Minimal Determinization Algorithm for Fuzzy Automata

Aitor G. de Mendívil Grau Stefan Stanimirović Federico Fariña

***Abstract*—The determinization of fuzzy automata is a well-studied problem in theoretical computer science celebrated for its practical applications. Indeed, in the fields of fuzzy discrete event systems, fault diagnosis, clinical monitoring, decision-making systems, and model checking, when a suitable model of a fuzzy automaton is employed, it is desirable to find its language-equivalent deterministic version because of its computational efficiency. Although many methods have been developed to convert a fuzzy automaton to its language equivalent fuzzy deterministic finite automaton (FDfA), they can be applied only for fuzzy automata defined over specific underlying sets of truth values. For example, recently developed determinization methods employ the concept of maximal factorization, which can be defined only on non-locally finite lattices or the Boolean lattice. In addition, not all such determinization methods result in a minimal FDfA. On the other hand, even though such determinization methods have been developed for fuzzy automata over specific underlying structures, these methods cannot be generalized for fuzzy automata over locally finite lattices. This article focuses on filling this gap and develops a novel method for computing a minimal FDfA for a fuzzy automaton defined over a locally finite and divisible residuated lattice. Our method uses the new concept of a reduction graph that emerges from the strict order relation on the resulting fuzzy states, according to which we can construct all minimal FDfAs equivalent to a given fuzzy automaton.**

***Index Terms*—Fuzzy finite automata, minimal determinization method, complete deterministic fuzzy automaton, Brzozowski's procedure, factorization of fuzzy states, locally finite lattices.**

I. Introduction

Nondeterministic finite automata (NfAs) are suitable mathematical models for designing language recognizers; however, deterministic finite automata (DfAs) are more computationally efficient. This is the main reason for designing procedures to convert a NfA into an equivalent DfA [1][2]. It is well-known that the worst-case time complexity of this conversion is exponential in the size of the input NfA. One way to offset this cost is that the *determinization procedure* outputs a minimal DfA instead of any DfA equivalent to the input automaton. Brzozowski's double reversal determinization algorithm is one of the best-known *minimal determinization procedures* [3].

Fuzzy finite automata (FfAs) effectively generalize NfAs and have practical applications in environments where uncertainty is naturally present, including fuzzy discrete event systems, decision making, fault diagnosis, clinical monitoring, artificial intelligence, and model checking (see [4], [5], [6], [7], [8], [9], [10], [11] for concrete examples). In all such practical situations, it is suitable to transform a FfA to an equivalent deterministic version of it. As an output of the determinization method, this paper uses the notion of the *fuzzy deterministic finite automaton* (FDfA) [12]. In a FDfA, initial state, final states, and transitions are labelled with any truth value. Thus, FDfAs include, as a particular case, ordinary DfAs equipped with fuzzy final states. Such fuzzy automata are also called crisp deterministic fuzzy finite automata (cDFfAs) in the literature [13]. An FDfA is minimal if it has the smallest size among all FDfAs equivalent to it. From the definitions of FDfAs and cDFfAs, we can conclude that: (i) the size of a minimal FDfA is always less than or equal to the size of a minimal cDFfA equivalent to it since a cDFfA is a particular case of an FDfA; and (ii) a minimal FDfA may not be unique, as there may exist many equivalent FDfAs of the same size that may have different values in their components, different topology, or both.

In this paper, we deal with FfAs with membership values in a complete residuated lattice (denoted $\mathcal{L}$ for short) [14][15]. Our main goal is to design a determinization procedure for FfAs with membership values in a *divisible* and *locally finite* $\mathcal{L}$ that returns a minimal FDfA equivalent to the input FfA. As there may not exist the unique minimal FDfA, the design of such a procedure is challenging.

A. Related works

We provide an overwiev of literature that has as an objective the design of a minimal determinization procedure for FfAs over $\mathcal{L}$. FfAs over a *locally finite* $\mathcal{L}$ only recognize fuzzy languages of finite image [13][16]; however, they are always determinizable by the *fuzzy accessible subset construction* provided in [17]. The output cDFfA is called the *Nerode automaton* of A [18], denoted $N(A)$. Further improvements of this construction have been achieved in [19], [20]. In [21], Jančić and Ćirić apply the construction $N(r(N(r(A))))$[1] to obtain an equivalent minimal cDFfA to A. In [22], Micić et al. propose the method of a degree of language inclusion and obtain a minimal cDFfA. Determinization methods whose objective is to convert a FfA to an equivalent (minimal) cDFfA do not achieve the primary goal of this paper by the argument given in (i) above.

FfAs over a *non-locally finite* $\mathcal{L}$ may recognize fuzzy languages of infinite image. Therefore, determinization methods for such kind of automata must necessarily return equivalent

Aitor G. de Mendívil Grau is with the Departamento de Estadística, Informática y Matemáticas, Universidad Pública de Navarra, Pamplona, Spain, 31006.
E-mail: aitor.gonzalezdemendivil@unavarra.es
Stefan Stanimirović is with the University of Niš, Faculty of Sciences and Mathematics, Višegradska 33, 18000 Niš, Serbia
E-mail: stefan.stanimirovic@pmf.edu.rs
Federico Fariña is with the Departamento de Estadística, Informática y Matemáticas, Universidad Pública de Navarra, Pamplona, Spain, 31006.
E-mail: fitxi@unavarra.es
Manuscript received March, 2023

[1] $r(A)$ denotes the reverse automaton of A.

FDfAs. Determinization based on *factorization* of fuzzy states is introduced in [23] to convert a FfA A over a *divisible* $\mathcal{L}$ into an equivalent FDfA, denoted $D(A)$. The construction $D(r(D(r(A))))$ is applied in [24] to obtain a minimal FDfA equivalent to A when the factorization of fuzzy states is *maximal*. A drawback of this construction is that many FfAs are not determinizable. A variant of this construction has been provided in [25] using the notion of right invariant quasi-orders and an extended determinization via factorization [26]. Conditions for the determinization of FfAs using maximal factorizations have been studied in [23], [24], [26]. Unfortunately, these methods are not applicable to FfAs over a locally finite $\mathcal{L}$ because there is no maximal factorization for such lattices except for the Boolean lattice [27].

To our knowledge, the only determinization methods that return a minimal FDfA over a locally finite $\mathcal{L}$ have been proposed in [28] and [29]. The method developed by Li and Qiu [28] is applicable for FfAs over totally ordered lattices using $\vee$ and $\wedge$ as underlying operations. Their method is based on solving a finite set of fuzzy polynomial equations and has not been generalized for other lattices. In [29], the authors proposed a method based on two phases: (phase i) the procedure first constructs $D(r(N(r(A)))$; and, (phase ii) this output automaton reduces its size when some pairs of states fulfil certain particular conditions. This method is more efficient than the previous one (see Table 4 in [29]).

The particular properties of the factorization of fuzzy states over the Gödel structure that allow to define the conditions for state reduction in (phase ii) [29], cannot be generalized to other divisible $\mathcal{L}$. We know, by the results in [23][26], that the construction $D(r(N(r(A)))$ is applicable to FfAs over a divisible and locally finite $\mathcal{L}$. This fact motivates the study of the properties of this construction for such lattices. However, the idea that the size of $D(A)$ is less than the size of $N(A)$ for any FfA A over the Gödel structure is wrong. A counterexample to Property 2 in [29] is provided in Figure 7 from [26]. In the method we carry out, we must ensure automata finiteness since no result in the literature guarantees that FfAs over divisible and locally finite $\mathcal{L}$ are always determinizable via factorization.

B. Paper contributions and organization

In this paper, we denote by A_c a crisp co-accessible co-deterministic FfA (cCFfA) equivalent to a FfA A. Properties of A_c are indicated in subsection II-C after presenting common preliminaries for lattices, factorization and FfAs (Section II). Section III presents FDfAs and their conditions for minimality. The determinization method via factorization of fuzzy states and a method for state reduction are briefly introduced in Section IV. In Section V, we state our first contribution (Theorem 1): the automaton $D(A_c)$, obtained by the determinization via factorization of a cCFfA, is a fuzzy deterministic *finite* automaton that satisfies the necessary conditions for minimality. Moreover, its size is always less than or equal to the size of a minimal cDFfA equivalent to it. This result is possible thanks to the fact that the factorization of the fuzzy states in $D(A_c)$ behaves like a maximal factorization for the values obtained through the paths of such an automaton (Lemma 3). $D(A_c)$ may not be a minimal FDfA; however, we demonstrate (Theorem 3) that, in that case, its fuzzy states fulfill a strict order relation $\ll$ that defines a graph, denoted $\mathcal{G}(D(A_c))$, called the *reduction graph of* $D(A_c)$. Using this graph, we can build all possible minimal FDfAs equivalent to A. This new contribution is provided in Section VI together the algorithms to obtain a minimal FDfA equivalent to A. The Supplementary Material contains the proofs of the results not provided in the paper.

II. Preliminaries

In this paper, we use complete residuated lattices as structures of membership values. A *complete residuated lattice* is an algebra $\mathcal{L} = (L, \vee, \wedge, \otimes, \rightarrow, 0, 1)$ such that

(L1) $(L, \vee, \wedge, 0, 1)$ is a *complete lattice*;
(L2) $(L, \otimes, 1)$ is a commutative monoid;
(L3) for all $x, y, z \in L$: $x \otimes y \leq z \Leftrightarrow x \leq y \rightarrow z$.

Operations $\otimes$ and $\rightarrow$ form an *adjoint pair* (L3). They are intended for modeling the conjunction and implication of the corresponding logical calculus. Supremum ($\bigvee$) and infimum ($\bigwedge$) are interpreted as the existential and general quantifier, respectively. With respect to the order $\leq$, $\otimes$ is isotonic in both arguments, and $\rightarrow$ is isotonic in the second and antitonic in the first argument. In addition, for any $x, y, z, w \in L$,

(L4) $x \otimes y \leq x$ and $x \leq y \rightarrow x$;
(L5) $x \rightarrow x = 1$ and $1 \rightarrow x = x$;
(L6) $(x \rightarrow y) \otimes (z \rightarrow w) \leq (x \otimes z) \rightarrow (y \otimes w)$;
(L7) $x \otimes (x \rightarrow y) = y$ if and only if $(\exists z)\, x \otimes z = y$;
(L8) $(x \otimes y) \rightarrow z = x \rightarrow (y \rightarrow z) = y \rightarrow (x \rightarrow z)$;
(L9) $x \otimes \bigvee_{i \in I} y_i = \bigvee_{i \in I} x \otimes y_i$, for any $\{y_i\}_{i \in I} \subseteq L$.

Further properties of complete residuated lattices are in [30], [31], [32]. In this paper, we assume that $\mathcal{L}$ satisfies the following conditions:

(C1) L is a **totally ordered** set w.r.t $\leq$.
(C2) $\mathcal{L}$ is **divisible**, i.e., for every $x, y \in L$ with $x \geq y$ there exists $z \in L$ such that $x \otimes z = y$.
(C3) $(L, \otimes, 1)$ is **zero-divisor free**: for every $x, y \in L$, $x \otimes y = 0$ if and only if $x = 0$ or $y = 0$.
(C4) The algebra $\mathcal{L}^* = (L, \vee, \otimes, 0, 1)$, obtained from $\mathcal{L}$, is **locally finite** (cf. [16]).

In the following, we present the notion of fuzzy states and fuzzy state-transition relations. Let Q be a finite non-empty set of states. Any mapping $S : Q \rightarrow L$, briefly $S \in L^Q$, is called a *fuzzy state* of Q. The degree of membership of each state q in a fuzzy state S is the value $S(q)$. Equality ($=$) and inclusion (ordering, $\leq$) of fuzzy states are defined coordinate-wise as it is common in fuzzy sets [30][31]. The *support* of a fuzzy state $S \in L^Q$ is the subset of states defined by $\text{Supp}(S) = \{q \in Q | S(q) \neq 0\}$. In this paper, $\mathbf{0}$ denotes the fuzzy state in which all states have value 0, i.e., $\text{Supp}(\mathbf{0}) = \emptyset$.

Any mapping $T : Q \times Q \rightarrow L$, briefly $T \in L^{Q \times Q}$, is called a *fuzzy (state-transition) relation* on Q. The degree of a transition from a state p to q under a fuzzy relation T is represented by the membership value $T(p, q)$. The (crisp) *equality relation* on Q is denoted by E_Q, where $E_Q(p, p) = 1$ for any $p \in Q$ and 0 otherwise.

In this paper, we use the standard fuzzy definition of composition operation ($\bigvee -\otimes$) for fuzzy relations and fuzzy subsets which is commonly denoted by $\circ$. In addition, we use $x \otimes S$ to denote the fuzzy subset on Q such that $(x \otimes S)(q) = x \otimes S(q)$ for any $x \in L$, $S \in L^Q$, and $q \in Q$.

A. Factorization of fuzzy states

Given $\mathcal{L}$ and a set of states Q (non-empty), the function $g : L^Q \to L$ is defined as follows for any $S \in L^Q$,

$$g(S) = \bigvee_{q \in Q} S(q) \text{ with } S \neq \mathbf{0}; \text{ and } g(\mathbf{0}) = 1 \tag{1}$$

By using the previous function $g(.)$, we define a new function $f : L^Q \to L^Q$ as follows

$$f(S)(q) = g(S) \to S(q) \tag{2}$$

for any $S \in L^Q$ and $q \in Q$. Let us observe that $g(.)$ and $f(.)$ are well defined functions. The pair of functions (f, g) satisfy the following properties for any $S \in L^Q$, $q \in Q$ and $x \in L$:

(F1) $g(S) \geq S(q)$;
(F2) $g(S) > 0$;
(F3) $f(S) \geq S$;
(F4) $f(S)(q) = 0 \Leftrightarrow S(q) = 0$;
(F5) $g(S) \otimes f(S) = S$;
(F6) $g(x \otimes S) = x \otimes g(S)$ with $x \neq 0$;
(F7) $f(S) \leq f(x \otimes S)$ with $x \neq 0$;
(F8) $f(S) = f(f(S))$;
(F9) $S \neq \mathbf{0} \Rightarrow \exists q : f(S)(q) = 1$

The pair (f, g) is a factorization of L^Q in the sense given in [23]. Properties (F1)-(F9) were proved in [23][26]. The factorization (f, g) depends on both $\mathcal{L}$ and Q. Each particular factorization will be clear in its context of application without introducing further notation.

B. Fuzzy languages and automata

As usual, an *alphabet* is a finite non-empty set of symbols denoted by Σ, and the set of words is the free monoid over Σ denoted by Σ^*. We denote the empty word Σ^* by ε. In other words, a word $\sigma \in \Sigma^*$ of length $|\sigma| = n$, $n \geq 0$, is a finite sequence $\sigma = \sigma_1 ... \sigma_n$ such that $\sigma_i \in \Sigma$, for every i such that $1 \leq i \leq n$. If $|\sigma| = 0$ then $\sigma = \varepsilon$. The concatenation of two words α and β of Σ^* is denoted simply as $\alpha\beta$.

Given $\mathcal{L}$, a *fuzzy language* is any mapping from Σ^* into L. Fuzzy languages can also be treated as formal power series over Σ^* and $\mathcal{L}$ [33][34]. The set of all possible fuzzy languages on Σ^* and $\mathcal{L}$ is denoted by L^{Σ^*}.

Let ℓ be a fuzzy language. The *derivative* of ℓ by a word α is the fuzzy language $\alpha^{-1}\ell$ defined by $(\alpha^{-1}\ell)(\gamma) = \ell(\alpha\gamma)$ for any word γ. In addition, $\beta^{-1}(\alpha^{-1}\ell) = (\alpha\beta)^{-1}\ell$ for any words α and β.

A *fuzzy finite automaton* (FfA) over $\mathcal{L}$, or simply just a FfA, is a tuple $A = (Q, \Sigma, I, T, F)$ where Q is a finite nonempty set of states, Σ is an alphabet, $I \in L^Q$ is the *initial fuzzy state*, $F \in L^Q$ is the *final fuzzy state*, and $T : \Sigma \to L^{Q \times Q}$ defines a fuzzy transition on Q for each symbol of the alphabet. The extension of $T : \Sigma \to L^{Q \times Q}$ to $T^* : \Sigma^* \to L^{Q \times Q}$ is defined as follows:

$$\begin{array}{l} \text{(i) } T^*(\varepsilon) = E_Q, \text{ and} \\ \text{(ii) } T^*(\alpha\sigma) = T^*(\alpha) \circ T(\sigma) \text{ for any } \alpha \in \Sigma^*, \sigma \in \Sigma \end{array} \tag{3}$$

To simplify notation, T^* is also denoted by T. By associativity, $T(\alpha\beta) = T(\alpha) \circ T(\beta)$ for any two words α and β. The fuzzy language recognized by A, denoted by $[A]$, is defined by (4). For any word α,

$$[A](\alpha) = I \circ T(\alpha) \circ F = \bigvee_{p,q \in Q} I(p) \otimes T(\alpha)(p, q) \otimes F(q) \tag{4}$$

Let us observe that as $\mathcal{L}^*$ is locally finite and Q is a finite set, the set $\{I \circ T(\alpha) | \alpha \in \Sigma^*\}$ is also finite. This fact implies that $[A]$ is a fuzzy language of finite image [17].

The size of a FfA A, denoted by $|A|$, is the cardinality of Q. In addition, two FAs, A and A', are (*language*) *equivalent* if and only if $[A] = [A']$.

C. Crisp co-accessible co-deterministic FfA

A FfA $A = (Q, \Sigma, I, T, F)$ is a crisp co-accessible co-deterministic FfA (cCFfA) if it satisfies the following conditions:

- (*unique final state*) $F = \{e/1\}$, i.e., it has a unique final state with membership value 1.
- (*crisp transitions*) $T(\sigma)(p, q) \in \{0, 1\}$ for any $p, q \in Q$ and $\sigma \in \Sigma$.
- (*complete backward*) For any $q \in Q$ and $\sigma \in \Sigma$ there is a state p such that $T(\sigma)(p, q) = 1$.
- (*co-deterministic*) For any $\sigma \in \Sigma$ and $p, p', q \in Q$, if $T(\sigma)(p, q) = 1$ and $T(\sigma)(p', q) = 1$ then $p = p'$.
- (*co-accessible*) For every state $p \in Q$ there is a word β such that $T(\beta)(p, e) = 1$.

Let $A = (Q, \Sigma, I, T, \{e/1\})$ be a cCFfA. If two states p and p' satisfy that $T(\beta)(p, e) = 1$ and $T(\beta)(p', e) = 1$ for the same word β then $p = p'$. This is because A is co-deterministic. Since A is complete and co-deterministic, the composition $T(\beta) \circ F = T(\beta) \circ \{e/1\}$ turns out a singleton crisp state $\{p/1\}$ where p is the unique state such that $T(\beta)(p, e) = 1$. This state is denoted by $e_{\overline{\beta}}$ (when A is clear in the context). In fact, the set of states Q is the finite set $\{e_{\overline{\beta}} \,|\, \beta \in \Sigma^*\}$ because A is co-accessible. This discussion allows us to calculate the fuzzy language recognized by a cCFfA A. Let us observe that for any fuzzy state $S \in L^Q$ and word β

$$S \circ T(\beta) \circ \{e/1\} = S(e_{\overline{\beta}}) \tag{5}$$

Thus, $[A](\beta) = I \circ T(\beta) \circ \{e/1\} = I(e_{\overline{\beta}})$ for any word β. In this paper, we denote by A_c a cCFfA equivalent to a FfA A.

III. Fuzzy Deterministic Finite Automata

Let $A = (Q, \Sigma, I, T, F)$ be a FfA. Then A is said to be a *fuzzy deterministic finite automaton* (FDfA) if it satisfies the following properties:

- A has a *unique initial state*, that is, the initial fuzzy state can be represented as $I = \{u/I(u)\}$.
- A is *complete*, that is, for any $\sigma \in \Sigma$ and $p \in Q$, there exists $q \in Q$ such that $T(\sigma)(p, q) > 0$.

- A is *deterministic*, that is, for any $\sigma \in \Sigma$ and p, q, $q' \in Q$, if $T(\sigma)(p,q) > 0$ and $T(\sigma)(p,q') > 0$ then $q = q'$.

In other words, a FDfA is a *fuzzyfication* of an ordinary DfA. For a FDfA $A = (Q, \Sigma, \{u/I(u)\}, T, F)$, the fuzzy transition relation T induces the function $\delta_T : Q \times \Sigma \to Q$ in the following way: $\delta_T(p,\sigma) = q \Leftrightarrow T(\sigma)(p,q) > 0$ for any $\sigma \in \Sigma$ and p, $q \in Q$.

Furthermore, we define the extension $\delta_T^* : Q \times \Sigma^* \to Q$ by $\delta_T^*(p,\varepsilon) = p$ and $\delta_T^*(p,\alpha\sigma) = \delta(\delta_T^*(p,\alpha),\sigma)$ for any $\sigma \in \Sigma$, $\alpha \in \Sigma^*$ and $p \in Q$. The state $\delta_T^*(p,\alpha)$ represents the unique *reachable* state from p via the word α, denoted simply by p_α. Let us observe that every state p is reachable by the empty word ε, that is, $p = p_\varepsilon$. By using previous notation, values of transitions between states are calculated as follows

$$\begin{array}{l} T(\varepsilon)(p,p_\varepsilon) = 1 \\ T(\alpha\sigma)(p,p_{\alpha\sigma}) = T(\alpha)(p,p_\alpha) \otimes T(\sigma)(p_\alpha,p_{\alpha\sigma}) \\ T(\alpha)(p,q) = 0 \text{ otherwise} \end{array} \quad (6)$$

with $\alpha \in \Sigma^*$ and $\sigma \in \Sigma$. By associativity, $T(\alpha\beta)(p,p_{\alpha\beta}) = T(\alpha)(p,p_\alpha) \otimes T(\beta)(p_\alpha,p_{\alpha\beta})$ for any $p \in Q$ and words α and β. Let us observe that for the zero-divisor free condition in $\mathcal{L}$, for any $\alpha \in \Sigma^*$, $T(\alpha)(p,q) > 0 \Leftrightarrow q = p_\alpha$ holds.

Thus, a state $q \in Q$ is said *accessible* if there is a word α such that $q = u_\alpha$, i.e., q is reachable from the initial state u by the word α. By (6) and (4), the fuzzy language recognized by a FDfA A is

$$[A](\alpha) = I(u) \otimes T(\alpha)(u,u_\alpha) \otimes F(u_\alpha) \quad (7)$$

for any $\alpha \in \Sigma^*$. Given a word α and an accessible state u_α, $[A_{u_\alpha}]$ is the fuzzy language defined by

$$[A_{u_\alpha}](\beta) = T(\beta)(u_\alpha,u_{\alpha\beta}) \otimes F(u_{\alpha\beta}) \quad (8)$$

for any word β. Thus, it is simple to prove that

$$\alpha^{-1}[A] = (I(u) \otimes T(\alpha)(u,u_\alpha)) \otimes [A_{u_\alpha}] \quad (9)$$

for any word α. In order to simplify the presentation of some equations, we denote the truth value of the path from the initial state u to the state u_α reached by a word α by:

$$w_A(\alpha) = I(u) \otimes T(\alpha)(u,u_\alpha) \quad (10)$$

We say that a FDfA A is *accessible* if the set of states Q of A is the finite set $\{u_\alpha | \alpha \in \Sigma^*\}$.

Let us note that, when a FDfA A satisfies $I(u) = 1$ and $T(\sigma)$ is a crisp relation on Q, for every $\sigma \in \Sigma$, then, equation (7) becomes $[A](\alpha) = F(u_\alpha)$. In this case, A is called *crisp-deterministic fuzzy finite automaton* (cDFfA) in the literature [13]. A cDFfA is just a particular case of a FDfA.

A. Conditions for a minimal FDfA

In this paper, we deal with the problem of finding a minimal FDfA equivalent to a given FfA. Unlike the case of ordinary DfAs, there may be several topological different minimal FDfAs that are equivalent to the same FfA. This is the reason to present the minimality conditions for FDfAs .

Definition *1:* A FDfA A is a *minimal* FDfA if $|A| \leq |A'|$ for any FDfA A' equivalent to A.

Property *1:* Let $A = (Q, \Sigma, \{u/I(u)\}, T, F)$ be a FDfA over $\mathcal{L}$. If A is minimal then A satisfies the next necessary conditions:

1) A is an *accessible* FDfA.
2) A is *observable*, i.e., for any $p, q \in Q$, $[A_p] = [A_q]$ implies $p = q$.

Previous necessary conditions are proved similarly to the given for ordinary DfAs [1]. However, these properties are not sufficient for characterizing a minimal FDfA. In other words, there may be FDfAs that are *accessible* and *observable*, but they are not minimal FDfAs. Next property is a sufficient condition for a FDfA to be a minimal one.

Property *2:* Let $A = (Q, \Sigma, \{u/I(u)\}, T, F)$ be an accessible FDfA over $\mathcal{L}$, then

$$\begin{array}{l} \neg(\exists \alpha, \beta \in \Sigma^*, \ell \in L^{\Sigma^*}, x, y \in L : \\ u_\alpha \neq u_\beta \text{ and } \alpha^{-1}[A] = x \otimes \ell \text{ and } \beta^{-1}[A] = y \otimes \ell) \\ \text{implies } A \text{ is a minimal FDfA} \end{array} \quad (11)$$

IV. Determinization of Fuzzy Finite Automata

The common method for determinization of a FfA $A = (Q, \Sigma, I, T, F)$ over $\mathcal{L}$ (locally finite) is via the construction of the Nerode automaton of A, $N(A)$. This is a well-known determinization method introduced by Ignjatović et al. [17]. $N(A)$ is a cDFfA equivalent to A whose finite set of states is $Q_{N(A)} = \{I_\alpha | \alpha \in \Sigma^*\}$ where $I_\alpha = I \circ T(\alpha)$ for any word α. A generalization of this construction, based on the notion of factorization of fuzzy states, was provided in [23]. We restate this construction in the following definition.

Definition *2:* For a FfA $A = (Q, \Sigma, I, T, F)$, the determinization of A via factorization (f, g) is the fuzzy automaton $D(A) = (Q_{D(A)}, \Sigma, I_{D(A)}, T_{D(A)}, F_{D(A)})$ defined as follows:
1) Set of states. $Q_{D(A)} = \{R_\alpha | \alpha \in \Sigma^*\}$ where

$$\begin{array}{l} \text{(i) } R_\varepsilon = f(I); \\ \text{(ii) } R_{\alpha\sigma} = f(R_\alpha \circ T(\sigma)), \alpha \in \Sigma^*, \sigma \in \Sigma. \end{array} \quad (12)$$

2) Initial fuzzy state. $I_{D(A)} = \{f(I)/g(I)\}$.
3) The fuzzy transition function $T_{D(A)} : \Sigma \to L^{Q_{D(A)} \times Q_{D(A)}}$ fulfills

$$\begin{array}{l} T_{D(A)}(\sigma)(P,S) = g(P \circ T(\sigma)) \text{ if } P = R_\alpha, S = R_{\alpha\sigma}; \\ T_{D(A)}(\sigma)(P,S) = 0 \text{ otherwise.} \end{array} \quad (13)$$

for any P, $S \in Q_{D(A)}$ and $\sigma \in \Sigma$.
4) Final fuzzy state. $F_{D(A)}(P) = P \circ F$, $P \in Q_{D(A)}$.

The automaton $D(A)$ provided in Definition 2 is an accessible fuzzy deterministic automaton[2] equivalent to A (see further details in [23][26]). By Definition 2, each state in $Q_{D(A)}$ is a fuzzy state of Q. As $D(A)$ is deterministic, then, $\delta^*_{T_{D(A)}}(R_\varepsilon,\alpha)$ is well-defined and R_α represents the accessible state from the initial state R_ε ($f(I)$) by the word α. In the rest of this paper, $D(A)$ denotes the fuzzy automaton obtained by means of Definition 2. The next lemma establishes the main properties of $D(A)$.

[2]Formally, $D(A)$ may be not a finite automaton. This fact does not invalidate the results provided in this Section since it is handled as a well-defined mathematical object.

Lemma *1:* Let $A = (Q, \Sigma, I, T, F)$ be a FfA. The automaton $D(A)$ satisfies the following properties for any $\alpha, \beta \in \Sigma^*$ and $\sigma \in \Sigma$:

1) $R_\alpha \circ T(\sigma) = T_{D(A)}(\sigma)(R_\alpha, R_{\alpha\sigma}) \otimes R_{\alpha\sigma}$
2) $I \circ T(\alpha) = w_{D(A)}(\alpha) \otimes R_\alpha$
3) $[A] = [D(A)]$ (*language equivalence*)
4) $[D(A)_{R_\alpha}](\beta) = R_\alpha \circ T(\beta) \circ F$

Let us assume that $D(A)$ is a finite automaton for a given FfA A (finiteness of $D(A)$ will be solved in Section V). The construction of $D(A)$ is based on the following algorithm [23] that we include here for the convenience of the reader.

Algorithm 1 Construction of $D(A)$

Input: A FfA $A = (Q, \Sigma, I, T, F)$
Output: $D(A) = (Q_{D(A)}, \Sigma, I_{D(A)}, T_{D(A)}, F_{D(A)})$
1: $T_{D(A)}(.)(.,.) \leftarrow 0$
2: $I_{D(A)} \leftarrow \{f(I)/g(I)\}$
3: $Q_{D(A)} \leftarrow \{f(I)\}$
4: Initialize an empty queue $NewSt$ of fuzzy states
5: **Enqueue**$(NewSt, f(I))$
6: **while** $NewSt \neq \emptyset$ **do**
7: $P \leftarrow$ **Dequeue**$(NewSt)$
8: $F_{D(A)}(P) \leftarrow P \circ F$
9: **for all** $\sigma \in \Sigma$ **do**
10: $S \leftarrow f(P \circ T(\sigma))$
11: $T_{D(A)}(\sigma)(P, S) \leftarrow g(P \circ T(\sigma))$
12: **if** $S \notin Q_{D(A)}$ **then**
13: $Q_{D(A)} := Q_{D(A)} \cup \{S\}$
14: **Enqueue**$(NewSt, S)$
15: **end if**
16: **end for**
17: **end while**
18: **return** $D(A)$

Algorithm 1 is a straightforward implementation of Definition 2 for a FfA A. In each step of the algorithm, we take a state that has not been processed yet (ln. 7). For each symbol σ, we generate (via factorization (f, g)) its successor (ln. 10) and add a transition marked σ from the state to its successor (ln. 11). If a follower represents a newly generated state (i.e., it is not equal to a previously generated state), then we add that state to the set of states of $Q_{D(A)}$ and the queue of unprocessed states ($NewSt$) (lns. 13 and 14 resp.). After we have created all successors for a given state P, this state will no longer participate in the computation since it has been removed from the queue (ln. 7). Initially, the queue of unprocessed states and the set of states of $Q_{D(A)}$ contains $f(I)$ (the unique initial state). The final fuzzy state is progressively calculated in line 8 by computing $F_{D(A)}(P)$ for each state. Algorithm 1 ends when the set of states $Q_{D(A)}$ is obtained, or equivalently, when there are no more unprocessed states (ln. 6). Thus, the algorithm is correct to output the accessible FDfA $D(A)$. In addition, as each state in $Q_{D(A)}$ is a fuzzy state of Q, it is simple to prove that the computation time of Algorithm 1 is proportional to $r(|Q| + m(2|Q| + |Q|^2 + r|Q|))$ where $r = |Q_{D(A)}|$ and $m = |\Sigma|$. In the case that $|Q_{D(A)}| = |Q|$, its asymptotic complexity is $\mathcal{O}(m|Q|^3)$.

V. Determinization of a cCFfA: Main results

Let $A_c = (Q, \Sigma, I, T, \{e/1\})$ be a cCFfA equivalent to the FfA A. In this section, we prove that $D(A_c)$ is a FDfA equivalent to A that satisfies the minimality necessary conditions, i.e., $D(A_c)$ is an accessible and observable FDfA. We start with two fundamental properties to achieve such goal.

Lemma *2:* Let $A_c = (Q, \Sigma, I, T, \{e/1\})$ be a cCFfA. For any fuzzy state $S \in L^Q$ and a symbol $\sigma \in \Sigma$,

$$f(f(S) \circ T(\sigma)) = f(S \circ T(\sigma)) \tag{14}$$

By a simple induction on the length of a word α and using (14) and Definition 2, it is relatively simple to prove (15.a) in the next lemma. In addition, (15.b) is derived directly by Lemma 1.2 and (15.a).

Lemma *3:* Let $A_c = (Q, \Sigma, I, T, \{e/1\})$ be a cCFfA. Then, $D(A_c)$ fulfills that, for each word $\alpha \in \Sigma^*$:

$$\begin{array}{ll} \text{(a) } f(I_\alpha) = R_\alpha \text{ where } I_\alpha = I \circ T(\alpha); \\ \text{(b) } f(w_{D(A_c)}(\alpha) \otimes R_\alpha) = R_\alpha \end{array} \tag{15}$$

The importance of Lemma 3 lies in the fact that if the factorization $f(.)$ had been maximal[3], then eq. (15.a) would have been obtained directly by using Lemma 1.2. This is not the case for the considered lattices, since they do not admit maximal factorizations (cf. [27]). However, under the conditions given in Lemma 3, the factorization behaves as a maximal factorization for the values of the paths that reach accessible states.

Theorem *1:* Let $A_c = (Q, \Sigma, I, T, \{e/1\})$ be a cCFfA equivalent to a FfA A. Then, $D(A_c)$ is an accessible and observable FDfA equivalent to A. Furthermore, $|D(A_c)| \leq |N(A_c)|$.

Proof: The automaton $D(A_c)$ is an accessible fuzzy deterministic automaton equivalent to A_c (see Section IV). As A_c is equivalent to A, then $D(A_c)$ is also equivalent to A. We prove that $D(A_c)$ is a finite automaton. Let $Q_{N(A_c)} = \{I_\alpha | \alpha \in \Sigma^*\}$ where $I_\alpha = I \circ T(\alpha) \in L^Q$ for any word α. We recall that $Q_{N(A_c)}$ is the set of states of the Nerode Automaton of A_c. We know that this set is finite because $\mathcal{L}$ is locally finite [17]. Therefore, the set $\{f(I_\alpha) | \alpha \in \Sigma^*\}$ is a finite set because $f(.)$ is a well defined function. By (15) in Lemma 3, $f(I_\alpha) = R_\alpha$. In conclusion, by Definition 2, the set of states $Q_{D(A_c)} = \{R_\alpha | \alpha \in \Sigma^*\}$ is finite. This fact also implies that $|D(A_c)| \leq |N(A_c)|$ is true. Therefore, $D(A_c)$ is an accessible FDfA equivalent to A.

Now, we prove that $D(A_c)$ is observable. $D(A_c)$ is an accessible FDfA. Let us consider two different words α and β and the accessible states R_α and R_β. Let us assume that $R_\alpha \neq R_\beta$ but $[D(A_c)_{R_\alpha}] = [D(A_c)_{R_\beta}]$. By Lemma 1.4, for each word γ, $[D(A_c)_{R_\alpha}](\gamma) = R_\alpha \circ T(\gamma) \circ \{e/1\} = R_\beta \circ T(\gamma) \circ \{e/1\} = [D(A_c)_{R_\beta}](\gamma)$. Then, by (5), $R_\alpha(e_{\overline{\gamma}}) = R_\beta(e_{\overline{\gamma}})$. Since, A_c is a cCFfA, $Q = \{e_{\overline{\gamma}} | \gamma \in \Sigma^*\}$; in addition, $R_\alpha, R_\beta \in L^Q$. Therefore, $R_\alpha = R_\beta$. This is a contradiction with the initial assumption. In conclusion, $D(A_c)$ is observable. The proof of the Theorem concludes. ■

A. Construction of $D(A_c)$ and discussion

Let $A = (Q_A, \Sigma, I_A, T_A, F_A)$ be a FfA over $\mathcal{L}$ (locally finite). The construction $r(N(r(A))$ by using automata reversal

[3]A maximal factorization satisfies that $f(S) = f(x \otimes S)$ with $x \neq 0$ for any fuzzy state S.

operation, $r(.)$, and the Nerode automaton, $N(.)$ [17], returns a cCFfA A_c equivalent to A as it was proved in [22][21]. This automaton has at most k^n states where n is the size of A and k is a number indicating the different values in the finite subsemiring induced by the finite set $\{I_A(q),\ F_A(q),\ T_A(\sigma)(p,q)|p,q \in Q_A,\ \sigma \in \Sigma\}$. The worst case computation time of this construction is $\mathcal{O}(mnk^{2n})$ for an alphabet with m symbols following the argumentation given in [20]. By the result in [21], the automaton $N(A_c)$ is the minimal cDFfA equivalent to A. Thus, $N(A_c)$ has at most k^n states and its worst case computation time is $\mathcal{O}(mk^{3n})$ [22][21].

Let $A_c = (Q, \Sigma, I, T, F)$ be the cCFfA $r(N(r(A))$ equivalent to A. In our case, the construction of $D(A_c)$ is based on the Algorithm 1. By Theorem 1 and the discussion above, $|D(A_c)| \leq |N(A_c)| \leq k^n$ holds. As each state in $Q_{D(A_c)}$ is a fuzzy state of Q ($|Q| \leq k^n$), then the worst computation time of Algorithm 1 is $\mathcal{O}(m|Q|^3)$; i.e., its worst case complexity is $\mathcal{O}(mk^{3n})$.

The relevance of Theorem 1 is due to the following facts: (a) it proves that $D(A_c)$ is an accessible and observable FDfA equivalent to A whose size is less than or equal to the size of the minimal cDFfA equivalent to A; and (b) the worst case time complexity to build $D(A_c)$ is less than or equal to the worst case time complexities obtained by current algorithms in the literature to build a minimal cDFfA.

Example *1:* In this example we show how to build the automaton $D(A_c)$ using Algorithm 1 presented in Section IV. The complete residuated lattice used in the example is $\mathcal{L} = ([0,1], max, min, \otimes, \rightarrow, 0, 1)$. $\mathcal{L}$ is based on a continuous t-norm in $[0,1]$ which is defined by

$$x \otimes y = \begin{cases} max(0.3, x+y-1) & \text{if } (x,y) \in [0.3,1]^2 \\ min(x,y) & \text{otherwise} \end{cases} \quad (16)$$

and whose residuum is

$$x \rightarrow y = \begin{cases} 1 & \text{if } x \leq y \\ 1-x+y & \text{if } 0.3 < y < x \leq 0.7 \\ y & \text{otherwise} \end{cases} \quad (17)$$

We consider that the cCFfA $A_c = (Q, \Sigma, I, T, F)$ has been previously constructed. In the example, the alphabet is $\Sigma = \{a, b, c, d\}$. The set of states Q has nine states numbered as q_0 to q_8. The initial fuzzy state is defined by the row $I = (1, 0.9, 0.4, 0.7, 0, 0.4, 0.3, 0.5, 0)$. The final fuzzy state is defined by the column $F = (1, 0, 0, 0, 0, 0, 0, 0, 0)^{\mathrm{T}}$. Each matrix defining the transitions for each symbol are long, so we only show here the transitions having value 1 (the rest of transitions have value 0):
For $T(a)$, (q_1, q_0), (q_1, q_1), (q_5, q_4), (q_5, q_5), (q_8, q_2), (q_8, q_3), (q_8, q_6), (q_8, q_7), (q_8, q_8). For $T(b)$, (q_2, q_0), (q_2, q_1), (q_6, q_4), (q_6, q_5), (q_8, q_2), (q_8, q_3), (q_8, q_6), (q_8, q_7), (q_8, q_8). For $T(c)$, (q_3, q_0), (q_3, q_1), (q_7, q_4), (q_7, q_5), (q_8, q_2), (q_8, q_3), (q_8, q_6), (q_8, q_7), (q_8, q_8). For $T(d)$, (q_4, q_0), (q_8, q_1), (q_8, q_2), (q_8, q_3), (q_8, q_4), (q_8, q_5), (q_8, q_6), (q_8, q_7), (q_8, q_8).

The cCFfA A_c is depicted in Figure 1. In this figure, we have not depicted the state q_8 and their transitions to simplify the graph.

Using Algorithm 1 we can build the automaton $D(A_c)$ as follows: $R_\varepsilon = f(I) = (1, 0.9, 0.4, 0.7, 0, 0.4, 0.3, 0.5, 0)$,

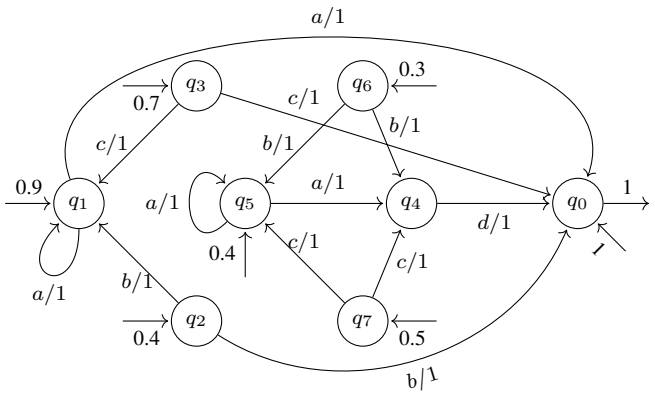

Fig. 1. The cCFfA A_c for the Example 1.

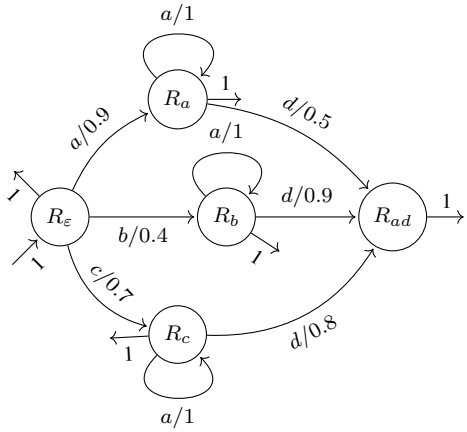

Fig. 2. The accessible and observable FDfA $D(A_c)$ equivalent to A_c.

where $g(I) = 1$. This is the unique initial state of $D(A_c)$. Starting with this initial state we have,

$R_a = f(R_\varepsilon \circ T(a)) = f((0.9, 0.9, 0, 0, 0.4, 0.4, 0, 0, 0)) = (1, 1, 0, 0, 0.5, 0.5, 0, 0, 0)$ with $g(R_\varepsilon \circ T(a)) = 0.9$;

$R_b = f(R_\varepsilon \circ T(b)) = f((0.4, 0.4, 0, 0, 0.3, 0.3, 0, 0, 0)) = (1, 1, 0, 0, 0.9, 0.9, 0, 0, 0)$ with $g(R_\varepsilon \circ T(b)) = 0.4$;

$R_c = f(R_\varepsilon \circ T(c)) = f((0.7, 0.7, 0, 0, 0.5, 0.5, 0, 0, 0)) = (1, 1, 0, 0, 0.8, 0, 8, 0, 0, 0)$ with $g(R_\varepsilon \circ T(c)) = 0.7$;

$R_d = f(R_\varepsilon \circ T(d)) = f((0, 0, 0, 0, 0, 0, 0, 0, 0)) = \mathbf{0}$ with $g(R_\varepsilon \circ T(d)) = g(\mathbf{0}) = 1$;

$R_{ad} = f(R_a \circ T(d)) = f((0.5, 0, 0, 0, 0, 0, 0, 0, 0)) = (1, 0, 0, 0, 0, 0, 0, 0, 0)$ with $g(R_a \circ T(d)) = 0.5$.

By following this procedure, it is simple to obtain the states, $R_{aa} = f(R_a \circ T(a)) = R_a$ with $g(R_a \circ T(a)) = 1$; $R_{ba} = f(R_b \circ T(a)) = R_b$ with $g(R_b \circ T(a)) = 1$; $R_{bd} = f(R_b \circ T(d)) = R_{ad}$ with $g(R_b \circ T(d)) = 0.9$; $R_{ca} = f(R_c \circ T(a)) = R_c$ with $g(R_c \circ T(a)) = 1$; and $R_{cd} = f(R_c \circ T(d)) = R_{ad}$ with $g(R_c \circ T(d)) = 0.8$. The rest of transitions falls in the state $\mathbf{0}$ with value 1.

It is also simple to observe that $F_{D(A_c)}(R.) = 1$ for all the states previously indicated exception the state $\mathbf{0}$ whose value is 0. The automaton $D(A_c)$ is depicted in Figure 2. In this figure, we have not depicted the state $\mathbf{0}$ and their transitions to simplify the graph.

Both automata $D(A_c)$ and A_c recognize the fuzzy language $\{\varepsilon/1,\ a^+/0.9,\ ba^*/0.4,\ ca^*/0.7,\ a^+d/0.4,\ ba^*d/0.3,\ ca^*d/0.5\}$. This example has been selected because the minimal cDFfA $N(A_c)$ obtained by the methods provided in [21] or [22] has the same size as $D(A_c)$. However, the automaton $D(A_c)$ can be modified to obtain a minimal FDfA with less size than $N(A_c)$ by the method we provide in the next Section.

VI. Towards a Minimal FDfA by means of $D(A_c)$

In this section, we consider that $A_c = (Q, \Sigma, I, T, \{e/1\})$ is a cCFfA equivalent to a FfA A and $D(A_c) = (Q_{D(A_c)}, \Sigma, I_{D(A_c)}, T_{D(A_c)}, F_{D(A_c)})$ is the accessible and observable FDfA (Theorem 1) obtained by the determinization of A_c via factorization (f, g). We recall that $Q_{D(A_c)} = \{R_\alpha | \alpha \in \Sigma^*\}$ and $R_\varepsilon = f(I)$ is the unique initial state of $D(A_c)$ with membership value $g(I)$ (see Definition 2).

Firstly, we introduce the concept of *reduction* of two states of $D(A_c)$. This concept is new and distinct from the well-established notion of bisimulation relation between two states of the observed automaton. Namely, the *reduction* relation in the set of states $Q_{D(A_c)}$ is a strict order relation, while the bisimulation relation is usually a (fuzzy) preorder or a (fuzzy) equivalence (cf. [36][37] for more details). This property allows us to construct a graph which we call the *Reduction Graph* of $D(A_c)$. We use this graph to characterize the minimality of the $D(A_c)$: if the Reduction Graph is empty (contains no reduction relations), then $D(A_c)$ is a minimal FDfA. If the Reduction Graph is not empty, the information it contains can be used to construct an equivalent minimal FDfA. Furthermore, if there are multiple equivalent minimal FDfAs with different topologies, the graph contains the necessary information to generate all of them.

A. Reduction Graph of $D(A_c)$

For each (accessible) state $S \in Q_{D(A_c)}$, we define the supremum value of all words α that reach S from the initial state R_ε, i.e., $S = R_\alpha$,

$$W_{D(A_c)}(S) = \bigvee_{\{\alpha \in \Sigma^* : S = R_\alpha\}} w_{D(A_c)}(\alpha) \tag{18}$$

The value $W_{D(A_c)}(S)$ is well-defined because (a) L is totally ordered; (b) $\otimes$ is monotone; and (c) $D(A_c)$ is a finite automaton. These conditions also assure that there is a word α_S such that $S = R_{\alpha_S}$ with $W_{D(A_c)}(S) = w_{D(A_c)}(\alpha_S)$.

Definition *3:* Given the automaton $D(A_c)$ and two states $P,\ S \in Q_{D(A_c)}$, we say that P *reduces* S (in short notation $P \ll S$) if and only if

$$\text{(a) } P \neq S \text{ and (b) } W_{D(A_c)}(S) \otimes S = W_{D(A_c)}(S) \otimes P \tag{19}$$

The intuition behind P reduces S lies in the observation that every word that reaches S in $D(A_c)$ could alternative reach P without modifying the language recognized by $D(A_c)$. This fact will be formally stated in the proof of minimality in Theorem 3. The particular properties of $D(A_c)$ obtained in Section V allow us to establish the following necessary conditions when $P \ll S$ occurs in $D(A_c)$.

Lemma *4:* Given the automaton $D(A_c)$, if $P \ll S$ for two states $P,\ S \in Q_{D(A_c)}$ then

$$\begin{array}{l}\text{(a) } \mathrm{Supp}(S) = \mathrm{Supp}(P);\ \text{(b) } S \geq P > \mathbf{0} \\ \text{(c) } W_{D(A_c)}(P) > W_{D(A_c)}(S);\ \text{and (d) } S \neq R_\varepsilon\end{array} \tag{20}$$

Let us observe that in a relation P reduces S, P can not be a crisp state by (20.b). By the condition (20.c) in Lemma 4, it is clear that the order relation $\ll$ on $Q_{D(A_c)}$ is *irreflexive* and *asymmetric* We also prove that $\ll$ is a transitive relation.

Lemma *5:* Given the automaton $D(A_c)$, the pair $(Q_{D(A_c)}, \ll)$ is a strict partial ordered set.

The strict partial order $\ll$ on $Q_{D(A_c)}$ allows us to depict a directed graph whose nodes are the states of $Q_{D(A_c)}$ and each directed arc (P, S) in the graph represents the relation $P \ll S$. This acyclic directed graph is called the *Reduction Graph of* $D(A_c)$, denoted $\mathcal{G}(D(A_c))$. In this paper, we say that $\mathcal{G}(D(A_c))$ is *empty* if there is no arc in the graph.

B. Characterization of minimality via $\mathcal{G}(D(A_c))$

The following theorem establishes the necessary condition for the assumption that a $D(A_c)$ is *not* a minimal FDfA.

Theorem *2:* Let A be a FfA, and let A_c be a cCFfA equivalent to A. Given the automaton $D(A_c)$, if $D(A_c)$ is not a minimal FDfA then there are (at least) two states $P,\ S \in Q_{D(A_c)}$ such that $P \ll S$, i.e., the Reduction Graph $\mathcal{G}(D(A_c))$ is not empty.

The following theorem establishes the necessary condition for the assumption that a $D(A_c)$ is a minimal FDfA. The proof is made in a constructive way and describes the main idea of states reduction which allows us to get a minimal FDfA from the automaton $D(A_c)$.

Theorem *3:* Let A be a FfA. Let A_c be a cCFfA equivalent to A. Given the automaton $D(A_c)$, if $D(A_c)$ is a minimal FDfA then the Reduction Graph $\mathcal{G}(D(A_c))$ is empty, i.e., there is no reduction relation $P \ll S$ for any $P, S \in Q_{D(A_c)}$.

Both theorems can be summarized in the following result.

Corollary *1:* Let A be a FfA. Let A_c be a cCFfA equivalent to A. Then $D(A_c)$ is a minimal FDfA if and only if the Reduction Graph $\mathcal{G}(D(A_c))$ is empty.

Let us observe that if the construction $D(A_c)$ is not a minimal FDfA, the structure of the graph $\mathcal{G}(D(A_c))$ represents all the ways that minimal FDfAs equivalent to A may be constructed from the automaton $D(A_c)$. We recall that a reduction arc $P \ll S$ is obtained via the Pigeonhole Principle with respect to the states of a minimal FDfA equivalent to A (Theorem 2). Thus, each isolated node (a state of $Q_{D(A_c)}$) in $\mathcal{G}(D(A_c))$ is naturally associated to a unique state of this minimal FDfA. In addition, each node in $\mathcal{G}(D(A_c))$ which is the left extreme of a path in the graph, is also associated to a unique state in the minimal FDfA. A node that is a left extreme of a path is a reducer state of all the rest of nodes (states) for its path. Therefore, the number of isolated nodes plus the number of left extremes is the size of any minimal FDfA equivalent to A. We illustrate these facts in the following example before presenting the required algorithms for the minimization procedure.

Example *2:* We show in this example how to built the Reduction Graph $\mathcal{G}(D(A_c))$ for the automaton $D(A_c)$ obtained

TABLE I

S	$(q_0, ..., q_8)$	$W_{D(A_c)}(S)$
R_ε	(1, 0.9, 0.4, 0.7, 0, 0.4, 0.3, 0.5, 0)	1
R_a	(1, 1, 0, 0, 0.5, 0.5, 0, 0, 0)	0.9
R_b	(1, 1, 0, 0, 0.9, 0.9, 0, 0, 0)	0.4
R_c	(1, 1, 0, 0, 0.8, 0,8, 0, 0, 0)	0.7
R_{ad}	(1, 0, 0, 0, 0, 0, 0, 0, 0)	0.5
$\mathbf{0}$	(0, 0, 0, 0, 0, 0, 0, 0, 0)	0.9

TABLE II

	R_ε	R_a	R_b	R_c	R_{ad}	$\mathbf{0}$
R_ε	-	-	-	-	-	-
R_a	-	-	$\ll$	-	-	-
R_b	-	-	-	-	-	-
R_c	-	-	$\ll$	-	-	-
R_{ad}	-	-	-	-	-	-
$\mathbf{0}$	-	-	-	-	-	-

in Example 1 (Figure 2). The graph $\mathcal{G}(D(A_c))$ is represented by its adjacency matrix. Let us recall that each arc represents a reduction relation between two states of $D(A_c)$. By following Definition 3, we need the definition of the (fuzzy) states of $D(A_c)$ and for each state S, the value $W_{D(A_c)}(S)$ (see (18)) which is the supremum value of the paths that reach S from the initial state. In Table I, we show the states obtained for the automaton $D(A_c)$ in Figure 2 (see Example 1) and for each state its value $W_{D(A_c)}(.)$.

In this simple example, the graph $\mathcal{G}(D(A_c))$ contains two arcs: $R_a \ll R_b$ and $R_c \ll R_b$. Indeed, by Definition 3, we have that $W_{D(A_c)}(R_b) \otimes R_b = 0.4 \otimes (1, 1, 0, 0, 0.9, 0.9, 0, 0, 0) = (0.4, 0.4, 0, 0, 0.3, 0.3, 0, 0, 0)$, and $W_{D(A_c)}(R_b) \otimes R_a = 0.4 \otimes (1, 1, 0, 0, 0.5, 0.5, 0, 0, 0) = (0.4, 0.4, 0, 0, 0.3, 0.3, 0, 0, 0)$. Recall that the operation to compute such values is the t-norm indicated in (16). Thus, R_a reduces R_b. Similarly, R_c reduces R_b. The adjacency matrix of $\mathcal{G}(D(A_c))$ is indicated in Table II.

By the characterization of minimality in this section, as the adjacency matrix contains arcs, then $D(A_c)$ in Figure 2 is not a minimal FDfA. In addition, by the comments indicated above this Example, the size of any minimal FDfA is 5. The state R_b may be reduced to R_a or to R_c. In this example, there are two possible minimal FDfAs equivalent to $D(A_c)$. In order to reduce the state R_b to R_a to get the first minimal FDfA, we simply move the transition ending in R_b in $D(A_c)$ to R_a in the new automaton. In this case the transition $(R_\varepsilon, b/0.4, R_b)$ in $D(A_c)$ is converted in the transition $(R_\varepsilon, b/0.4, R_a)$ in the minimal FDfA. This movement makes R_b an inaccessible state and it is removed. The result is illustrated in Figure 3. In Figure 4, we show the other case when R_b is reduced by R_c.

C. Algorithms for minimization procedure and complexity

The steps that we have followed in Example 2 to obtain a minimal FDfA have been: (1) to construct the automaton $D(A_c)$; (2) to calculate the values $W_{D(A_c)}(S)$ for each state S of $D(A_c)$; (3) to construct the adjacency matrix of the Reduction Graph $\mathcal{G}(D(A_c))$; and (4) to select the reduction relations from the adjacency matrix and to construct the minimal FDfA by transforming $D(A_c)$ via the movement of

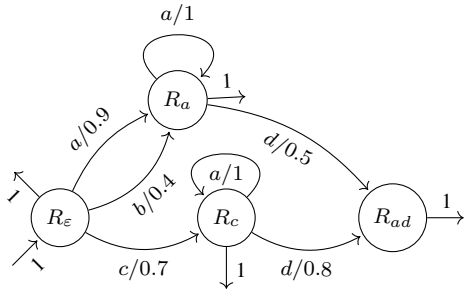

Fig. 3. The first option to obtain a minimal FDfA when R_a reduces R_b in $D(A_c)$ from Figure 2.

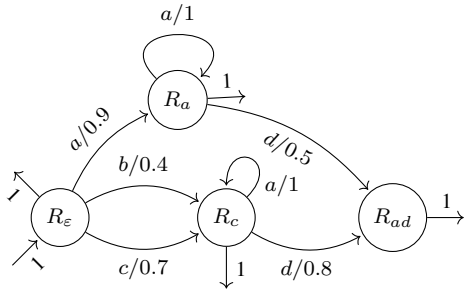

Fig. 4. The second option to obtain a minimal FDfA when R_c reduces R_b in $D(A_c)$ from Figure 2.

arcs defined in the selected reduction relations. In general, the Reduction Graph $\mathcal{G}(D(A_c))$ may have an arbitrary shape (but it is an acyclic graph), so it is necessary to provide a general way to obtain a minimal FDfA. The algorithms that we present below have such a purpose. In these algorithms, we use *associative array* Arr whose generic operations are:

- $\mathbf{Insert}(Arr, (key, value))$ that adds a new $(key, value)$ pair to Arr, mapping the key to its new value. Any existing mapping is overwritten;
- $\mathbf{Lookup}(Arr, key)$ that finds the value that is bound to a given key. The arguments to this operation are the associative array Arr and the key, while the value is returned from the operation.

Both operations have a worst-case time complexity linear with respect to the size of the associative array when it is implemented via a hash table.

Algorithm 1 can be easily modified so that it calculates both $D(A)$ and the values $W_{D(A)}(S)$ for each S in $D(A)$. In Algorithm 2, every time a transition is obtained (ln. 15), the new possible maximum value (ln. 16) is calculated in an auxiliary variable w. The new maximum value is updated on line 20 for a new state, and it is modified in line 23 for an existing state. Algorithm 2 is used with a cCFfA A_c as an input to obtain the automaton $D(A_c)$ and the associative array $W_{D(A_c)}$ which stores the values $W_{D(A_c)}(S)$ (see (18) for each (fuzzy) state $S \in Q_{D(A_c)}$.

After the construction of $D(A_c)$, we apply Algorithm 3. This algorithm does not explicitly construct the adjacency matrix of $\mathcal{G}(D(A_c))$ as we did in Example 2. By the explanations given before Example 2, what we need is to obtain the left

Algorithm 2 Construction of $D(A)$ and $W_{D(A)}$

Input: A FfA $A = (Q, \Sigma, I, T, F)$
Output: The FDfA $D(A) = (Q_{D(A)}, \Sigma, I_{D(A)}, T_{D(A)}, F_{D(A)})$ of A and the mapping $W_{D(A)} : Q_{D(A)} \to L$

1: $T_{D(A)}(.)(.,.) \leftarrow 0$
2: $I_{D(A)} \leftarrow \{f(I)/g(I)\}$
3: $Q_{D(A)} \leftarrow \{f(I)\}$
4: $W_{D(A)}(f(I)) \leftarrow g(I)$
5: Initialize an empty queue $NewSt$ of fuzzy states
6: $\mathbf{Enqueue}(NewSt, f(I))$
7: Initialize an empty associative array $W_{D(A)}$ whose keys are fuzzy states from the set $Q_{D(A)}$ and values are from the set L
8: $\mathbf{Insert}(W_{D(A)}, (f(I), g(I)))$
9: **while** $NewSt \neq \emptyset$ **do**
10: $P \leftarrow \mathbf{Dequeue}(NewSt)$
11: $w_P \leftarrow \mathbf{Lookup}(W_{D(A)}, P)$
12: $F_{D(A)}(P) \leftarrow P \circ F$
13: **for all** $\sigma \in \Sigma$ **do**
14: $S \leftarrow f(P \circ T(\sigma))$
15: $T_{D(A)}(\sigma)(P, S) \leftarrow g(P \circ T(\sigma))$
16: $w \leftarrow w_P \otimes g(P \circ T(\sigma))$
17: **if** $S \notin Q_{D(A)}$ **then**
18: $Q_{D(A)} := Q_{D(A)} \cup \{S\}$
19: $\mathbf{Enqueue}(NewSt, S)$
20: $\mathbf{Insert}(W_{D(A)}, (S, w))$
21: **else**
22: $w_S \leftarrow \mathbf{Lookup}(W_{D(A)}, S)$
23: $\mathbf{Insert}(W_{D(A)}, (S, w_S \vee w))$
24: **end if**
25: **end for**
26: **end while**
27: **return** $(D(A), W_{D(A)})$

Algorithm 3 Construction of the reduction sets of $D(A_c)$

Input: A cCFfA $A_c = (Q, \Sigma, I, T, \{e/1\})$
Output: The reduction graph of A_c

1: Apply Algorithm 2 with A_c as an input, and store the result in $(D(A_c), W_{D(A_c)})$, where $W_{D(A_c)}$ is an associative array storing values (18) for each fuzzy state from $Q_{D(A_c)}$ as a key.
2: Initialize an empty associative array Ir whose keys are fuzzy states from the set $Q_{D(A_c)}$ and values are initially empty sets
3: **for all** $(k_S, val_S) \in W_{D(A_c)}$ **do**
4: $(k_{reducer}, w_{reducer}) \leftarrow (k_S, 0)$
5: **for all** $(k_P, val_P) \in W_{D(A_c)}$ **do**
6: **if** $k_P \neq k_S$ **and** $val_S \otimes k_S = val_S \otimes k_P$ **then**
7: **if** $w_{reducer} < val_P$ **then**
8: $(k_{reducer}, w_{reducer}) \leftarrow (k_P, val_P)$
9: **end if**
10: **end if**
11: **end for**
12: **if** $k_{reducer} \neq k_S$ **then**
13: $\mathbf{Insert}(Ir, (k_{reducer}, val_{k_{reducer}} \cup \{k_S\}))$
14: **end if**
15: **end for**
16: **return** Ir

Algorithm 4 Reduction of $D(A_c)$

Input: A cCFfA $A_c = (Q, \Sigma, I, T, \{e/1\})$
Output: A minimal FDfA B equivalent to A_c

1: Apply Algorithm 2 and Algorithm 3 with A_c as an input. Let $D(A_c)$ be the resulting FDfA and Ir the associative array with each fuzzy state from $Q_{D(A_c)}$ as a key, and the set of fuzzy states it reduces as a value, denoted $(P, R[P])$
2: $B \leftarrow D(A_c)$, define B to be exactly equal to $D(A_c)$
3: For each non empty tuple $(P, R[P])$ in Ir and $S \in R[P]$:
4: **for all** $T_D(\sigma)(S', S) > 0$ in $D(A_c)$ **do**
5: $T_B(\sigma)(S', P) \leftarrow T_D(\sigma)(S', S)$;
6: $T_B(\sigma)(S', S) \leftarrow 0$;
7: **end for**
8: Remove all inaccessible state from B
9: **return** B

extremes of each path of reduction. Each left extreme is an irreducible state under $\ll$ where the rest of states in its path can be reduced to it. Thus, Algorithm 3 returns in the associative array Ir, tuples of the form $(P, R[P])$ where P is a *reducer* (fuzzy) state from $Q_{D(A_c)}$ and $R[P] \subseteq Q_{D(A_c)}$ is a set of states such that each $S \in R[P]$ satisfies $P \ll S$ (S is reducible to P). In addition, in Ir, two distinct tuples $(P, R[P])$ and $(P', R[P'])$ satisfy that their reduction sets are disjoint, i.e., $R[P] \cap R[P'] = \emptyset$. Finally, with the information provided by Algorithm 3 and the automaton $D(A_c)$, we can construct the minimal FDfA equivalent to A_c (and to A) following the procedure indicated in the proof of Theorem 3.

Let us recall that A_c is the cCFfA equivalent to A. The size $|A_c|$ is bounded by k^n (see Subsection V-A). Algorithm 2 computes $D(A_c)$ and $W_{D(A_c)}$ (when its input is A_c) with a worst case time complexity $\mathcal{O}(mk^{3n})$ since associative array operations are linear, the size $|D(A_c)|$ is also bounded by k^n, and states of $D(A_c)$ are fuzzy states of the states of A_c. Algorithm 3 has a computational time proportional to $|A_c| \times |D(A_c)|^2$ (lines 3, 5 and 6), i.e, its worst case time complexity is $\mathcal{O}(k^{3n})$. The computational time of Algorithm 4 depends on the number of reducible pairs of states. For each reducible state S, in line 4 is calculated the set of states with transitions to it. This requires going through the m symbols of the alphabet with respect to the set of states in $D(A_c)$. In line 8, the transitions of the modified automaton is traversed to eliminate inaccessible states. This yields to a computational time proportional to $m \times (|D(A_c)| + |D(A_c)|^2)$, i.e., its worst case time complexity is $\mathcal{O}(mk^{2n})$. In conclusion, the worst case time complexity of the minimization algorithm is $\mathcal{O}((m+1)k^{3n})$. In Table III, we show a comparison with those methods that result in a minimal FDfA or a minimal cDFfA.

TABLE III
WORST CASE TIME COMPLEXITY OF SOME MINIMIZATION ALGORITHMS

Reference	Complexity	Output
[28]	$\mathcal{O}(k^{mr^2} m^{k^{(n+r)}} r^2)$	minimal FA over $(L, \vee, \wedge)$
[29]	$\mathcal{O}(mk^{3n} + 2^{2n} k^{2n})$	minimal FDfA over $(L, \vee, \wedge)$
[21]	$\mathcal{O}(mk^{3n})$	minimal cDFfA over $\mathcal{L}$
[22]	$\mathcal{O}(mk^{3n})$	minimal cDFfA over $\mathcal{L}$
this paper	$\mathcal{O}((m+1)k^{3n})$	minimal FDfA over $\mathcal{L}$

VII. CONCLUSIONS

In the literature, the problem of the conversion of a FfA to a minimal FDfA has been solved for FfAs over totally ordered lattices $(L, \vee, \wedge)$ [28][29]. The main problem of this minimal conversion is that several equivalent minimal FDfAs with different topologies may exist. In this paper, we provide a general algorithmic solution for a FfA over a complete residuated lattice that must be: (1) locally finite, (2) totally

ordered, (3) divisible, and (4) zero-divisor-free. Our algorithm does not exceed the computation complexity of the algorithms provided in [28][29]. This complexity is achieved by the determinization via factorization of the cCFfA equivalent to the input FfA. The determinization outputs an equivalent FDfA satisfying the necessary conditions for minimality, but the output may not be a minimal FDfA. To obtain a minimal FDfA, we study the sufficient condition for minimality that defines a strict partial order in the set of states. This allows us to introduce a new form of the state reduction different from the usual equivalence relation between states. In this way, we provide all the required algorithms to obtain the minimal FDfA. In future work, we will search for a solution without the condition of zero divisor free on the lattice, and to study the minimization for DFfAs to generalize the solution given in [35] for a DFfA over $(L, \vee, \wedge)$.

VIII. Acknowledgements

Dr. Stefan Stanimirović acknowledges the support of the Science Fund of the Republic of Serbia, GRANT No 7750185, Quantitative Automata Models: Fundamental Problems and Applications - QUAM, and the Ministry of Education, Science and Technological Development, Republic of Serbia, Contract No. 451-03-47/2023-01/200124.

We thank the editors and reviewers for their thoughtful comments and suggestions that helped to improve this paper. We also thank Prof. J. R. González de Mendívil for his ideas that gave rise to this paper.

References

[1] J. E. Hopcroft, R. Motwani, J. D. Ullman. *Introduction to Automata Theory*. 3rd Edition. Addison-Wesley, 2007.

[2] R. van Glabbeek, B. Ploeger. "Five Determinization algorithms". O.H Ibarra and B. Ravikumar (eds.), CIAA 2008, Lecture Notes in Computer Sicence 5148, pp. 161–170, 2008.

[3] J. A. Brzozowski. "Canonical regular expressions and minimal state graphs for definite events". In Proc. Sympos. Math. Theory of Automata (New York, 1962), Polytechnic Press of Polytechnic Inst. of Brooklyn, Brooklyn, N.Y., pp. 529–561, 1963.

[4] D. Qiu. "Supervisory control of fuzzy discrete event systems: A formal approach". IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, vol. 35, no. 1, pp. 72-88, 2005.

[5] D. Qiu and F. Liu. "Fuzzy discrete-event systems under fuzzy observability and a test algorithm". IEEE Transactions on Fuzzy Systems, vol.17, no. 3, pp. 578-589, 2009.

[6] W. Deng and D. Qiu. "Supervisory Control of Fuzzy Discrete Event Systems for Simulation Equivalence". IEEE Transactions on Fuzzy Systems, vol. 23, no. 1, pp.178-192, 2015.

[7] G. Bailador, G. Triviño. "Pattern recognition using temporal fuzzy automata". Fuzzy Sets and Systems, vol. 161, pp. 37–55, 2010.

[8] Y. Li, J. Wei. "Possibilistic fuzzy linear temporal logic and its model checking". IEEE Transactions on Fuzzy Systems, vol. 29, pp. 1899–1913, 2021.

[9] J. Mordeson, D. Malik. *Fuzzy Automata and Languages: Theory and Applications*. Chapman & Hall, CRC Press, London, Boca Raton, FL., 2002.

[10] G.G. Rigatos. "Fault detection and isolation based on fuzzy automata". Information Sciences, vol. 179 (12), pp. 1893–1902, 2009.

[11] Q. Wu, Z. Han, Q.E. Wu. "Application of fuzzy automata decision-making system in target control". Journal of Computer and Communications, vol. 05(10), pp. 16–25, 2017.

[12] J. R. Gonzalez de Mendivil. "A generalization of Myhill-Nerode Theorem". Fuzzy Sets and Systems, vol. 301, pp. 103–115, 2016.

[13] R. Bělohlávek. "Determinism and fuzzy automata". Information Sciences, vol. 143, pp. 205–209, 2002.

[14] D.W. Qiu. "Automata theory based on complete residuated lattice-valued logic". Science in China (Series F: Information Sciences), vol. 44(6), pp. 419-429, 2001.

[15] D.W. Qiu. "Automata theory based on complete residuated lattice-valued logic (II)". Science in China (Series F: Information Sciences), vol. 45(6), pp. 442–452, 2002.

[16] Y. M. Li, W. Pedrycz. "Fuzzy finite automata and fuzzy regular expressions with membership values in lattice ordered monoids". Fuzzy Sets and Systems, vol. 156, pp. 68–92, 2005.

[17] J. Ignjatović, M. Ćirić, S. Bogdanović. "Determinization of fuzzy automata with membership values in complete residuated lattices". Information Sciences, vol. 178, pp. 164–180, 2008.

[18] J. Ignjatović, M. Ćirić, S. Bogdanović, T. Petković. "Myhill-Nerode type theory for fuzzy languages and automata". Fuzzy Sets and Systems, vol. 161, pp. 1288–1324, 2010.

[19] Z. Jančić, J. Ignjatović, M. Ćirić. "An improved algorithm for determinization of weighted and fuzzy automata". Information Sciences, vol. 181, pp. 1358–1368, 2011.

[20] Z. Jančić, I. Micić, J. Ignjatović, M. Ćirić. "Further improvement of determinization methods for fuzzy finite automata". Fuzzy Sets and Systems, vol. 301, pp. 79–102, 2015.

[21] Z. Jančić, M. Ćirić. "Brzozowski type determinization for fuzzy automata". Fuzzy sets and Systems, vol 249, pp. 73–82, 2014.

[22] I. Micić, Z. Jančić, J. Ignjatović, M. Ćirić. "Determinization of fuzzy automata by means of degrees of language inclusion". IEEE Transactions on Fuzzy Systems, vol 23, no. 6, pp. 2144–2153, 2015.

[23] J. R. Gonzalez de Mendivil, J. R. Garitagoitia. "Determinization of fuzzy automata via factorization of fuzzy states". Information Sciences, vol 283, pp. 165–179, 2014.

[24] J. R. Gonzalez de Mendivil. "Conditions for Minimal Fuzzy Deterministic Finite Automata via Brzozowski's Procedure". IEEE Transactions on Fuzzy Systems, vol. 26, no. 4, 2018.

[25] S. Stanimirović. "Improved algorithms for determinization of fuzzy and weighted automata". Doctoral dissertation [In Serbian]. Faculty of Sciences and Mathematics, University of Niš. 2019.

[26] S. Stanimirović, M. Ćirić, J. Ignjatović. "Determinization of fuzzy automata by factorizations of fuzzy states and right invariant fuzzy quasi-orders". Information Sciences, vol. 469(12), pp. 79–100, 2018.

[27] S. Gerdjikov, J. R. Gonzalez de Mendivil. "Conditions for the existence of maximal factorizations". Fuzzy Sets and Systems, vol. 397, pp. 186–196, 2020.

[28] L. Li, D.W. Qiu. "On the state minimization of fuzzy automata". IEEE Transactions on Fuzzy Systems, vol. 23, no.2, pp. 434–443, 2015.

[29] J. R. Gonzalez de Mendivil, F. Fariña. "Canonization of max-min fuzzy automata". Fuzzy Sets and Systems, vol. 376, pp. 152–168, 2019.

[30] R. Bělohlávek. "Fuzzy Relational Systems: Foundations and Principles". Kluwer, New York, 2002.

[31] R. Bělohlávek, V. Vychodil. "Fuzzy Equational Logic". Studies in Fuzziness and Soft Computing, Springer, Berlin-Heidelberg, 2005.

[32] D. Qiu. "A note on Trillas' CHC models". Artificial Intelligence, vol. 171, no. 4, pp. 239–254, 2007.

[33] G. Rahonis. "Fuzzy languages". *Handbook of Weighted Automata*. M. Droste, W. Kuich, H. Vogler (Eds.), Springer-Verlag, Berlin, 2009.

[34] J. Ignjatović, M. Ćirić. "Formal power series and regular operations on fuzzy languages". Information Sciences, vol. 180, pp. 1104–1120, 2010.

[35] S. Halamish and O. Kupferman. "Minimizing deterministic lattice automata". ACM Transactions on Computational Logic (TOCL), vol. 16.1, pp. 1–21, 2015.

[36] I. Micić, Z. Jančić, and S. Stanimirović. "Computation of the greatest right and left invariant fuzzy quasi-orders and fuzzy equivalences". Fuzzy Sets and Systems, vol. 339, pp. 99–118, 2018.

[37] H. Wu, Y. Deng. "Logical characterizations of simulation and bisimulation for fuzzy transition systems". Fuzzy Sets and Systems, vol. 301, pp. 19–36, 2016.