand side vector C is (is not, respectively) linear combination of the vectors forming the matrix of coefficients A.

Example 3 in Appendix illustrates how we solve linear combination problem.

Implementation. The finite behavior matrix $B_{\mathscr{A}}$ contains only linearly independent columns from $T_{\mathscr{A}}$. Hence, when computing $B_{\mathscr{A}}$ from $T_{\mathscr{A}}$, each column from $T_{\mathscr{A}}$ that is a linear combination of the previous columns should be removed.

4.2. Algorithm for computing behavior matrix $B_{\mathscr{A}}$

We denote by $T(i)$ the finite submatrix of $T_{\mathscr{A}}$ containing the columns $T(u|v)$ for words of length not greater than i, $i \in \mathbb{N}$. Let $B(i)$ be a submatrix of $T(i)$ obtained by omitting all columns from $T(i)$ that are linear combination of the previous columns.

For arbitrary matrices C and D we write $C \subseteq D$, if each column of C is a column of D. If each column in D is a linear combination of columns from C, we write $D \cong C$. Obviously for each $i \in \mathbb{N}$, we have [20]:

(1) $T(i) \subseteq T(i+1) \subseteq \cdots \subseteq T$;
(2) $B(i) \subseteq B(i+1) \subseteq \cdots \subseteq B$;
(3) $B(i) \subseteq T(i)$.

For any FFM $\mathscr{A}$ we can obtain the matrix $B(i)$ from $T(i)$ for arbitrary i – it suffices to remove all columns from $T(i)$ that are linear combination of the previous columns. This is possible because we have the method and software to solve fuzzy linear system of equations [21,22], we develop functions for establishing linear dependence or linear independence [23,24] and here we develop functions for computing $B(i)$.

Definition 5. The matrix $B_{\mathscr{A}}$ obtained by omitting all columns from $T_{\mathscr{A}}$ that are linear combination of the previous columns is called **behavior matrix** of $\mathscr{A}$.

Theorem 3. *For any max–min or min–max FFM $\mathscr{A}$ the following statements hold:*

(1) *There exists $k \in \mathbb{N}$, such that $T(k) \cong T(k+1)$ and $B(k) = B_{\mathscr{A}}$.*
(2) *If $T(k) \cong T(k+1)$, then:*
   - *$T(k) \cong T(k+p) \cong \cdots \cong T_{\mathscr{A}}$ for each $p = 1, \ldots$;*
   - *$B(k) = B(k+p) = \cdots \cong B_{\mathscr{A}}$ for each $p = 1, \ldots$;*
(3) *$B_{\mathscr{A}} \cong T_{\mathscr{A}}$.*

The results for max–min FFM are proved in [20], their validity for min–max FFM follows by dualization principle – $(\vee, \wedge, \neg)$ is a dual triple [1]. These statements are not valid for max–product machines.

Algorithm. for computing the behavior matrix $B_{\mathscr{A}}$ for max–min or min–max FFM $\mathscr{A} = (X, Q, Y, \mathscr{M})$.

(1) Enter the set of matrices $\mathscr{M}$.
(2) Find $k \in \mathbb{N}$, such that $T(k) \cong T(k+1)$.
(3) Obtain $B(k) = B_{\mathscr{A}}$ excluding all linear combinations from $T(k)$.
(4) End.

It is established in [20] that the behavior matrix $B_{\mathscr{A}}$ is finite for max–min FFM $\mathscr{A} = (X, Q, Y, \mathscr{M})$ and the time complexity function for computing $B_{\mathscr{A}}$ is exponential. The same is valid for min–max FFM.

Computing behavior matrix using this algorithm is given in Examples 2 and 5 in Appendix.

5. Equivalence of states, reduction, minimization

Various equivalence problems and their algorithmical solvability are theoretically investigated in [20]. We demonstrate in this section how the software for finding $T_{\mathscr{A}}$ and $B_{\mathscr{A}}$ can be implemented for solving them.

Let the FFM $\mathscr{A} = (X, Q, Y, \mathscr{M})$ be given. The states $q_i \in Q$ and $q_j \in Q$ are called **equivalent** [17,20] if the input–output behavior of $\mathscr{A}$ when beginning with initial state q_i is the same as its input–output behavior when beginning with initial state q_j. As proved in [20], it means that the ith and jth rows are identical in $T_{\mathscr{A}}$ and in $B_{\mathscr{A}}$ (see Appendix, Examples 2 and 5). Since $T_{\mathscr{A}}$ is semi-infinite, we can not derive equivalence of states from it. In order to solve this problem we first have to extract $B_{\mathscr{A}}$ from $T_{\mathscr{A}}$ and then to find identical rows in $B_{\mathscr{A}}$.

FFM $\mathscr{A} = (X, Q, Y, \mathscr{M})$ is in **reduced form** if there does not exist equivalent states in Q.

Example 2 in Appendix illustrates equivalence of states and whether a FFM is in reduced form.

An FFM $\mathscr{A} = (X, Q, Y, \mathscr{M})$ is not in **minimal form** if the input–output behavior of $\mathscr{A}$ when it begins with initial state q_i is the same as its input–output behavior when it begins with initial distribution over Q, isolating the state q_i. Formally it means that the ith row of $T_{\mathscr{A}}$ is a linear combination of the other rows. Since $B_{\mathscr{A}}$ preserves all properties of $T_{\mathscr{A}}$ with respect to minimal forms, we transform the problem: if the ith row of $B_{\mathscr{A}}$ is a linear combination of the other rows of $B_{\mathscr{A}}$, the FFM $\mathscr{A}$ is not in minimal form. Hence for any FFM $\mathscr{A} = (X, Q, Y, \mathscr{M})$ with known behavior matrix $B_{\mathscr{A}}$, it is algorithmically solvable whether $\mathscr{A}$ is in minimal form [20].

Remark. All other equivalence, reduction and minimization problems as introduced in [20] are solved by suitable functions developed by the authors. We give here illustration only for some fundamental ones. The reason is that the theoretical background for these problems is tremendous, see [17,20] and this will embarrass the reader. The essence is that we propose software (under the request to the authors) for finding behavior matrix and solving equivalence, reduction and minimization problems for FFM.

6. Brief software description

The main functionality of the software is concentrated in **find_t** (computing $T_{\mathscr{A}}$) and **find_b** (computing $B_{\mathscr{A}}$) functions. The substantial difference between them is that **find_t** computes complete behavior matrix for a given fuzzy machine and therefore it does not remove the columns that are linear combination of the other columns, while **find_b** removes these columns. Function **find_b** can also reduce or minimize the computed behavior matrix, while **find_t** does not do this. Their parameters are described below.

For **find_t** they are:

- *m* – a cell array with all matrices from $\mathscr{M}$ representing FFM behavior for words of length 1.
- *composition* – the composition law for the type of FFM. It can be one of 'maxmin', 'minmax' or 'maxprod'.
- *word_length* – the maximum word length for which we compute the complete behavior matrix.

The function **find_t** returns the complete behavior matrix T_k, where k is the desired word length.

The parameters for **find_b** are:

- m – as above
- *composition* – as above
- *cond* – this parameter can be one of 'none', 'minimize' or 'reduce'. It defines if we want: to find behavior matrix ('none'), to make minimization ('minimize') or reduction ('reduce').
- *word_length* – the default value is '−1' which is used for 'unlimited'. In this case the function returns behavior matrix for the given fuzzy machine.

The function **find_b** returns the behavior matrix B_k when k is the desired word length or B for 'unlimited'. In case of reduction or minimization the answer is behavior matrix of reduced (minimized, respectively) form machine.

This algorithm has exponential time complexity and exponential memory complexity. If we have $|X|$ input letters and $|Y|$ output letters, we produce $(|X| \cdot |Y|)^k$ matrices, where k is the length of the word.

7. Conclusions

In this paper we define finite fuzzy machines over fuzzy algebra and propose algorithm and software for solving open problems, namely, given FFM:

- to compute initial fragment of its complete behavior matrix;
- to compute its behavior matrix (when possible);
- to establish equivalence of states;
- to obtain its reduced form behavior matrix;
- to obtain its minimal form behavior matrix.

The described software may be extended for other types of finite machines – weighted, probabilistic, deterministic, nondeterministic, over semiring, over Łukasiewicz algebra (using corresponding new composition).

Acknowledgements

The authors would like to express their gratitude to the anonymous reviewers for their constructive comments and proposals that improve this exposition.

Appendix A

Example 1. Compute $M(xx|y_1y_2)$ if the max–product FFM $\mathscr{A} = (X, Q, Y, \mathscr{M})$ is given with the following data:

$$X = \{x\}, \quad Y = \{y_1, y_2\}, \quad Q = \{q_1, q_2\},$$
$$m1 = M(x \mid y_1) = \begin{pmatrix} 0.3 & 0.7 \\ 0.5 & 0.2 \end{pmatrix}, \quad m2 = M(x \mid y_2) = \begin{pmatrix} 0.4 & 0.6 \\ 0.2 & 0.8 \end{pmatrix}.$$

Its input–output behavior for $(xx \mid y_1y_2)$ according to (1) is

$$M(xx \mid y_1y_2) = M(x \mid y_1) \odot M(x \mid y_2) = \begin{pmatrix} 0.3 & 0.7 \\ 0.5 & 0.2 \end{pmatrix} \odot \begin{pmatrix} 0.4 & 0.6 \\ 0.2 & 0.8 \end{pmatrix} = \begin{pmatrix} 0.14 & 0.56 \\ 0.2 & 0.3 \end{pmatrix}.$$

Obviously, computing input–output behavior of the FFM implements direct problem resolution from fuzzy relational calculus [21,22]. Hence, the functions for direct problem resolution **fuzzy_maxmin** [21], **fuzzy_minmax** [21], **fuzzy_maxprod** [22] may be used to compute $M(u|v)$ for arbitrary $(u|v) \in (X \mid Y)^*$. For instance, calculation for $M(xx \mid y_1y_2)$ with **fuzzy_maxprod** results:

```
≫ ml2 = fuzzy_maxprod(ml, m2)
ml2 =
 0.1400    0.5600
 0.2000    0.3000
```

We give example for computing $T_{\mathscr{A}}$ for words of fixed length. Computations are made by two different computational ways: with software from [21,22] or with a new software developed by the authors.

Example 2. Compute initial fragment from $T_{\mathscr{A}}$ for words of length $\leqslant 2$, find the behavior matrix, find equivalent states and the behavior matrix of reduced FFM if the max–min FFM $\mathscr{A} = (X, Q, Y, \mathscr{M})$ is given with the following data:

$$X=\{x_1,x_2\},\quad Y=\{y_1,y_2\},\quad Q=\{q_1,q_2,q_3\},$$

$$m1=M(x_1\mid y_1)=\begin{pmatrix}0 & 0.6 & 0.5\\ 0.6 & 0.1 & 0.5\\ 0.2 & 0.1 & 0.2\end{pmatrix}, \tag{A.1}$$

$$m2=M(x_1\mid y_2)=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0.2 & 0.1 & 0.1\end{pmatrix}, \tag{A.2}$$

$$m3=M(x_2\mid y_1)=\begin{pmatrix}0.4 & 0.2 & 0.1\\ 0.3 & 0.4 & 0.1\\ 0 & 0 & 0\end{pmatrix}, \tag{A.3}$$

$$m4=M(x_2\mid y_2)=\begin{pmatrix}0 & 0.3 & 0.2\\ 0.3 & 0.1 & 0.2\\ 0.1 & 0 & 0.1\end{pmatrix}. \tag{A.4}$$

- First we compute initial fragment from $T_{\mathscr{A}}$ for words of length $\leqslant 2$.

I way. We implement expressions (1) and (2) and software from [21].

(a) Input the matrices (A.1)–(A.4) and E in MATLAB workspace

```
≫m1=[0 0.6 0.5; 0.6 0.1 0.5; 0.2 0.1 0.2]
≫m2=[0 0 0; 0 0 0; 0.2 0.1 0.1]
≫m3=[0.4 0.2 0.1; 0.3 0.4 0.1; 0 0 0]
≫m4=[0 0.3 0.2; 0.3 0.1 0.2; 0.1 0 0.1]
≫E=[1 1 1]
```

(b) Calculate the corresponding columns from $T_{\mathscr{A}}$ for words of length 1. Let

$$T1=T(x_1\mid y_1),\quad T2=T(x_1\mid y_2),\quad T3=T(x_2\mid y_1),\quad T4=T(x_2\mid y_2).$$

From (A.1)–(A.4), applying (2) and the function **fuzzy_maxmin** from [21], we compute the behavior of the max–min FFM $\mathscr{A}=(X,Q,Y,\mathscr{M})$ for the words of length 1. In this case the function **fuzzy_maxmin** results in the maximal element in each row of the matrices (A.1)–(A.4):

```
≫T1=fuzzy_maxmin(m1,E')
T1=
0.6000
0.6000
0.2000
≫T2=fuzzy_maxmin(m2,E')
T2=
0
0
0.2000
≫T3=fuzzy_maxmin(m3,E')
T3=
0.4000
0.4000
0
≫T4=fuzzy_maxmin(m4,E')
T4=
0.3000
0.3000
0.1000
```

(c) Multiplying (A.1)–(A.4) by two's and applying (2), we obtain the behavior of the max–min FFM $\mathscr{A}=(X,Q,Y,\mathscr{M})$ for words of length 2.

(c1) An initial fragment of computations, by expression (1), is presented below:

$$m11 = M(x_1x_1 \mid y_1y_1) = M(x_1 \mid y_1) \bullet M(x_1 \mid y_1) = \begin{pmatrix} 0.6 & 0.1 & 0.5 \\ 0.2 & 0.6 & 0.5 \\ 0.2 & 0.2 & 0.2 \end{pmatrix},$$

$$m12 = M(x_1x_1 \mid y_1y_2) = M(x_1 \mid y_1) \bullet M(x_1 \mid y_2) = \begin{pmatrix} 0.2 & 0.1 & 0.1 \\ 0.2 & 0.1 & 0.1 \\ 0.2 & 0.1 & 0.1 \end{pmatrix},$$

$$m13 = M(x_1x_2 \mid y_1y_1) = M(x_1 \mid y_1) \bullet M(x_2 \mid y_1) = \begin{pmatrix} 0.3 & 0.4 & 0.1 \\ 0.4 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.1 \end{pmatrix}, \ldots$$

We apply (2) to the last three matrices and obtain:

$$T11 = T_{11|11} = T(x_1x_1 \mid y_1y_1) = \begin{pmatrix} 0.6 \\ 0.6 \\ 0.2 \end{pmatrix},$$

$$T12 = T_{11|12} = T(x_1x_1 \mid y_1y_2) = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \end{pmatrix},$$

$$T13 = T_{11|21} = T(x_1x_1 \mid y_2y_1) = \begin{pmatrix} 0.4 \\ 0.4 \\ 0.2 \end{pmatrix}, \ldots$$

(c2) These results, but from software session with the function **fuzzy_maxmin** from [21], are displayed below:

```
≫ m11 = fuzzy_maxmin(m1,m1)
m11 =
 0.6000    0.1000    0.5000
 0.2000    0.6000    0.5000
 0.2000    0.2000    0.2000

≫ T11 = fuzzy_maxmin(m11,E')
T11 =
0.6000
0.6000
0.2000
≫ m12 = fuzzy_maxmin(m1,m2)
m12 =
 0.2000    0.1000    0.1000
 0.2000    0.1000    0.1000
 0.2000    0.1000    0.1000

≫ T12 = fuzzy_maxmin(m12,E')
T12 =
0.2000
0.2000
0.2000
≫ m13 = fuzzy_maxmin(m1,m3)
m13 =
 0.3000    0.4000    0.1000
 0.4000    0.2000    0.1000
 0.2000    0.2000    0.1000
≫ T13 = fuzzy_maxmin(m13,E')
T13 =
0.4000
0.4000
0.2000
```

It is obvious that this method of computation is tremendous even for words having fixed small length. This motivated us to develop functions for computing $T_{\mathscr{A}}$, as explained in the next II way.*II way.* We develop software for computing the initial fragment T_k of the matrix $T_{\mathscr{A}}$ for words with fixed length $\leqslant k$ for finite fuzzy machines. The function (briefly described in Section 6) is:**find_t(m, composition, word_length)** where

- **m** are the initial data matrices from the set $\mathscr{M}$ of the FFM.
- **composition** is the composition used for the type of the FFM. It can be one of 'maxmin', 'minmax' or 'maxprod'.
- **word_length** is the desired word length k. It also gives the number of executive steps for finding T_k.

Function **find_t** is developed using theoretical results for finite fuzzy machines as given in this paper. It implements functions from [21,22] and automatizes all operations as described in the I way of Example 2.

The function **find_t(m,'maxmin',2)** results (compare with Table A.1) in:

```
≫ find_t(m,'maxmin',2)

step = 1; time = 0.0013616;

t =

1.0000   0.6000   0        0.4000   0.3000

1.0000   0.6000   0        0.4000   0.3000

1.0000   0.2000   0.2000   0        0.1000

step = 2; time = 0.0003087;

t =

Columns 1 through 7

1.0000   0.6000   0        0.4000   0.3000   0.6000   0.2000

1.0000   0.6000   0        0.4000   0.3000   0.6000   0.2000

1.0000   0.2000   0.2000   0        0.1000   0.2000   0.2000

Columns 8 through 14

0.4000   0.3000   0        0        0        0        0.4000

0.4000   0.3000   0        0        0        0        0.4000

0.2000   0.2000   0.2000   0.1000   0.2000   0.2000   0

Columns 15 through 21

0.1000   0.4000   0.3000   0.3000   0.2000   0.3000   0.3000

0.1000   0.4000   0.3000   0.3000   0.2000   0.3000   0.3000

0        0        0        0.1000   0.1000   0.1000   0.1000
```

Table A.1
An initial fragment from $T_{\mathscr{A}}$, Example 2 – I way of computation

	$T0$	$T1$	$T2$	$T3$	$T4$	$T11$	$T12$	$T13$	⋯
q_1	1	0.6	0	0.4	0.3	0.6	0.2	0.4	⋯
q_2	1	0.6	0	0.4	0.3	0.6	0.2	0.4	⋯
q_3	1	0.2	0.2	0	0.1	0.2	0.2	0.2	⋯
l	$l=0$	$l=1$				$l=2$			

- Compute the behavior matrix for the given max–min FFM. The function **find_b** if **composition='maxmin'** results in $T(1) = T(2)$, and hence $B = T(1)$:

```
≫ find_b(m,'maxmin')

step = 1; time = 0.0011988;

t =

1.0000   0.6000   0        0.4000

1.0000   0.6000   0        0.4000

1.0000   0.2000   0.2000   0
step = 2; time = 0.0044941;

t =

1.0000   0.6000   0        0.4000

1.0000   0.6000   0        0.4000

1.0000   0.2000   0.2000   0
ans =

1.0000   0.6000   0        0.4000

1.0000   0.6000   0        0.4000

1.0000   0.2000   0.2000   0
```

- Find equivalent states and behavior matrix of the reduced FFM.

Function **find_b** establishes equivalence of states and whether the max–min, min–max or max $-\odot$ FFM is in reduced form, if cond='reduce'.

The max–min machine is not in reduced form:

```
≫ find_b(m,'maxmin','reduce')

step = 1; time = 0.0013407;

t =

1.0000   0.6000   0        0.4000

1.0000   0.6000   0        0.4000

1.0000   0.2000   0.2000   0
step = 2; time = 0.0044715;

t =

1.0000   0.6000   0        0.4000

1.0000   0.6000   0        0.4000

1.0000   0.2000   0.2000   0
```

Since the first and the second row in $B_{\mathscr{A}}$ are identical, the states q_1 and q_2 are equivalent and $\mathscr{A} = (X, Q, Y, \mathscr{M})$ is not in reduced form. The behavior matrix of the reduced form FFM follows:

```
ans =
1.0000   0.6000   0        0.4000
1.0000   0.2000   0.2000   0
```

Example 3. With data in Example 2, the vector $T4$ is max–min linear combination of the vectors $T2$ and $T3$, because

$$T4 = (T2 \wedge 0.1) \vee (T3 \wedge 0.3).$$

For checking whether C is a linear combination of $A(i)$, $1 \leqslant i \leqslant k$, we implement software for inverse problem resolution from [21,22] for solving fuzzy linear system of equations. For instance using software from [21] for the system formed by $T2$ and $T3$ as matrix of coefficients A and with right hand side $T4$

$$\begin{pmatrix} 0 & 0.4 \\ 0 & 0.4 \\ 0.2 & 0 \end{pmatrix} \bullet X = \begin{pmatrix} 0.3 \\ 0.3 \\ 0.1 \end{pmatrix},$$

we obtain that the system is consistent with solution $x_1 = 0.1, x_2 = 0,3$, i.e. $T4$ is a max–min linear combination of $T2$ and $T3$.

Example 4. Find $T(2)$ for max–product FFM with the same data as in Example 2.

We obtain $T(2)$ after calling the function **find_t** if **composition='maxprod'** and **word_length=2**:

```
≫ find_t(m,'maxprod',2)

step = 1; time = 0.00012152;

t =

1.0000   0.6000   0        0.4000   0.3000

1.0000   0.6000   0        0.4000   0.3000

1.0000   0.2000   0.2000   0        0.1000
step = 2; time = 0.00022964;

t =

Columns 1 through 7

1.0000   0.6000   0        0.4000   0.3000   0.3600   0.1000

1.0000   0.6000   0        0.4000   0.3000   0.3600   0.1000

1.0000   0.2000   0.2000   0        0.1000   0.1200   0.0400
Columns 8 through 14

0.2400   0.1800   0        0        0        0        0.2400

0.2400   0.1800   0        0        0        0        0.2400

0.0800   0.0600   0.1200   0.0200   0.0800   0.0600   0
Columns 15 through 21

0.0200   0.1600   0.1200   0.1800   0.0400   0.1200   0.0900

0.0200   0.1600   0.1200   0.1800   0.0400   0.1200   0.0900

0        0        0        0.0600   0.0200   0.0400   0.0300
```

Using matrix $T(2)$ we see that the first and the second rows in $T(2)$ are identical, the states q_1 and q_2 are equivalent for words of length no longer than 2.

Example 5. With the same data as in Example 2, but for min–max FFM we obtain B after calling the function **find_b** if **composition='minmax'**. The result is $T(1) = T(2)$ and hence $B = T(1)$:

```
≫ find_b(m,'minmax')


step = 1; time = 0.0015471;

t =

0  0       0       0.1000

0  0.1000  0       0.1000

0  0.1000  0.1000  0


step = 2; time = 0.0070741;

t =

0  0       0       0.1000

0  0.1000  0       0.1000

0  0.1000  0.1000  0

ans =
0  0       0       0.1000

0  0.1000  0       0.1000

0  0.1000  0.1000  0
```

Using this behavior matrix, we establish that the min–max machine is not in reduced form – the first and the second rows in $B_{\mathscr{A}}$ are identical, the states q_1 and q_2 are equivalent.

References

[1] B. De Baets, Analytical solution methods for fuzzy relational equations, in: D. Dubois, H. Prade (Eds.), Fundamentals of Fuzzy Sets, The Handbooks of Fuzzy Sets Series, vol. 1, Kluwer Academic Publishers, 2000, pp. 291–340.
[2] S.R. Chaudhari, Homomorphisms of fuzzy recognizers, Kybernetes 36 (5–6) (2007) 768–775.
[3] W. Cheng, Z.W. Mo, Minimization algorithm of fuzzy finite automata, Fuzzy Sets and Systems 141 (2004) 439–448.
[4] A. Di Nola, A. Lettieri, Relation Equations in Residuated Lattices, Rendiconti del Circolo Matematico di Palermo, s. II, XXXVIII, 1989, pp. 246–256.
[5] A. Di Nola, A. Lettieri, I. Perfilieva, V. Novák, Algebraic analysis of fuzzy systems, Fuzzy Sets and Systems 158 (1) (2007) 1–22.
[6] P. Hájek, Metamathematics of Fuzzy Logic, Kluwer, Dordrecht, 1998.
[7] G. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice Hall PTR, NJ, 1995.
[8] S. Konstantinidis, S. Nikolae, S. Yu, Fuzzification of rational and recognizable sets, Fundamenta Informaticae 76 (4) (2007) 413–447.
[9] H. Lei, Y. Li, Reduction and minimization algorithm of synchronous lattice-valued automata, Computer Engineering and Applications 42 (16) (2006) 57–60.
[10] H. Lei, S. Li, Algorithm for determining equivalence between lattice-valued finite automata, Computer Engineering and Applications 42 (22) (2006) 39–41. 45.
[11] H. Lei, Y. Li, Reduction and minimization algorithm of synchronous lattice-valued automata, Computer Engineering and Applications 42 (16) (2006) 57–60.
[12] H.X. Lei, Y.M. Li, Minimization of states in automata theory based on finite lattice-ordered monoids, Information Sciences 177 (6) (2007) 1413–1421.
[13] Y. Li, A categorical approach to lattice-valued fuzzy automata, Fuzzy Sets and Systems 157 (6) (2006) 855–864.
[14] Z. Li, P. Li, Y. Li, The relationships among several types of fuzzy automata, Information Sciences 176 (15) (2006) 2208–2226.
[15] Y. Li, W. Pedrycz, Minimization of lattice finite automata and its application to the decomposition of lattice languages, Fuzzy Sets and Systems 158 (2007) 1423–1436.
[16] Y. Li, W. Pedrycz, Fuzzy finite automata and regular expressions with membership values in lattice-ordered monoids, Fuzzy Sets and Systems 156 (2005) 68–92.
[17] J.N. Mordeson, D.S. Malik, Fuzzy Automata and Languages – Theory and Applications, Chapman & Hall/CRC, London, 2002.

[18] K. Peeva, Equivalence, reduction and minimization of finite automata over semirings, Theoretical Computer Science 88 (1991) 269–285.
[19] K. Peeva, Behaviour, reduction and minimization of finite L-automata, Fuzzy Sets and Systems 28 (2) (1988) 171–181.
[20] K. Peeva, Finite L-fuzzy machines, Fuzzy Sets and Systems 141 (3) (2004) 415–437.
[21] K. Peeva, Y. Kyosev, Fuzzy relational calculus-theory, Applications and software (with CD-ROM), Advances in Fuzzy Systems – Applications and Theory, vol. 22, World Scientific, Publishing Company, 2004. CD-ROM %3chttp://mathworks.net%3e.
[22] K. Peeva, Y. Kyosev, Algorithm for solving max–product fuzzy relational equations, Soft Computing 11 (7) (2007) 593–605.
[23] K. Peeva, Zl. Zahariev, Linear dependence in fuzzy algebra, in: Proceedings of 31st International Conference AME, Sozopol, June 2005, Softrade, 2006, pp. 71–83.
[24] K. Peeva, Zl. Zahariev, Software for testing linear dependence in fuzzy algebra, in: Second International Scientific Conference Computer Science, Chalkidiki, 30 September–2 October 2005, Part I, 2006, pp. 294–299.
[25] I. Perfilieva, L. Nosková, System of fuzzy relation equations with inf-→ composition: complete set of solutions, Fuzzy Sets and Systems, in press., doi:10.1016/j.fss.2007.12.012.
[26] G. Rahonis, Infinite fuzzy computations, Fuzzy Sets and Systems 153 (2) (2005) 275–288.
[27] A.K. Ray, B. Chattererjee, A.K. Majumdar, A formal power series approach to the construction of minimal fuzzy automata, Information Sciences 55 (1–3) (1991) 189–207.
[28] E. Sanchez, Resolution of composite fuzzy relation equations, Information and Control 30 (1976) 38–48.
[29] E.S. Santos, Maximin automata, Information and Control 13 (1968) 363–377.
[30] E.S. Santos, Maximin, minimax and composite sequential machines, Journal of Mathematical Analysis and Applications 24 (1968) 246–259.
[31] E.S. Santos, On reduction of maxi-min machines, Journal of Mathematical Analysis and Applications 40 (1972) 60–78.
[32] E.S. Santos, Max-product machines, Journal of Mathematical Analysis and Applications 37 (1972) 677–686.
[33] E.S. Santos, W.G. Wee, General formulation of sequential machines, Information and Control 12 (1) (1968) 5–10.
[34] H. Xing, D. Qui, Fuchun Liu, Z. Fan, Equivalence in automata theory based on complete residuated lattice-valued logic, Fuzzy Sets and Systems 158 (13) (2007) 1407–1422.