University of Niš
Faculty of Sciences and Mathematics
Department of Computer Science

Ivana Z. Micić

BISIMULATIONS FOR FUZZY AUTOMATA

PhD thesis

Niš,2014

Univerzitet u Nišu
Prirodno-matematički fakultet
Departman za računarske nauke

Ivana Z. Micić

BISIMULACIJE ZA FAZI AUTOMATE

Doktorska disertacija

Niš,2014

Supervisor:	**Jelena Ignjatović**
	Associate Professor
	Faculty of Sciences and Mathematics
	University of Niš,Serbia
Members of	**Miroslav Ćirić**
the Commission:	Full Professor
	Faculty of Sciences and Mathematics
	University of Niš, Serbia
	Heiko Vogler
	Full Professor
	Faculty of Computer Science
	Technische Universität,Dresden,Germany
	Manfred Droste
	Full Professor
	Institut für Informatik
	Universität Leipzig, Germany
	Nada Damljanović
	Assistant Professor
	Faculty of Tehnical Sciences in Čačak
	University of Kragujevac, Serbia
Date of Defense	

Contents

Preface

The theory of fuzzy sets originated from the need to formalize the imprecise aspects of human knowledge and to present them by using mathematical models. It was initiated by L.A.Zadeh in 1965, who provided a way to distinguish the subtle nuances of the membership of an element to a particular class. As such problems are common in operation with classes of objects encountered in the real, physical world, fuzzy sets have a large field of application. Zadeh, also introduced the notion of fuzzy relations, in particular the notion of fuzzy equivalences and fuzzy quasi-orders. Due to the fact that fuzzy relations enable freedom in expressing barely noticeable traces of connectivity between elements, they are extensively applied in the modeling of various concepts in the so-called " soft " sciences such as psychology, linguistics and in many other scientific fields.

From the very beginning of the theory of fuzzy sets, fuzzy automata and languages are studied as a means for bridging the gap between the precision of computer languages and vagueness and imprecision, which are frequently encountered in the study of natural languages. Study of fuzzy automata and languages was initiated in 1960s by Santos [110, 111, 113], Wee [122], Wee and Fu [123], and Lee and Zadeh [73]. During the decades, fuzzy automata and languages have gained wide field of application including lexical analysis, description of natural and programming languages, learning systems, control systems, clinical monitoring, pattern recognition, error correction, neural networks, knowledge representation, databases, discrete event systems etc.

One of the most important problems of automata theory is to determine whether two given automata are equivalent, i.e. whether they have the same behavior. The behavior of an deterministic, nondeterministic or fuzzy automaton can be considered as the language or fuzzy language recognized by it. Namely, two automata are equivalent, or more precisely language-equivalent, if they recognize the same language. For deterministic finite automata the equivalence problem is solvable in polynomial time, but for nondeterministic and fuzzy finite automata it is computationally hard (PSPACEcomplete). Another important issue is to express the language-equivalence of two automata as a relation between their states, if such relationship exists, or find some kind of relations between states which would approximate the language-equivalence. The language-equivalence of two deterministic automata can be expressed in terms of relationships between their states, but in the case of nondeterministic and fuzzy automata the problem is more complicated, and we can only examine various approximations of the language-equivalence.

A widely-used concept for modeling "equivalence" between states of automata is that of bisimulation. Bisimulations have been introduced by Milner [81] and Park [89] in concurrency theory, and they were successfully exploited to model equivalence between various systems, as well as to reduce the number of states of these systems. Roughly, at the same time, they have also been discovered in some areas of mathematics, e.g., in modal logic and set theory. Nowedays, the bisimulations are employed in many areas of computer science, such as functional languages, object-oriented languages, domains, types, data types, compiler optimizations, databases, verification tools, program analysis, etc.

The main aim of this dissertation is to investigate bisimulations for fuzzy automata, with the special emphasis on the problem of finding the greatest bisimulation of appropriate systems. Besides, bisimulations will be considered as a means for approximating the language-equivalence, as well as for use in the state reduction of fuzzy automata.

In the first chapter we will introduce basic concepts of the theory of ordered sets and lattices. The special attention will be devoted to complete residuated lattices because we will observe them as the structures of truth values in the further work. Then we will introduce the basic notions of the fuzzy set theory and afterwards, main properties of fuzzy functions and fuzzy relations, especially uniform fuzzy relations. Main features of fuzzy equivalences and fuzzy quasi-orders will be presented at the end of this chapter.

In the second chapter of the dissertation fundamental concepts and important famous results of fuzzy automata theory will be presented. We will observe the most important type of crisp deterministic fuzzy automata, the Nerode fuzzy automaton. This automaton can be obtained from a given fuzzy automaton using the generalization of the classical subset construction method for fuzzy automata [53, 56]. Further, we will introduce a factor fuzzy automaton obtained employing fuzzy equivalence and an afterset and a foreset fuzzy automaton obtained using fuzzy quasi-order on the set of states of the given automaton. Finally, the two types of simulations between fuzzy automata, named forward and backward simulations, will be observed. Considering four possible cases when a fuzzy relation and its inverse are forward or backward simulations we will consider four types of bisimulations: forward, backward, forward-backward and backward-forward bisimulations. These concepts have been introduced by Ćirić et al. in [22].

In the chapter three, we will provide efficient algorithms which decide whether there exist any of the above-mentioned types of simulations/bisimulations between the given fuzzy automata and compute the greatest one, whenever it exists. The algorithms are based on the computing of the greatest post-fixed point, contained in a given fuzzy relation, of an isotone function on the lattice of fuzzy relations. At the end of this chapter, the work of the algorithm would be illustrated by a few examples, which also show that no type of bisimulations has a distinct advantage with respect to the others. In other words, given examples show that between two fuzzy automata there may exist the certain type of bisimulation, while there is no other of tree types of bisimulations.

In the fourth chapter we will introduce two new types of bisimulations for fuzzy automata, weak forward and weak backward bisimulations. Weak forward and weak backward bisimulations provide better approximations of the language-equivalence and when employed in the state reduction they provide better reductions than forward and backward bisimulations. Here we will propose the procedures for deciding whether there exist weak forward or weak backward simulations and bisimulations, and for computing the greatest ones, whenever they exist. Then weak bisimulations will be studied in conjunction with the concept of a uniform fuzzy relations. So called uniform weak bisimulations between two fuzzy automata will be examined here and we will give their characterization in terms of isomorphisms between their Nerode and reverse Nerode automata.

The state reduction of fuzzy automata will be studied in the fifth chapter. The right invariant and the left invariant fuzzy quasi-orders and fuzzy equivalences, introduced in [117] are, in general, equally good in the state reduction of fuzzy automata, but in [117] it was also shown that right and left invariant fuzzy quasi-orders give better reductions than right and left invariant fuzzy equivalences. The new algorithm for computing the greatest right (left) invariant fuzzy quasi-order (equivalence) for the given fuzzy automaton, based on the famous Paige-Tarjan's coarsest partition problem [88] will be introduced here. Afterwards, we will determine the complexity of this algorithm and we will propose its modification- an algorithm for computing the greatest right invariant equivalence on the given non-deterministic automaton.

At the end, I want to thank my mentor, Professor Jelena Ignjatović, for the generous help and friendly support during the preparation of this dissertation and beyond, the professor Miroslav Ćirić, for the constant motivation and inspiration for scientific research. Moreover I want to thank to Mr.Ivan Stanković for kind help in drafting of the program which implements the results from the dissertation. Also I want to thank to my family and to all people who have supported me and supplied the necessary understanding during the preparation of the thesis.

Chapter 1
Fundamental concepts

In this chapter we introduce some basic notions and notations of the lattice theory and the theory of fuzzy sets, as well as some fundamental properties of the introductory concepts that will be used in our further work.

This chapter is composed of four sections. In Section 1.1, crisp relations and their important types will be introduced and special attention will be paid to partial orders and ordered sets. Afterwards, we will introduce notions of the order preserving mappings, isomorphisms, and basic notions from the lattice theory related to fuzzy sets. A notion of the residuated lattice, which will be used as a structure of truth values for fuzzy automata and fuzzy relations, and its properties will be presented in Section 1.2. In Section 1.3, notions of a fuzzy set and a fuzzy relation will be given. In the same section, we will define the inverse fuzzy relation, composition of fuzzy relations and notions of fuzzy equivalence and fuzzy quasi-order relations. The special type of fuzzy relations, called uniform fuzzy relations, will be introduced in Section 1.4. The uniform fuzzy relations play an important role in the theory of fuzzy simulations and bisimulations, and their main features will be established in this section.

The terms and notations in this chapter will be introduced in accordance with notions and notations from the following books: Blyth [5] and Birkhoff [6] for Section Ordered sets and lattices, R. Bělohlávek [4] for Section Complete residuated lattices and Section Fuzzy sets and relations. The notations from Section Uniform fuzzy relation are according to the work of M. Ćirić, J. Ignjatović, S. Bogdanović[20].

1.1. Ordered sets and lattices

Let A be a non-empty set. A *binary relation*, or just a relation on A is any subset R of the Cartesian product $A \times A$.

The special types of relations which are widely used are so called, trivial relations and the identity relation given by:

- the *empty relation*, usually denoted by $\emptyset$;
- the *identity relation* $\triangle_A = \{(x,x) | x \in A\}$;
- the *universal relation* $\nabla_A = \{(x,y) | x,y \in A\}$.

Let R be a relation on a set A. If elements $a,b \in A$ are in a relation R, it can be written $(a,b) \in R$, or more usual aRb.

The *composition* of relations R and S on the set A is a relation $R \circ S$ on A defined by:

$$R \circ S = \{(a,c) \in A \times A | \ (\exists\, b \in A)(a,b) \in R \text{ and } (b,c) \in S\}.$$

The *inverse* relation of the given relation R on A is a relation R^{-1} on A defined by:

$$R^{-1} = \{(a,b) | (b,a) \in R\}.$$

Given relation R on a non-empty set A is called:

(1) *reflexive* if $(a,a) \in R$ for every $a \in A$, that is, if $\triangle_A \subseteq R$;
(2) *symmetric* if $(a,b) \in R$ implies $(b,a) \in R$ for all $a,b \in A$, i.e., $R^{-1} \subseteq R$;
(3) *antisymmetric* if $(a,b) \in R$ and $(b,a) \in R$ implies $a = b$ for all $a,b \in A$, i.e.,

$$R^{-1} \cap R = \triangle_A;$$

(4) *transitive* if $(a,b) \in R$ and $(b,c) \in R$ implies $(a,c) \in R$ for all $a,b,c \in A$, i.e.,

$$R \circ R \subseteq R.$$

A reflexive, symmetric and transitive relation on the non-empty set A is called a *equivalence relation* on A. A reflexive, antisymmetric and transitive relation on the non-empty set A is called a *partial order* on A, or briefly an *order* on A. The order is usually denoted by $\leqslant$.

Thus $\leqslant$ is an order on A if and only if :

(1) $a \leqslant a$ for every $a \in A$;
(3) If $a \leqslant b$ and $b \leqslant a$ implies $a = b$ for all $a,b \in A$;
(4) If $a \leqslant b$ and $b \leqslant c$ implies $a \leqslant c$ for all $a,b,c \in A$.

A pair $(A,\leqslant)$, where A is a non-empty set and $\leqslant$ is an order on A is called a *partially ordered set* or just an *ordered set*. For the sake of simplicity instead of statement "$(A,\leqslant)$ is a ordered set" we will use "A is an ordered set". Important examples are:

Example 1 Ordered set $(\mathbb{N}, |)$, where $\mathbb{N}$ is the set of natural numbers and the relation $|$ of divisibility, defined by $m|n$ if and only if m divides n, is an order on $\mathbb{N}$;

Example 2 Ordered set $(\mathscr{P}(A),\subseteq)$, where $\mathscr{P}(A)$ is the set of all subsets of a non-empty set A and the relation $\subseteq$ of set inclusion is an order on $\mathscr{P}(A)$.

A partially ordered set P is said to satisfy the *descending chain condition* (briefly *DCC*) if every descending sequence of elements of P eventually terminates, i.e., if for every descending sequence $\{a_k\}_{k\in\mathbb{N}}$ of elements of P there exists $k \in \mathbb{N}$ such that $a_k = a_{k+l}$, for all $l \in \mathbb{N}$. In other words, P satisfies DCC if there is no infinite descending chain in P.

If $\leqslant$ is an order on A, then $<$ denote the relation on A given by:

$$a < b \text{ if and only if } a \leqslant b \text{ and } a \neq b,$$

and with $\geqslant$ and $>$ we denote the inverse of relations $\leqslant$ and $<$, respectively. An ordered $\leqslant$ on A is a *linear order* on A if for every $a,b \in A$ holds $a \leqslant b$ or $b \leqslant a$. In this case A is a *linearly ordered set*.

A mapping ϕ from the ordered set A to the ordered set B is called *isotonic* or *order preserving* if $a \leqslant b$ implies $\phi(a) \leqslant \phi(b)$ for all $a,b \in A$. Similarly, a mapping ϕ from the ordered set A to the ordered set B is called *antitonic* if $a \leqslant b$ implies $\phi(a) \geqslant \phi(b)$ for all $a,b \in A$. A mapping ϕ is an *isomorphism* of ordered sets A and B, or *ordered isomorphism* from A to B, if ϕ is a bijection from A to B then ϕ and ϕ^{-1} both are isotonic mappings.

Let A be an ordered set. An element $a \in A$ is called:

- the *minimal element* of the set A, if $x \leqslant a$ implies $x = a$ for every $x \in A$, that is, if there is no element in the set A which is strictly smaller than a;
- the *maximal element* of the set A, if $a \leqslant x$ implies $x = a$ for every $x \in A$, that is, if there is no element in the set A which is strictly larger than a;
- *the least element* of the set A, if for every $x \in A$ holds $a \leqslant x$, i.e., if a is less or equal than any element from A;
- *the greatest element* of the set A, if for every $x \in A$ holds $x \leqslant a$, i.e., if a is greater or equal than any element from A.

Let H be a non-empty subset of ordered set A. An element $a \in A$ is called:

- the *upper bound* of the set H, if $x \leqslant a$ for every $x \in H$;
- the *lower bound* of the set H, if $a \leqslant x$ for every $x \in H$;
- the *least upper bound* or the *supremum* of the set H, if it is the least element in the set of all upper bounds of H, in other words, if it is the upper bound of H and for any upper bound b of the set H there holds $a \leqslant b$;
- the *greatest upper bound* or the *infimum* of the set H, if it is the greatest element in the set of all lower bounds of H, in other words, if it is the lower bound of H and for any lower bound b of the set H there holds $b \leqslant a$.

The supremum of the set H, if it exists, is denoted by $\bigvee H$, whereas the infimum of H, if it exists, is usually denoted by $\bigwedge H$. If $H = \{a_i \mid i \in I\}$, instead of $\bigvee H$ and $\bigwedge H$ we can write, respectively:

$$\bigvee_{i\in I} a_i \quad \text{and} \quad \bigwedge_{i\in I} a_i.$$

An ordered set, such that every two-element subset has the supremum and the infimum is called a *lattice*. It can be easily proven, by induction, that every finite subset of a lattice has a supremum and infimum. However, for an infinite subset of a lattice, it doesn't have to be the case.

Let L be a lattice. Then, we can define two binary operations $\bigvee$ and $\bigwedge$ on L, as follows here:

$$\bigvee(a,b) = a \vee b \quad \text{and} \quad \bigwedge(a,b) = a \wedge b.$$

Operations $\bigvee$ and $\bigwedge$ are called *union* and *intersection*, respectively. Therefore, $\bigvee H$ is a *union of the set H* and $a \vee b$ is a *union of elements a and b*, and similarly $\bigwedge H$ is a *intersection of the set H* and $a \wedge b$ is a *intersection of elements a and b*.

Theorem 1.1. *Let L be a lattice. Then for every $a,b,c \in L$ the following holds:*

(L1) $a \wedge a = a, \quad a \vee a = a$ *(idempotents);*
(L2) $a \wedge b = b \wedge a, \quad a \vee b = b \vee a$ *(commutativity);*
(L3) $(a \wedge b) \wedge c = a \wedge (b \wedge c), \quad (a \vee b) \vee c = a \vee (b \vee c)$ *(associativity);*
(L4) $a \wedge (a \vee b) = a, \quad a \vee (a \wedge b) = a$ *(absorption).*

The conditions $(L1)-(L4)$ of the Theorem 1.1 are called *the lattice axioms*.

A subset X of the lattice L is called *sublattice* if $a \wedge b \in X$ and $a \vee b \in X$ for all elements $a,b \in X$.

For the lattice L and an element $a \in L$, sublattices

$$[a) = \{x \in L \mid a \leqslant x\} \quad \text{and} \quad (a] = \{x \in L \mid x \leqslant a\}$$

are half-open intervals of the lattice L, and for all $a,b \in L$ sublattices

$$(a,b) = \{x \in L \mid a < x < b\} \quad \text{and} \quad [a,b] = \{x \in L \mid a \leqslant x \leqslant b\}$$

are the open interval and the closed interval, respectively.

If a non empty set L, together with arbitrary elements $a,b \in L$ contains $a \wedge b$, or $a \vee b$ then L is called $\wedge$*-sublattice* (*lower sublattice*), or $\vee$*-sublattice* (*upper sublattice*), respectively.

A non-empty subset I of a lattice L is called *ideal* if:

(1) $x \leqslant a$ implies $x \in I$, for every element $a \in I$ and $x \in L$;
(2) $a \vee b \in I$, for all $a,b \in I$.

It can easily be shown that subset I is an ideal of the lattice L if:

$$a \vee b \in I \quad \text{if and only if both } a \text{ and } b \text{ are elements of } I.$$

The dual notion of this concept is the *dual ideal*. Therefore, a non-empty subset D of a lattice L is the dual ideal if:

(1) $a \leqslant x$ implies $x \in I$, for every element $a \in D$ and $x \in L$;
(2) $a \wedge b \in I$, for all $a,b \in I$.

Let $a \in L$. Notice that the half-open intervals $(a]$ and $[a)$, are the ideal and the dual ideal of the lattice L, respectively, and they are called *the principal ideal generated by* a and *the principal dual ideal generated by* a .

The least element of the lattice L, if it exists, is denoted by 0, and the greatest element,if it exists, is denoted by 1. A *bounded lattice* is a lattice that has the greatest element 1 and the least element 0.

It can easily be shown that on every lattice L the following conditions are equivalent:

(L5) $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$, for all $a,b,c \in L$;
(L5′) $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$, for all $a,b,c \in L$.

The lattice which satisfies any of previous conditions is called a *distributive* lattice.

Let L be a bounded lattice with 0 and 1. An element $b \in L$ is called a complement of a element $a \in L$ if :

$$a \wedge b = 0, \quad a \vee b = 1.$$

In this case, $a \in L$ is the complement of the element $b \in L$. As we can see, the relation " to be a complement" is a symmetric relation. In every distributive lattice it can easily be shown that every element $a \in L$ has at most one complement, which will be denoted with $a' \in L$.

A bounded distributive lattice in which every element has a complement is called a *Boolean lattice*. The mapping $a \rightarrow a'$ is a unary operation on L called *complement operation*.

As we mentioned before, every finite subset of a lattice has a supremum and infimum. However, this is not necessarily true when a subset of the network is infinite. A lattice, in which every subset (finite as well as infinite), has a supremum and infimum is called a *complete lattice*. Clearly, every complete lattice is bounded. A subset K of a complete lattice L is a complete sublattice of L if supremum and infimun (in L) of every non-empty subset of K belongs to K.

Most studied examples of lattices are:

Example 1. The set $\mathbb{N}$ of natural numbers is partially ordered set with respect to the division relation, and with respect to this relation the set $\mathbb{N}$ is a lattice, where operations of infimum and supremum are defined by:

$$a \wedge b = gcd(a,b), \quad \text{and} \quad a \vee b = lcm(a,b).$$

Example 2. For every set A, the partitive set of the set A is denoted by $\mathscr{P}(A)$. This set is partially ordered with the inclusion relation, and with respect to this relation $\mathscr{P}(A)$ is a lattice. The operations infimum and supremum are set intersection and set union, respectively. Zero and one are $\emptyset$ and A, respectively.

Example 3. With $\mathscr{E}(A)$ denote the set of all equivalence relations on a nonempty set A. This set is partially ordered with the inclusion relation, and with respect to that order it is a complete lattice. Namely, the operation of infimum on $\mathscr{E}(A)$ is the operation of intersection of two relations, whereas the operation of supremum is not a union of relations, because of the fact that the union of two equivalences does not have to be an equivalence relation. For an arbitrary non-empty subset B of $\mathscr{E}(A)$, the set $\langle B\rangle$ is the sub-semigroup of the semigroup of binary relations on the set A generated by B. Then $\langle B\rangle$ is equal to the set union of all relations from $\langle B\rangle$. Zero and one are $\triangle_A$ and ∇_A, respectively.

1.2. Complete residuated lattices

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth - truth values between "completely true" and "completely false". It is the logic underlying modes of reasoning which are approximate rather than exact. The importance of fuzzy logic derives from the fact that most modes of human reasoning and especially common sense reasoning are approximate in nature.

In the fuzzy logic, each proposition assigns a truth degree taken from some scale L of truth degrees. If propositions φ and ψ are assigned truth degrees a and b, which are denoted by $||\varphi|| = a$ and $||\psi|| = b$, then $a \leqslant b$ means that φ is less true than ψ. According to this, L is required to be partially ordered. Further, it is convenient to require 0 to assign the truth value of "completely false" and 1 to assign the truth value of "completely true". Therefore, L is equipped with partial order $\leqslant$ and 0 and 1 are the least and the greatest element in L, respectively.

Let $\{\varphi_i \,|\, i \in I\}$ denote the set of propositions. The generalization of the classical bivalent logic, leads to the assumption that the truth value of expression " there exists i such that φ_i" is a supreme of all truth values φ_i, that is, ||"there exists i such that φ_i" || $= \bigvee_{i\in I} ||\varphi_i||$ (where $||\varphi||$ is a truth value of φ). Therefore, L is required to have arbitrary supremum and (dually) infimum, i.e. $(L, \leqslant, 0, 1)$ is required to be a complete lattice.

Scale L needs to be equipped with (truth functions of) logical connectives. As in classical logic, we need binary functions $\otimes : L \times L \to L$ of conjunction and $\to: L \times L \to L$ of implication.

Naturally, the conjunction should be commutative and associative. Moreover, the fact that the truth degree of conjunction of a proposition φ with a fully true proposition $\psi(\,||\psi|| = 1)$ equals the truth degree of φ, implies that 1 should be a natural element for $\otimes$. Thus, it is required $(L, \otimes, 1)$ to be a commutative monoid. Also, the desirable property of conjunction is monotony - more true the propositions, more true their conjunction.

In the classical logic one of the most commonly used concept is modus ponens. Modus ponens is an inference rule saying: if φ is valid and $\varphi \to \psi$ is valid then we may infer that ψ is valid. An appropriate formulation of modus ponens in the fuzzy setting is the following: if φ is valid in degree at least a and $\varphi \to \psi$ is valid in degree at least b then we may infer that ψ is valid in degree at least $a \otimes b$.

Next, we want modus ponens to satisfy two points: it should be sound and, at the same time, it should yield the highest possible estimation of validity of ψ. Soundness: The requirement of soundness says that when evaluating formulas, if the truth degree of φ is at least a ($a \leqslant \|\varphi\|$) and the truth degree of $\varphi \to \psi$ is at least b ($b \leqslant \|\varphi \Rightarrow \psi\|$) then the truth degree of ψ is at least as high as the degree obtained by modus ponens ($a \otimes b \leqslant \|\psi\|$). Particularly, if $\|\varphi\| = a$ and $\|\psi\| = c$, then since $\|\varphi \Rightarrow \psi\| = a \to c$, soundness says that $b \leqslant a \to c$ implies $a \otimes b \leqslant c$.

The highest possible estimation of validity of ψ: Let $\|\varphi\| = a$ and $\|\psi\| = c$. Then $\|\varphi \Rightarrow \psi\| = a \to c$ and since we require soundness, $a \otimes (a \to c)$ needs to be a lower estimation of c, i.e. $a \otimes (a \to c) \leqslant c$. Since $\otimes$ is non-decreasing, the lower estimation $a \otimes (a \to c)$ is higher, higher is $a \to c$. Since we want $a \otimes (a \to c)$ to be the highest possible estimation of c, $a \to c$ needs to be the highest degree for which $a \otimes (a \to c) \leqslant c$. That is, if b is any truth degree for which $a \otimes b \leqslant c$ then $b \leqslant a \to c$. In other words, the requirement of the highest estimation of validity of ψ yields that $a \otimes b \leqslant c$ implies $b \leqslant a \to c$.

According to this, $\to$ and $\otimes$ should form an *adjoint pair*, i.e., they should satisfy the *adjunction property*: for all $a, b, c \in L$,

$$a \otimes b \leqslant c \;\Leftrightarrow\; a \leqslant b \to c. \tag{1.1}$$

The algebraic structure which satisfies all previous conditions is a *complete residuated lattice*.

A *residuated lattice* is an algebra $\mathscr{L} = (L, \wedge, \vee, \otimes, \to, 0, 1)$ such that:

(L1) $(L, \wedge, \vee, 0, 1)$ is a lattice with the least element 0 and the greatest element 1,
(L2) $(L, \otimes, 1)$ is a commutative monoid with the unit 1,
(L3) $\otimes$ and $\to$ form an adjoint pair.

If, in addition, $(L, \wedge, \vee, 0, 1)$ is a complete lattice, then $\mathscr{L}$ is called a *complete residuated lattice*.

The operations $\otimes$ (called *multiplication*) and $\to$ (called *residuum*) are intended for modeling the conjunction and implication of the corresponding logical calculus, and supremum ($\bigvee$) and infimum ($\bigwedge$) are intended for modeling of the existential and general quantifier, respectively.

On the complete residuated lattice the following operations can be defined:

$$\textit{biresiduum}\ (\text{or } \textit{biimplication}) : a \leftrightarrow b = (a \to b) \wedge (b \to a),$$
$$\textit{negation} : \neg a = a \to 0,$$
$$\textit{n-th degree} : a^0 = 1 \text{ and } a^{n+1} = a^n \otimes a.$$

Biimplication is the operation used for modeling the equivalence of truth values, whereas the negation is used for modeling the complement of a truth value.

The most studied and applied structures of truth values, defined on the real unit interval [0,1] with:

$$a \wedge b = \min(a,b) \quad \text{and} \quad a \vee b = \max(a,b)$$

are: the *Łukasiewicz structure*:

$$a \otimes b = \max(a+b-1,0), \quad a \rightarrow b = \min(1-a+b,1),$$

the *Goguen* (*product*) *structure*:

$$a \otimes b = a \cdot b, \qquad a \rightarrow b = \begin{cases} 1, & \text{if } a \leqslant b, \\ b/a, & \text{otherwise,} \end{cases}$$

and the *Gödel structure*:

$$a \otimes b = \min(a,b), \qquad a \rightarrow b = \begin{cases} 1, & \text{if } a \leqslant b, \\ b, & \text{otherwise.} \end{cases}$$

Another important set of truth values is the set $\{a_0, a_1, \ldots, a_n\}$, $0 = a_0 < \cdots < a_n = 1$, with

$$a_k \otimes a_l = a_{\max(k+l-n,0)} \quad \text{and } a_k \rightarrow a_l = a_{\min(n-k+l,n)}.$$

A special case of the latter algebras is the two-element Boolean algebra of classical logic with the support {0,1}. The only adjoint pair on the two-element Boolean algebra consists of the classical conjunction and implication operations. This structure of truth values we call the *Boolean structure*.

All structures: the Łukasiewicz, Goguen and Gödel are residuated lattices induced by t-norms.

t-norm is a binary operation on real [0,1] which is associative, commutative, monotone and where 1 is unit element, that is $\otimes$ is mapping $\otimes : [0,1] \times [0,1] \rightarrow [0,1]$ which satisfies the following conditions:

$$\begin{aligned} (a \otimes b) \otimes c &= a \otimes (b \otimes c), \\ a \otimes b &= b \otimes a, \\ b_1 \leqslant b_2 &\Rightarrow a \otimes b_1 \leqslant a \otimes b_2, \\ a \otimes 1 &= a. \end{aligned}$$

Generally, an algebra $([0,1], \vee, \wedge, \otimes, \rightarrow, 0, 1)$ is a complete residuated lattice if and only if $\otimes$ is a left-continuous t-norm (i.e. $lim_{n\rightarrow\infty}(a_n \otimes b) = (lim_{n\rightarrow\infty} a_n) \otimes b$) and then the residuum is defined by $x \rightarrow y = \bigvee\{u \in [0,1] \,|\, u \otimes x \leqslant y\}$.

Here, we recall some important truth structures, which are residuated lattices, but satisfy some additional conditions.

A residuated lattice $\mathscr{L}$ satisfying $x \otimes y = x \wedge y$ is called a *Heyting algebra*. Moreover, if $\mathscr{L}$ is a complete lattice, than it is called a *complete Heyting algebra*, if the partial order in $\leqslant$ in $\mathscr{L}$ is linear, then $\mathscr{L}$ is a linearly ordered Heyting algebra. The most important example of the linearly ordered Heyting algebra is real unit interval $[0,1]$ with the Gödel pair of adjoint operations, i.e. with the standard minimum t-norm.

BL-algebra (Basic Logic Algebra) is a residuated lattice which satisfies the condition $a \wedge b = a \otimes (a \rightarrow b)$ (divisibility) and $(a \rightarrow b) \vee (b \rightarrow a) = 1$ (prelinearity).

MV-algebra (Multi Valued Algebra) is a BL-algebra in which $a = \neg\neg a$ (double negation is allowed).

P-algebra (product algebra) is BL-algebra which satisfies $(c \rightarrow 0) \rightarrow 0 \leqslant ((a \otimes c) \rightarrow (b \otimes c)) \rightarrow (a \rightarrow b)$ and $a \wedge (a \rightarrow 0) = 0$.

G-algebra (Gödel algebra) is BL-algebra which satisfies $a \otimes a = a$ (idempotents).

Boolean algebra is a residuated lattice which is both the Heyting algebra and the MV-algebra.

Because of its prominent monoid structures, in some papers the residuated lattices are called integral, commutative, residuated l-monoids, whereas some authors the notion of residuated lattice use for more general structures.

If every finitely generated subalgebra of a residuated lattice $\mathscr{L}$ is finite, then $\mathscr{L}$ is called *locally finite*. For example, every Gödel algebra, and hence, the Gödel structure, is locally finite, whereas the product structure is not locally finite.

The following theorem recalls the basic properties of residuated lattices:

Theorem 1.2. *For every residuated lattice the following holds:*

$$b \leqslant a \rightarrow (a \otimes b), \qquad a \leqslant (a \rightarrow b) \rightarrow b, \tag{1.2}$$

$$a \otimes (a \rightarrow b) \leqslant b, \tag{1.3}$$

$$a \leqslant b \quad \Leftrightarrow \quad a \rightarrow b = 1, \tag{1.4}$$

$$a \rightarrow a = 1, \quad a \rightarrow 1 = 1, \quad 1 \rightarrow a = a, \tag{1.5}$$

$$0 \rightarrow a = 1, \tag{1.6}$$

$$a \otimes 0 = 0 \otimes a = 0, \tag{1.7}$$

$$a \otimes b \leqslant a, \quad a \leqslant b \rightarrow a, \tag{1.8}$$

$$a \otimes b \leqslant a \wedge b, \tag{1.9}$$

$$a \otimes b \rightarrow c = a \rightarrow (b \rightarrow c), \tag{1.10}$$

$$(a \rightarrow b) \otimes (b \rightarrow c) \leqslant (a \rightarrow c), \tag{1.11}$$

$$a \rightarrow b \textit{ is the greatest element of } \{c \,|\, a \otimes c \leqslant b\} \tag{1.12}$$

$$a \otimes b \textit{ is the least element of } \{c \,|\, a \leqslant b \rightarrow c\}. \tag{1.13}$$

The next theorem shows that with respect to $\leqslant$, the operation $\otimes$ is isotonic in both arguments, the operation $\to$ is isotonic in the second and antitonic in the first argument.

Theorem 1.3. *In every residuated lattice the following holds:*

$$b_1 \leqslant b_2 \Rightarrow a \otimes b_1 \leqslant a \otimes b_2, \tag{1.14}$$

$$b_1 \leqslant b_2 \Rightarrow a \to b_1 \leqslant a \to b_2, \tag{1.15}$$

$$a_1 \leqslant a_2 \Rightarrow a_2 \to b \leqslant a_1 \to b. \tag{1.16}$$

In the sequel, some other properties of residuated lattices will be presented.

Theorem 1.4. *In every residuated lattice the following inequalities hold:*

$$a \to b \leqslant (a \wedge c) \to (b \wedge c), \tag{1.17}$$

$$a \to b \leqslant (a \vee c) \to (b \vee c), \tag{1.18}$$

$$a \to b \leqslant (a \otimes c) \to (b \otimes c), \tag{1.19}$$

$$a \to b \leqslant (b \to c) \to (a \to c), \tag{1.20}$$

$$a \to b \leqslant (c \to a) \to (c \to b). \tag{1.21}$$

The next theorem describes a relationship between operations $\vee$ and $\wedge$, for any family of residuated lattice elements, and operations $\otimes$ and $\to$.

Theorem 1.5. *The following assertions hold for every index set I:*

$$a \otimes \Big(\bigvee_{i \in I} b_i\Big) = \bigvee_{i \in I} (a \otimes b_i), \tag{1.22}$$

$$a \to \Big(\bigwedge_{i \in I} b_i\Big) = \bigwedge_{i \in I} (a \to b_i), \tag{1.23}$$

$$\Big(\bigvee_{i \in I} a_i\Big) \to b = \bigwedge_{i \in I} (a_i \to b), \tag{1.24}$$

$$\bigvee_{i \in I} (a_i \to b) = \Big(\bigwedge_{i \in I} a_i\Big) \to b, \tag{1.25}$$

$$a \otimes \bigwedge_{i \in I} b_i \leqslant \bigwedge_{i \in I} (a \otimes b_i). \tag{1.26}$$

$$\bigwedge_{i \in I} (a_i \to b_i) \leqslant \Big(\bigwedge_{i \in I} a_i\Big) \to \Big(\bigwedge_{i \in I} b_i\Big) \tag{1.27}$$

$$\bigwedge_{i \in I} (a_i \to b_i) \leqslant \Big(\bigvee_{i \in I} a_i\Big) \to \Big(\bigvee_{i \in I} b_i\Big) \tag{1.28}$$

$$\bigvee_{i \in I} (a \to b_i) \leqslant a \to \Big(\bigwedge_{i \in I} b_i\Big) \tag{1.29}$$

Theorem 1.6. *In every residuated lattice the following assertions hold:*

$$a \to b = ((a \to b) \to b) \to b, \tag{1.30}$$
$$a \otimes (a \to b) = b \quad \Leftrightarrow \quad (\exists c)(a \otimes c = b), \tag{1.31}$$
$$a \to (a \otimes b) = b \quad \Leftrightarrow \quad (\exists c)(a \to c = b), \tag{1.32}$$
$$(b \to a) \to a = b \quad \Leftrightarrow \quad (\exists c)(c \to a = b), \tag{1.33}$$
$$(a \wedge b) \otimes (a \vee b) \leqslant a \otimes b, \tag{1.34}$$
$$a \vee b \leqslant ((a \to b) \to b) \wedge ((b \to a) \to a), \tag{1.35}$$
$$a \wedge b \geqslant a \otimes (a \to b), \tag{1.36}$$
$$a \otimes (b \to c) \leqslant b \to (a \otimes c). \tag{1.37}$$

1.3. Fuzzy sets and fuzzy relations

The complete residuated lattice $\mathscr{L}$ will be the structure of truth values in further text.

A *fuzzy subset* of a set A *over* $\mathscr{L}$, or simply a *fuzzy subset* of A, is any mapping from A into L. Ordinary crisp subsets of A are considered as fuzzy subsets of A taking membership values in the set $\{0,1\} \subseteq L$.

Let f and g be two fuzzy subsets of A. The *equality* of f and g is defined as the usual equality of mappings, i.e.,

$$f = g \text{ if and only if } f(x) = g(x), \quad \text{for every } x \in A.$$

The *inclusion* $f \leqslant g$ is also defined pointwise:

$$f \leqslant g \text{ if and only if } f(x) \leqslant g(x), \quad \text{for every } x \in A.$$

Endowed with this partial order the set $\mathscr{F}(A)$ of all fuzzy subsets of A forms a complete residuated lattice, in which the meet (intersection) $\bigwedge_{i\in I} f_i$ and the join (union) $\bigvee_{i\in I} f_i$ of an arbitrary family $\{f_i\}_{i\in I}$ of fuzzy subsets of A are mappings from A to L defined by

$$\left(\bigwedge_{i\in I} f_i\right)(x) = \bigwedge_{i\in I} f_i(x), \qquad \left(\bigvee_{i\in I} f_i\right)(x) = \bigvee_{i\in I} f_i(x),$$

and the *product* $f \otimes g$ is a fuzzy subset defined by $f \otimes g(x) = f(x) \otimes g(x)$, for every $x \in A$.

The *crisp part* of a fuzzy subset $f \in \mathscr{F}(A)$ is a crisp subset $\widehat{f} = \{a \in A \mid f(a) = 1\}$ of A. It may be noted that many authors use the term "kernel" instead of the term "crisp part" and denote "$ker f$" instead of $\widehat{f}$. However, in this work we will use the term "kernel" in its usual meaning. Namely, the kernel of fuzzy subset f of a set A, denoted by $ker f$, is an equivalence relation on A defined by:

$$ker f = \{(x,y) \in A \times A \mid f(x) = f(y)\}.$$

The *hight of fuzzy subset f*, denoted by $\|f\|$, is defined by:

$$\|f\| = \bigvee_{x \in A} f(x).$$

A *fuzzy relation* between sets A and B (in this order) is any mapping from $A \times B$ to L, i.e. , any fuzzy subset of $A \times B$, and the equality, inclusion (ordering), joins and meets of fuzzy relations are defined as for fuzzy sets. The set of all fuzzy relations between A and B will be denoted by $\mathscr{R}(A,B)$. In particular, a fuzzy relation on a set A is any function from $A \times A$ to L, i.e., any fuzzy subset of $A \times A$. The set of all fuzzy relations on A will be denoted by $\mathscr{R}(A)$.

The *inverse*(*converse*, or *transpose*) of a fuzzy relation $\varphi \in \mathscr{R}(A,B)$ is a fuzzy relation $\varphi^{-1} \in \mathscr{R}(B,A)$ defined by

$$\varphi^{-1}(b,a) = \varphi(a,b), \text{ for all } a \in A \text{ and } b \in B.$$

A crisp relation is a fuzzy relation which takes values only in the set $\{0,1\}$, and if φ is a crisp relation of A to B, then expressions "$\varphi(a,b) = 1$" and "$(a,b) \in \varphi$" will have the same meaning.

The crisp part of fuzzy relation φ, denoted by $\widehat{\varphi}$ is a crisp relation which satisfies:

$$(a,b) \in \widehat{\varphi} \quad \Leftrightarrow \quad \varphi(a,b) = 1,$$

for every $a,b \in A$.

For non-empty sets A, B, C and fuzzy relations $\varphi \in \mathscr{R}(A,B)$ and $\psi \in \mathscr{R}(B,C)$, their *composition* $\varphi \circ \psi$ is a fuzzy relation from $\mathscr{R}(A,C)$ defined by

$$(\varphi \circ \psi)(a,c) = \bigvee_{b \in B} \varphi(a,b) \otimes \psi(b,c), \tag{1.38}$$

for all $a \in A$ and $c \in C$. Next, if $f \in \mathscr{F}(A)$, $\varphi \in \mathscr{R}(A,B)$ and $g \in \mathscr{F}(B)$, the compositions $f \circ \varphi$ and $\varphi \circ g$ are fuzzy subsets of B and A, respectively, which are defined by

$$(f \circ \varphi)(b) = \bigvee_{a \in A} f(a) \otimes \varphi(a,b), \quad (\varphi \circ g)(a) = \bigvee_{b \in B} \varphi(a,b) \otimes g(b), \tag{1.39}$$

for every $a \in A$ and $b \in B$.

In particular, for fuzzy subsets f and g of A we write

$$f \circ g = \bigvee_{a \in A} f(a) \otimes g(a). \tag{1.40}$$

Let A, B, C and D be non-empty sets. Then for any $\varphi_1 \in \mathscr{R}(A,B)$, $\varphi_2 \in \mathscr{R}(B,C)$ and $\varphi_3 \in \mathscr{R}(C,D)$ we have:

$$(\varphi_1 \circ \varphi_2) \circ \varphi_3 = \varphi_1 \circ (\varphi_2 \circ \varphi_3), \tag{1.41}$$

and for $\varphi_0 \in \mathscr{R}(A,B)$, $\varphi_1, \varphi_2 \in \mathscr{R}(B,C)$ and $\varphi_3 \in \mathscr{R}(C,D)$ we have that

$$\varphi_1 \leqslant \varphi_2 \quad \text{implies} \quad \varphi_0 \circ \varphi_1 \leqslant \varphi_0 \circ \varphi_2 \quad \text{and} \quad \varphi_1 \circ \varphi_3 \leqslant \varphi_2 \circ \varphi_3. \tag{1.42}$$

Further, for any $\varphi \in \mathscr{R}(A,B)$, $\psi \in \mathscr{R}(B,C)$, $f \in \mathscr{F}(A)$, $g \in \mathscr{F}(B)$ and $h \in \mathscr{F}(C)$ we can easily verify that

$$(f \circ \varphi) \circ \psi = f \circ (\varphi \circ \psi), \quad (f \circ \varphi) \circ g = f \circ (\varphi \circ g), \quad (\varphi \circ \psi) \circ h = \varphi \circ (\psi \circ h), \tag{1.43}$$

and consequently, the parentheses in (1.43) can be omitted, as well as the parentheses in (1.41). Finally, for all $\varphi, \varphi_i \in \mathscr{R}(A,B)$, ($i \in I$) and $\psi, \psi_i \in \mathscr{R}(B,C)$, ($i \in I$) we have that

$$(\varphi \circ \psi)^{-1} = \varphi^{-1} \circ \psi^{-1} \tag{1.44}$$

$$\varphi \circ \Big(\bigvee_{i \in I} \psi_i\Big) = \bigvee_{i \in I} (\varphi \circ \psi_i), \quad \Big(\bigvee_{i \in I} \varphi_i\Big) \circ \psi = \bigvee_{i \in I} (\varphi_i \circ \psi) \tag{1.45}$$

$$\Big(\bigvee_{i \in I} \varphi_i\Big)^{-1} = \bigvee_{i \in I} \varphi_i^{-1}. \tag{1.46}$$

We note that if A, B and C are finite sets of cardinality $|A| = k$, $|B| = m$ and $|C| = n$, then $\varphi \in \mathscr{R}(A,B)$ and $\psi \in \mathscr{R}(B,C)$ can be treated as $k \times m$ and $m \times n$ fuzzy matrices over $\mathscr{L}$, and $\varphi \circ \psi$ is the matrix product. Analogously, for $f \in \mathscr{F}(A)$ and $g \in \mathscr{F}(B)$ we can treat $f \circ \varphi$ as the product of a $1 \times k$ matrix f and a $k \times m$ matrix φ (vector-matrix product), $\varphi \circ g$ as the product of an $k \times m$ matrix φ and an $m \times 1$ matrix g^t, the transpose of g (matrix-vector product), and $f \circ g$ as the scalar product of vectors f and g.

A fuzzy relation R on A is said to be:

(R) *reflexive* (or *fuzzy reflexive*) if $R(a,a) = 1$, for every $a \in A$;
(S) *symmetric* (or *fuzzy symmetric*) if $R(a,b) = R(b,a)$, for all $a, b \in A$;
(T) *transitive* (or *fuzzy transitive*) if $R(a,b) \otimes R(b,c) \leqslant R(a,c)$, for all $a,b,c \in A$.

It can easily be shown, that $R \circ R = R$ holds for any reflective and transitive relation R on A.

For a fuzzy relation R on the set A, the fuzzy relation R^∞ on A defined by

$$R^\infty = \bigvee_{n \in N} R^n.$$

is the least transitive fuzzy relation on A containing R, and it is called the transitive closure of R.

A reflexive, symmetric and transitive fuzzy relation on A is called a *fuzzy equivalence*. With the respect to the inclusion of fuzzy relations, the set $\mathscr{E}(A)$ of all fuzzy equivalences on A is a complete lattice, in which the meet coincide with the ordinary intersection of fuzzy relations, but in the general case, the join in $\mathscr{E}(A)$ does not coincide with the ordinary union of fuzzy relations.

A fuzzy equivalence E on a set A is called a *fuzzy equality* if $E(a,b) = 1$ implies $a = b$, for all $a,b \in A$. In other words, E is a fuzzy equality if and only if its crisp part $\widehat{E}$ is a crisp equality.

The *equivalence class* of fuzzy relation E on A determined by $a \in A$ is the fuzzy subset E_a of A defined by

$$E_a(b) = E(a,b), \quad \text{for every } b \in A.$$

The set $A/E = \{E_a \,|\, a \in A\}$ is called the *factor set* of A w.r.t. E (cf. [4]). The *natural function* from A to A/E is the fuzzy relation $\varphi_E \in \mathscr{R}(A, A/E)$ defined by

$$\varphi_E(a, E_b) = E(a,b), \quad \text{for all } a,b \in A.$$

A fuzzy relation on a set A which is reflexive and transitive is called a *fuzzy quasi-order*, and a reflexive and transitive crisp relation on A is called a *quasi-order*. As well as the set $\mathscr{E}(A)$, the set $\mathscr{Q}(A)$ of all fuzzy quasi-orders on $\mathscr{A}$ is a complete lattice, in which the meet coincide with the ordinary intersection of fuzzy relations and, in the general case, the join in $\mathscr{Q}(A)$ does not coincide with the ordinary union of fuzzy relations. Namely, if R is the join in $\mathscr{Q}(A)$ of a family $\{R_i\}_{i\in I}$ of fuzzy quasi-orders on A, then R can be represented by:

$$R = \Big(\bigvee_{i\in I} R\Big)^{\infty} = \bigvee_{n\in N}\Big(\bigvee_{i\in I} R\Big)^{n}.$$

The *R-afterset* of a, $a \in A$, is the fuzzy set $R_a \in L^A$ defined by:

$$R_a(b) = R(a,b), \text{ for any } b \in A,$$

while the *R-foreset* of a is the fuzzy set $R^a \in L^A$ defined by:

$$R^a(b) = R(b,a), \text{ for any } b \in A.$$

The set of all R-aftersets will be denoted by A/R, and the set of all R-foresets will be denoted by $A\backslash R$. Clearly, if R is a fuzzy equivalence, then $A/R = A\backslash R$ is the set of all equivalence classes of R.

For a fuzzy quasi-order R on a set A, a fuzzy relation E_R defined by $E_R = R \wedge R^{-1}$ is a fuzzy equivalence on A, which is called a natural fuzzy equivalence of R.

A fuzzy quasi-order R on a set A is a fuzzy order if $R(a,b) = R(b,a) = 1$ implies $a = b$, for all $a,b \in A$, i.e., if the natural fuzzy equivalence E_R of R is a fuzzy equality. Clearly, a fuzzy quasi-order R is a fuzzy order if and only if its crisp part $\hat{R}$ is a crisp order.

If f is an arbitrary fuzzy subset of A, then fuzzy relations R_f and R^f on A defined by

$$R_f(a,b) = f(a) \to f(b), \quad R^f(a,b) = f(b) \to f(a), \tag{1.47}$$

for all $a,b \in A$, are fuzzy quasi-orders on A.

Also, for arbitrary fuzzy subset f on A, the fuzzy relation E_f defined by:

$$E_f(a,b) = f(a) \leftrightarrow f(b), \quad a,b \in A,$$

is a fuzzy equivalence on A.

The following theorem recalls some important features of quasi orders and natural equivalences.

Theorem 1.7. *Let R be a fuzzy quasi-order on a set A and E the natural fuzzy equivalence of R. Then*

(a) For arbitrary $a,b \in A$ the following conditions are equivalent:

(1) $E(a,b) = 1$;
(2) $E_a = E_b$;
(3) $R^a = R^b$;
(4) $R_a = R_b$.

(b) Functions $R_a \to E_a$ of A/R to A/E, and $R_a \to R^a$ of A/R to $A \setminus R$, are bijective functions.

If A is a finite set with n elements and a fuzzy quasi-order R on A is treated as an $n \times n$ fuzzy matrix over $\mathscr{L}$, then R-aftersets are row vectors, whereas R-foresets are column vectors of this matrix. The previous theorem says that i-th and j-th row vectors of this matrix are equal if and only if its i-th and j-th column vectors are equal, and vice versa. Moreover, we have that R is a fuzzy order if and only if all its row vectors are different, or equivalently, if and only if all its column vectors are different.

The next lemmas recall some properties of fuzzy quasi orders and fuzzy equivalences which will be needed in our further work.

Lemma 1.1. *Let $P,R \in \mathscr{Q}(A)$ be fuzzy quasi-orders on A. Then, relation $P \wedge R$ is also a fuzzy-quasi order.*

Lemma 1.2. *Let $E,F \in \mathscr{E}(A)$ be fuzzy equivalences on A. Then, relation $E \wedge F$ is also a fuzzy equivalence.*

Lemma 1.3. *Let $P,R \in \mathscr{Q}(A)$ and $P \leqslant R$. Then $P \circ R = R$.*

Lemma 1.4. *Let $P,R \in \mathscr{Q}(A)$ and $P \leqslant R$. Then following holds for every $a \in A$:*

$$P(b,c) \leqslant R^a(c) \to R^a(b), \quad b,c \in A.$$

Proof. According to Lemma 1.3 we have $P \circ Q = Q$. Hence, for all $a,b \in A$

$$P \circ R(b,a) = \bigvee_{c \in A} P(b,c) \otimes R(c,a) = R(b,a),$$

therefore, for all $a,b,c \in A$

$$P(b,c) \otimes R(c,a) \leqslant R(b,a),$$

which is by adjunction property equivalent to

$$P(b,c) \leqslant R(c,a) \to R(b,a) = R^a(c) \to R^a(b) = (R^a \leftarrow R^a)(b,c).$$

Therefore, $P(b,c) \leqslant R^a(c) \to R^a(b)$, for all $a,b,c \in A$. □

Lemma 1.5. *Let $E,F \in \mathscr{E}(A)$ and $E \leqslant F$. Then for all $a \in A$ the following holds:*

$$E(b,c) \leqslant F^a(b) \leftrightarrow F^a(c), \quad b,c \in A.$$

Proof. It can be proved in the similar way as in the proof of Lemma 1.4 □

1.4. Uniform fuzzy relations

Let A and B be non-empty sets and let E and F be fuzzy equivalences on A and B, respectively. If a fuzzy relation $\varphi \in \mathscr{R}(A,B)$ satisfies:

(EX1) $\varphi(a_1,b) \otimes E(a_1,a_2) \leqslant \varphi(a_2,b)$, for all $a_1,a_2 \in A$ and $b \in B$,
then it is called *extensional with respect to* E, and if it satisfies
(EX2) $\varphi(a,b_1) \otimes F(b_1,b_2) \leqslant \varphi(a,b_2)$, for all $a \in A$ and $b_1,b_2 \in B$,
then it is called *extensional with respect to* F. If φ is extensional with respect to E and F, and it satisfies
(PFF) $\varphi(a,b_1) \otimes \varphi(a,b_2) \leqslant F(b_1,b_2)$, for all $a \in A$ and $b_1,b_2 \in B$,

then it is called a *partial fuzzy function* with respect to E and F.

By the adjoint property and the symmetry the conditions (EX1) and (EX2) are equivalent to:

(EX1′) $E(a_1,a_2) \leqslant \varphi(a_1,b) \leftrightarrow \varphi(a_2,b)$, for all $a_1,a_2 \in A$ and $b \in B$,
(EX2′) $F(b_1,b_2) \leqslant \varphi(a,b_1) \leftrightarrow \varphi(a,b_2)$, for all $a \in A$ and $b_1,b_2 \in B$.

For any fuzzy relation $\varphi \in \mathscr{R}(A,B)$ we can define a fuzzy equivalence E_A^{φ} on A by

$$E_A^{\varphi}(a_1,a_2) = \bigwedge_{b \in B} \varphi(a_1,b) \leftrightarrow \varphi(a_2,b), \tag{1.48}$$

for all $a_1,a_2 \in A$, and a fuzzy equivalence E_B^{φ} on B by

$$E_B^{\varphi}(b_1,b_2) = \bigwedge_{a\in A} \varphi(a,b_1) \leftrightarrow \varphi(a,b_2), \tag{1.49}$$

for all $b_1,b_2 \in B$. They will be called *fuzzy equivalences* on A and B *induced by* φ, and in particular, E_A^{φ} will be called the *kernel* of φ, and E_B^{φ} the *co-kernel* of φ. According to (EX1′) and (EX2′), E_A^{φ} and E_B^{φ} are the greatest fuzzy equivalences on A and B, respectively, such that φ is extensional with respect to them.

A fuzzy relation $\varphi \in \mathscr{R}(A,B)$ is called just a *partial fuzzy function* if it is a partial fuzzy function with respect to E_A^{φ} and E_B^{φ} . Partial fuzzy functions were characterized in [20] as follows:

Theorem 1.8. *Let A and B be non-empty sets and let $\varphi \in \mathscr{R}(A,B)$ be a fuzzy relation. Then the following conditions are equivalent:*

(1) *φ is a partial fuzzy function;*
(2) *φ^{-1} is a partial fuzzy function;*
(3) $\varphi^{-1}\circ\varphi \leqslant E_B^{\varphi}$;
(4) $\varphi\circ\varphi^{-1} \leqslant E_A^{\varphi}$;
(5) $\varphi\circ\varphi^{-1}\circ\varphi \leqslant \varphi$.

A fuzzy relation $\varphi \in \mathscr{R}(A,B)$ is called an *$\mathscr{L}$-function* if for each $a \in A$ there exists $b \in B$ such that $\varphi(a,b) = 1$ [36, 35], and it is called *surjective* if for each $b \in B$ there exists $a \in A$ such that $\varphi(a,b) = 1$, i.e., if φ^{-1} is an $\mathscr{L}$-function. For a surjective fuzzy relation $\varphi \in \mathscr{R}(A,B)$ we also say that it is a fuzzy relation of A *onto* B. If φ is an $\mathscr{L}$-function and it is surjective, i.e., if both φ and φ^{-1} are $\mathscr{L}$-functions, then φ is called a *surjective $\mathscr{L}$-function*.

Let us note that a fuzzy relation $\varphi \in \mathscr{R}(A,B)$ is an $\mathscr{L}$-function if and only if there exists a function $\psi : A \to B$ such that $\varphi(a,\psi(a)) = 1$, for all $a \in A$. A function ψ with this property we will call a crisp description of φ, and we will denote by $CR(\varphi)$ the set of all such functions.

An $\mathscr{L}$-function which is a partial fuzzy function with respect to E and F is called a *perfect fuzzy function* with respect to E and F. Perfect fuzzy functions were introduced and studied by Demirci [34, 35]. A fuzzy relation $\varphi \in \mathscr{R}(A,B)$ which is a perfect fuzzy function with respect to E_A^{φ} and E_B^{φ} will be called just a *perfect fuzzy function*.

Let A and B be non-empty sets and let $\varphi \in \mathscr{R}(A,B)$ be a partial fuzzy function. If, in addition, φ is a surjective $\mathscr{L}$-function, then it will be called a *uniform fuzzy relation*. In other words, a uniform fuzzy relation is a perfect fuzzy function having the additional property that it is surjective.

Next, we recall the characterizations of uniform fuzzy relation from [20] which will be useful in our further work.

Theorem 1.9. *Let A and B be non-empty sets and let $\varphi \in \mathscr{R}(A,B)$ be a fuzzy relation. Then the following conditions are equivalent:*

(1) *φ is a uniform fuzzy relation;*
(2) *φ^{-1} is a uniform fuzzy relation;*

(3) *φ is a surjective $\mathscr{L}$-function and $\varphi \circ \varphi^{-1} \circ \varphi = \varphi$;*
(4) *φ is a surjective $\mathscr{L}$-function and $E_A^{\varphi} = \varphi \circ \varphi^{-1}$;*
(5) *φ is a surjective $\mathscr{L}$-function and $E_B^{\varphi} = \varphi^{-1} \circ \varphi$;*
(6) *φ is an $\mathscr{L}$-function, and for all $\psi \in CR(\varphi)$, $a \in A$ and $b \in B$ we have that:*

$$\psi \text{ is } E_B^{\varphi}\text{-surjective and } \varphi(a,b) = E_B^{\varphi}(\psi(a),b);$$

(7) *φ is an $\mathscr{L}$-function, and for all $\psi \in CR(\varphi)$, $a_1,a_2 \in A$ we have that:*

$$\psi \text{ is } E_B^{\varphi}\text{-surjective and } \varphi(a_1,\psi(a_2)) = E_A^{\varphi}(a_1,a_2).$$

Corollary 1.1. *Let A and B be non-empty sets and let $\varphi \in \mathscr{R}(A,B)$ be a uniform fuzzy relation. Then for all $\psi \in CR(\varphi)$ and $a_1,a_2 \in A$ we have that*

$$E_A^{\varphi}(a_1,a_2) = E_B^{\varphi}(\psi(a_1),\psi(a_2)) \tag{1.50}$$

Let A and B be non-empty sets. According to Theorem 1.9, a fuzzy relation $\varphi \in \mathscr{R}(A,B)$ is a uniform fuzzy relation if and only if its inverse relation φ^{-1} is a uniform fuzzy relation. Further, from conditions (4) and (5) of the same theorem, we have that the kernel of φ^{-1} is the co-kernel of φ and conversely, the co-kernel of φ^{-1} is the kernel of φ, that is

$$E_B^{\varphi^{-1}} = E_B^{\varphi} \text{ and } E_A^{\varphi^{-1}} = E_A^{\varphi}$$

The following theorems will be very useful in our further work.

Theorem 1.10. *Let A and B be non-empty sets, and let $\varphi \in \mathscr{R}(A,B)$ be a uniform fuzzy relation, let $E = E_A^{\varphi}$ and $F = E_B^{\varphi}$, and let $\widetilde{\varphi} : A/E \to B/F$ be the function given by*

$$\widetilde{\varphi}(E_a) = F_{\psi(a)}, \text{ for any } a \in A \text{ and } \psi \in CR(\varphi). \tag{1.51}$$

Then $\widetilde{\varphi}$ is a well-defined function (it does not depend on the choice of $\psi \in CR(\varphi)$ and $a \in A$), it is a bijective function of A/E onto B/F and $(\widetilde{\varphi})^{-1} = \widetilde{\varphi^{-1}}$.

Theorem 1.11. *Let A and B be non-empty sets, and let $\varphi_1,\varphi_2 \in \mathscr{R}(A,B)$ be uniform fuzzy relations. Then the following conditions are equivalent:*

(1) $\varphi_1 \leqslant \varphi_2$;
(2) $\varphi_1^{-1} \leqslant \varphi_2^{-1}$;
(3) $CR(\varphi_1) \subseteq CR(\varphi_2)$ *and* $E_A^{\varphi_1} \leqslant E_A^{\varphi_2}$;
(4) $CR(\varphi_1) \subseteq CR(\varphi_2)$ *and* $E_B^{\varphi_1} \leqslant E_B^{\varphi_2}$.

As a direct consequence of the previous theorem we obtain the following corollary which shows that a uniform fuzzy relation is uniquely determined by its crisp representation and kernel, as well as by its crisp representation and co-kernel.

Lemma 1.6. *Let A and B be non-empty sets, and let $\varphi_1, \varphi_2 \in \mathscr{R}(A,B)$ be uniform fuzzy relations. Then the following conditions are equivalent:*

(1) $\varphi_1 = \varphi_2$;
(2) $\varphi_1^{-1} = \varphi_2^{-1}$;
(3) $CR(\varphi_1) = CR(\varphi_2)$ *and* $E_A^{\varphi_1} = E_A^{\varphi_2}$;
(4) $CR(\varphi_1) = CR(\varphi_2)$ *and* $E_B^{\varphi_1} = E_B^{\varphi_2}$.

The composition of two uniform fuzzy relations need not be a uniform fuzzy relation. However, if the co-kernel of the first fact or of the composition is contained in the kernel of the second factor, then the composition is uniform, as the following theorem shows.

Theorem 1.12. *Let A, B and C be non-empty sets, and let $\varphi_1 \in \mathscr{R}(A,B)$ and $\varphi_2 \in \mathscr{R}(B,C)$.*

(1) *If φ_1 and φ_2 are surjective $\mathscr{L}$-functions, then $\varphi_1 \circ \varphi_2$is also a surjective $\mathscr{L}$-function.*
(2) *If φ_1 and φ_2 are uniform fuzzy relations such that $E_B^{\varphi_1} \leqslant E_B^{\varphi_2}$, then $\varphi_1 \circ \varphi_2$ is also a uniform fuzzy relation.*

Chapter 2
Fundamentals of fuzzy automata

Study of fuzzy automata and languages was initiated in 1960s by Santos [110, 111, 113], Wee [122], Wee and Fu [123], and Lee and Zadeh [80]. From late 1960s until early 2000s mainly fuzzy automata and languages with membership values in the Gödel structure have been considered (cf., e.g., [38, 45, 84]). The idea of studying fuzzy automata with membership values in some structured abstract set comes back to Wechler [121]. Nowadays, fuzzy automata over complete residuated lattices, lattice-ordered monoids, and other kinds of lattices are extensively studied. Fuzzy automata taking membership values in a complete residuated lattice were first introduced by Qiu in [96, 97], where some basic concepts were discussed, and later, Qiu and his coworkers have carried out comprehensive research of these fuzzy automata (cf. [98, 100, 124, 125, 126, 127, 128]). From a different point of view, fuzzy automata over a complete residuated lattice were studied by Ignjatović, Ćirić and their coworkers in [22, 26, 27, 52, 53, 54, 56, 116].

Bisimulation is a binary relation between state transition systems (e.g. automata), connecting systems which behave in the same way in the sense that one system simulates the other and vice-versa. Bisimulations have two main roles. First, they are used to model the equivalence between states of two different automata and approximate the language-equivalence. As we have already said, the problem of determining whether two given automata are language-equivalent is computationally hard, but we will see that the problem of determining whether two given automata are Şbisimulation equivalentŤ, i.e., whether there is a bisimulation between them, is solvable in polynomial time. Hence, bisimulations are generally considered as the best way to model the language-equivalence, because they give a close enough approximation of the language-equivalence and are efficiently computable.

Bisimulations are also used to reduce the number of states of automata. To illustrate the role of the state reduction, note that in real-life applications of automata we typically start from an ordinary or fuzzy regular expression, which is then converted to a nondeterministic or fuzzy finite automaton (cf. [76, 116]). However, the practical implementation usually requires a deterministic finite automaton or a deterministic fuzzy finite automaton, and the obtained nondeterministic or fuzzy automaton has to be determinized. On the other hand, determinization can cause an exponential blow up in the number of states, and in the case of fuzzy finite automata over certain structures of membership values (such as the product structure), determinization can even result in an infinite automaton (cf. [19, 53, 56, 65, 76]). That is why the number of states of a fuzzy finite automaton has to be reduced prior to determinization.

Another important example that illustrates the significance of the state reduction is modeling of discrete event systems. A discrete event system is a dynamical system whose state space is described by a discrete set, and states evolve as a result of asynchronously occurring discrete events over time [24, 51]. Such systems have significant applications in many fields of computer science and engineering, such as concurrent and distributed software systems, computer and communication networks, manufacturing, transportation and traffic control systems, etc. In many situations states and state transitions, as well as control strategies, are somewhat imprecise, uncertain and vague. To take this kind of uncertainty into account, Lin and Ying extended classical discrete event systems to *fuzzy discrete event systems* by proposing a fuzzy finite automaton model [78, 79]. Fuzzy discrete event systems have been studied in [29, 30, 31, 68, 78, 79, 72, 75, 99, 101], and they have been successfully applied to biomedical control for HIV/AIDS treatment planning, robotic control, intelligent vehicle control, waste-water treatment, examination of chemical reactions, and in other fields.

Usually, a discrete event system is modeled by a deterministic or nondeterministic finite automaton, and recently by a fuzzy finite automaton, with events modeled by input letters, and the behavior of a discrete event system is described by the language or fuzzy language generated by the automaton. Discrete event models of complex dynamical systems are built rarely in a monolithic manner. Instead, a modular approach is used where models of individual components are built first, followed by the composition of these models to obtain the model of the overall system. In the automaton modeling formalism the composition of individual automata (that model interacting system components) is typically formalized by the parallel composition of automata. Once a complete system model has been obtained by parallel composition of a set of automata, the resulting monolithic model can be used to analyze the properties of the system, such as safety properties, blocking properties, observability, diagnosability, controllability, etc. (cf. [24, 51]). The main problem that may arise here is that the size of the state set of the parallel composition may in the worst case grow exponentially in the number of automata that are composed. This process is known as the *curse of dimensionality* in the study of complex systems composed of many interacting components. The mentioned problem may be mitigated if we adopt modular reasoning, which can make it possible to replace components in the parallel composition by smaller equivalent automata that are obtained by the state reduction of the components, and then to analyze a simpler system. For example, such an approach has been applied in [117] in conflict analysis of fuzzy discrete event systems.

The most common structures on which bisimulations have been studied are labelled transition systems, but they have also been investigated in the context of deterministic, nondeterministic, weighted, probabilistic, timed and hybrid automata. Recently, bisimulations have been discussed in the context of fuzzy automata in [18, 22, 26, 27, 54, 94, 117, 118].In the study of bisimulation for fuzzy automata, two general approaches can be destinguish. The first approach, uses ordinary crisp relations and functions [18, 94, 118]. Another approach, proposed in [26, 27, 54, 117], is based on the use of fuzzy relations, which have been shown to provide better results both in the state reduction and the modeling of equivalence of fuzzy automata. The same approach has been used in [22], in the study of simulations and bisimulations between fuzzy automata, where simulations and bisimulations have also been defined as fuzzy relations. There have been introduced two types of simulations (forward and backward simulations) and four types of bisimulations (forward, backward, forward-backward, and backward-forward bisimulations).

In this chapter fuzzy automata with membership values in complete residuated lattices will be considered. In the first section the notion of fuzzy automata and fuzzy language will be stated. Also the definition of the left (right) derivative of a fuzzy language and the definition of the minimal automaton of a fuzzy language will be given. Further, in Section 2.2., the crisp-deterministic fuzzy automata will be discussed. As an significant member of the class of all crisp-deterministic fuzzy automata, the Nerode fuzzy automaton will be presented. Moreover, the minimal crisp-deterministic fuzzy automaton of fuzzy language will be defined and the derivative automaton as the representative of this class will be discussed. In the Section 2.3. we will present the factor, afterset and foreset fuzzy automata as well as their main features. At the and of this chapter, simulations and bisimulations, introduced in [22], will be observed.

The results in this section are presented according to the results given by M. Ćirić et all. in [56, 22, 117].

2.1. Fuzzy automata

Without loss of generality, in further text we suppose that $\mathscr{L}$ is a complete residuated lattice and that X is an (finite) alphabet.
A *fuzzy transition system over* $\mathscr{L}$ and X is a pair $\mathscr{A} = (A, \delta^A)$, where:

- A is a non-empty set, called the *set of states*;
- $\delta^A : A \times X \times A \to L$ is a fuzzy subset of $A \times X \times A$, called the *fuzzy transition function*.

A *fuzzy automaton over* $\mathscr{L}$ and X , or simply a *fuzzy automaton*, is a quadruple $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$, where:

- A is a non-empty set, called the *set of states;*
- $\delta^A : A \times X \times A \to L$ is a fuzzy subset of $A \times X \times A$, called the *fuzzy transition function;*
- $\sigma^A : A \to L$ is the fuzzy subset of A, called the *fuzzy set of initial states;*
- $\tau^A : A \to L$ is the fuzzy subset of A, called the *fuzzy set of terminal states*.

In cases where it is clear which set of states is underlying for δ^A, σ^A and τ^A, the superscript A will be omitted, i.e., instead of $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ we will write simply$\mathscr{A} = (A, \delta, \sigma, \tau)$.

We can interpret $\delta(a, x, b)$ as the degree to which an input letter $x \in X$ causes a transition from a state $a \in A$ into a state $b \in A$, whereas we can interpret $\sigma(a)$ and $\tau(a)$ as the degrees to which a is respectively an input state and a terminal state. For methodological reasons we sometimes allow the set of states A to be infinite. A fuzzy automaton whose set of states is finite is called a *fuzzy finite automaton*.

Let X^* denote the free monoid over the alphabet X, and let $e \in X^*$ be the empty word. The function δ can be extended up to a function $\delta^* : A \times X^* \times A \to L$ as follows: If $a, b \in A$, then

$$\delta^*(a, e, b) = \begin{cases} 1, & \text{if } a = b, \\ 0, & \text{otherwise}, \end{cases} \tag{2.1}$$

and if $a, b \in A$, $u \in X^*$ and $x \in X$, then

$$\delta^*(a, ux, b) = \bigvee_{c \in A} \delta^*(a, u, c) \otimes \delta(c, x, b). \tag{2.2}$$

By (1.22) and Theorem 3.1 [76] (see also [96, 100]), we have that

$$\delta^*(a, uv, b) = \bigvee_{c \in A} \delta^*(a, u, c) \otimes \delta^*(c, v, b), \tag{2.3}$$

for all $a, b \in A$ and $u, v \in X^*$. In other words, if $w = x_1 x_2 \cdots x_n$, such that $x_1, x_2, .., x_n \in X$, then

$$\delta^*(a, w, b) = \bigvee_{c_1, .., c_{n-1} \in A^{n-1}} \delta(a, x_1, c_1) \otimes \delta(c_1, x_2, c_2) \otimes \cdots \otimes \delta(c_{n-1}, x_n, b). \tag{2.4}$$

If for any $u \in X^*$ we define a fuzzy relation δ_u on A by

$$\delta_u(a, b) = \delta_*(a, u, b), \tag{2.5}$$

for all $a, b \in A$, called the *fuzzy transition relation* determined by u, then (2.3) can be written as

$$\delta_{uv} = \delta_u \circ \delta_v, \tag{2.6}$$

for all $u,v \in X^*$. Equality (2.6) means that with respect to the composition of fuzzy relations, the fuzzy transition relation is a semigroup. This semigroup will be called the semigroup of transition of fuzzy automaton $\mathscr{A}$.

Let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton. Then $\widehat{\delta}$, the crisp part of δ, is a crisp subset of $A\times X\times A$, and $\widehat{\sigma}$ and $\widehat{\tau}$, crisp subsets of σ and τ, respectively, are ordinary subsets of A. The automaton $\widehat{\mathscr{A}} = (A,\widehat{\delta},\widehat{\sigma},\widehat{\tau})$ is a nondeterministic automaton which is called a *crisp part of* $\mathscr{A}$.

If δ is a crisp subset of $A\times X\times A$, that is, $\delta : A\times X\times A \to \{0,1\}$, and σ and τ are crisp subsets of A, then $\mathscr{A}$ is an ordinary *nondeterministic automaton*, which means that the notion of fuzzy automaton generalize the notion of nondeterministic automaton. Precisely, nondeterministic automata are fuzzy automata over the Boolean structure. They will also be called *Boolean automata*. Clearly, the extension of the mapping δ is a crisp subset of $A\times X^*\times A$.

If δ is a mapping from $A\times X \to A$, then $\mathscr{A}$ is ordinary deterministic automaton. Also, extension of the mapping δ is mapping from $A\times X^*$ to A. For the deterministic automaton $\mathscr{A} = (A,\delta,\sigma,\tau)$, $a\in A$ and $u\in X^*$ the state $\delta(a,u)$ will be denoted by au.

The *reverse fuzzy automaton* of a fuzzy automaton $\mathscr{A} = (A,\delta,\sigma,\tau)$ is defined as the fuzzy automaton $\bar{\mathscr{A}} = (A,\bar{\delta},\bar{\sigma},\bar{\tau})$ whose fuzzy transition function is defined by :

$$\bar{\delta}(a_1,x,a_2) = \delta(a_2,x,a_1) \quad \text{for all } a_1,a_2 \in A, \quad x\in X,$$

and fuzzy sets of initial and terminal states are $\bar{\sigma} = \tau$ and $\bar{\tau} = \sigma$.

Fuzzy automata $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{A}' = (A',\delta^{A'},\sigma^{A'},\tau^{A'})$ are *isomorphic* if there is a bijective function $\phi : A \to A'$ such that $\delta^A_x(a,b) = \delta^{A'}_x(\phi(a),\phi(b))$, for all $a,b\in A$ and $x\in X$, and also, $\sigma^A(a) = \sigma^{A'}(\phi(a))$ and $\tau^A(a) = \tau^{A'}(\phi(a))$, for every $a\in A$. It is easy to check that in this case we also have that

$$\delta^A_u(a,b) = \delta^{A'}_u(\phi(a),\phi(b)),\ \text{for all } a,b\in A \text{ and } u\in X^*. \tag{2.7}$$

A *fuzzy language* in X^* over $\mathscr{L}$, or briefly a *fuzzy language*, is any fuzzy subset of X^*, i.e., any function from X^* into L. A *fuzzy language recognized by a fuzzy automaton* $\mathscr{A} = (A,\delta,\sigma,\tau)$, denoted as $[[\mathscr{A}]]$, is a fuzzy language in $\mathscr{F}(X^*)$ defined by

$$\begin{aligned} [[\mathscr{A}]](e) &= \sigma^A \circ \tau^A, \\ [[\mathscr{A}]](u) &= \sigma^A \circ \delta^A_{x_1} \circ \delta^A_{x_2} \circ \cdots \circ \delta^A_{x_n} \circ \tau^A, \end{aligned} \tag{2.8}$$

for any $u = x_1x_2\dots x_n \in X^+$, where $x_1,x_2,\dots,x_n\in X$. In other words, the equality (2.8) means that the membership degree of the word u to the fuzzy language $[[\mathscr{A}]]$ is equal to the degree to which $\mathscr{A}$ recognizes or accepts the word u. Using notation from (1.39), and the second equality in (1.41), we can state

(2.8) as

$$[\![\mathscr{A}]\!](u) = \sigma \circ \delta_u \circ \tau. \tag{2.9}$$

Fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ are called *language equivalent*, or sometimes just *equivalent*, if $[\![\mathscr{A}]\!] = [\![\mathscr{B}]\!]$.

Cardinality of a fuzzy automaton $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$, in notation $|\mathscr{A}|$, is defined as the cardinality of its set of states A. A fuzzy automaton $\mathscr{A}$ is called *minimal fuzzy automaton* of a language $f \in \mathscr{F}(X^*)$ if it recognizes f and $|\mathscr{A}| < |\mathscr{A}'|$, for any fuzzy automaton $\mathscr{A}'$ recognizing f. A minimal fuzzy automaton recognizing a given fuzzy language f is not necessarily unique up to an isomorphism. This is also true for nondeterministic automata.

For more information on fuzzy automata over complete residuated lattices we refer to [27, 117, 52, 53, 54, 96, 100, 124, 127, 56]

2.2. Crisp-deterministic fuzzy automata

Let $\mathscr{A} = (A, \delta, \sigma, \tau)$ be a fuzzy automaton over X and $\mathscr{L}$. The fuzzy transition function δ is called *crisp-deterministic* if for every $x \in X$ and every $a \in A$ there exists $a' \in A$ such that $\delta_x(a, a') = 1$, and $\delta_x(a, b) = 0$, for all $b \in A \setminus \{a'\}$. The fuzzy set of initial states σ is called *crisp-deterministic* if there exists $a_0 \in A$ such that $\sigma(a_0) = 1$ and $\sigma(a) = 0$ for every $a \in A \setminus \{a_0\}$. If both σ and δ are crisp-deterministic, then $\mathscr{A}$ is called a *crisp-deterministic fuzzy finite automaton* (for short: *cdffa*).

Equivalently, we can define a crisp-deterministic fuzzy finite automaton over X and $\mathscr{L}$ as a quadruple $\mathscr{A} = (A, \delta, a_0, \tau)$, where A is a non-empty *set of states*, $\delta : A \times X \to A$ is a *transition function*, $a_0 \in A$ is an *initial state* and $\tau : A \to L$ is a *fuzzy set of final states*.

The transition function δ can be extended to a function $\delta^* : A \times X^* \to A$ as follows: $\delta^*(a, \varepsilon) = a$, for each $a \in A$, and $\delta^*(a, ux) = \delta(\delta^*(a, u), x)$, for every $a \in A$, $u \in X^*$ and $x \in X$. Also, we allow the set A to be infinite, and then $\mathscr{A}$ is called a *crisp-deterministic fuzzy automaton* (for short: *cdfa*). The *language* of $\mathscr{A}$ is the fuzzy language $[\![\mathscr{A}]\!] : X^* \to L$ defined by

$$[\![\mathscr{A}]\!](u) = \tau(\delta^*(a_0, u)) \ . \tag{2.10}$$

for every $u \in X^*$. Obviously, the image of $[\![\mathscr{A}]\!]$ is contained in the image of τ which is finite if the set of states A is finite.

A fuzzy language $\varphi : X^* \to L$ is called *cdffa-recognizable* if there exists a crisp-deterministic fuzzy finite automaton $\mathscr{A}$ over X and $\mathscr{L}$ such that $[\![\mathscr{A}]\!] = \varphi$. We also say that $\mathscr{A}$ *recognizes* φ.

Next, the *Nerode automaton* of a fuzzy automaton $\mathscr{A} = (A, \delta, \sigma, \tau)$ is a crisp-deterministic fuzzy automaton $\mathscr{A}_N = (A_N, \delta_N, \sigma^A_e, \tau_N)$, such that $A_N = \{\sigma^A_u | u \in X^*\}$ where $\sigma^A_u = \sigma^A \circ \delta^A_u$,for every $u \in X^*$ and $\delta_N : A_N \times X \longrightarrow A_N$ and $\tau_N \in \mathscr{F}(A_N)$ are defined with

$$\delta_N(\sigma_u^A, x) = \sigma_{ux}^A, \qquad \tau_N(\sigma_u^A) = \sigma_u^A \circ \tau^A,$$

for every $u \in X^*$ and $x \in X$. The automaton $\mathscr{A}_N$ is language equivalent to $\mathscr{A}$. The notion of Nerode automaton of a given fuzzy automaton was first introduced by Ignjatović, Ćirić, Bogdanović and Petković in [56], for fuzzy automata over complete residuated lattices and lattice ordered monoids.

A crisp-deterministic fuzzy automaton $\mathscr{A}$ is called *minimal crisp-deterministic fuzzy automaton* of a language $f \in \mathscr{F}(X^*)$ if it recognizes f and $|\mathscr{A}| < |\mathscr{A}'|$, for any crisp-deterministic fuzzy automaton $\mathscr{A}'$ recognizing f.

Given a fuzzy language $\varphi : X^* \to L$ and $v \in X^*$, we define the fuzzy language $v^{-1}\varphi : X^* \to L$ and $\varphi v^{-1} : X^* \to L$ by letting $v^{-1}\varphi(u) = \varphi(vu)$ and $\varphi v^{-1}(u) = \varphi(uv)$ for $u \in X^*$. The fuzzy language $v^{-1}\varphi$ is called a *left derivative*, and the fuzzy language φv^{-1} a *right derivative* of φ with respect to v.

For a fuzzy language $\varphi : X^* \to L$, let $A_\varphi = \{u^{-1}\varphi \mid u \in X^*\}$ denote the set of all left derivatives of φ, and let $\delta_\varphi : A_\varphi \times X \to A_\varphi$ and $\tau_\varphi : A_\varphi \to L$ be mappings defined by

$$\delta_\varphi(\psi, x) = x^{-1}\psi \quad \text{and} \quad \tau_\varphi(\psi) = \psi(\varepsilon), \tag{2.11}$$

for every $\psi \in A_\varphi$ and $x \in X$. Then $\mathscr{A}_\varphi = (A_\varphi, \delta_\varphi, \varphi, \tau_\varphi)$ is an accessible cdfa, and it is called the *derivative automaton* of the fuzzy language φ [56, 52]. It has been proved in [56] that the derivative automaton $\mathscr{A}_\varphi$ is finite if and only if the fuzzy language φ is cdffa-recognizable. Namely, $\mathscr{A}_\varphi$ is a minimal cdfa which recognizes φ [56]. An algorithm for construction of the derivative automaton of a fuzzy language, has been also given in [56].

2.3. Factor and afterset fuzzy automata

Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be a fuzzy automaton and let E a fuzzy equivalence relation on A. The *factor fuzzy automaton* of $\mathscr{A}$ with respect to E is the fuzzy automaton $\mathscr{A}/E = (A/E, \delta^{A/E}, \sigma^{A/E}, \tau^{A/E})$, where: the set of state is $A/E = \{E_a \mid a \in A\}$, the fuzzy transition relation $\delta^{A/E} : A/E \times X \times A/E \to L$ is defined by:

$$\delta^{A/E}(E_a, x, E_b) = \bigvee_{a', b' \in A} E(a, a') \otimes \delta^A(a', x, b') \otimes E(b', b) = E_a \circ \delta_x^A \circ E_b,$$

for every $E_a, E_b \in A/E$, and $x \in X$, the fuzzy set $\sigma^{A/E} \in \mathscr{F}(A)$ of initial states is defined by:

$$\sigma^{A/E}(E_a) = \sigma^A \circ E_a, \text{ for every } E_a \in A/E,$$

and the fuzzy set $\tau^{A/E} \in \mathscr{F}(A)$ of terminal states by:

$$\tau^{A/E}(E_a) = E_a \circ \tau^A, \text{ for every } E_a \in A/E.$$

The fuzzy language $[\![\mathscr{A}/E]\!]$ recognized by the factor fuzzy automaton $\mathscr{A}/E$ is given by

$$[\![\mathscr{A}/E]\!](e) = \sigma^A \circ E \circ \tau^A, \tag{2.12}$$
$$[\![\mathscr{A}/E]\!](u) = \sigma \circ E \circ \delta_{x_1} \circ E \circ \delta_{x_2} \circ E \cdots \circ E \circ \delta_{x_n} \circ E \circ \tau, \tag{2.13}$$

for $u = x_1x_2 \dots x_n \in X^+$, where $x_1, x_2, \dots, x_n \in X$.

Here, we recall the results from [117], concerning factor fuzzy automata, which will be useful in the further work.

Theorem 2.1. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be an automaton, and let E and F be fuzzy equivalences on A such that $E \leqslant F$. Then a relation $F/E \in \mathscr{F}(A/E)$ defined by:*

$$F/E(E_a, E_b) = F(a,b), \quad E_a, E_b \in A/E. \tag{2.14}$$

is a fuzzy equivalence on A/E, and the factor fuzzy automata $(\mathscr{A}/E)/(F/E)$ and $\mathscr{A}/F$ are isomorphic.

Theorem 2.2. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be a fuzzy automaton and E a fuzzy equivalence on A.*

The function $\Phi : \mathscr{E}_E(A) \to \mathscr{E}(A/E)$, where $\mathscr{E}_E(A) = \{F \in \mathscr{E}(A) \mid E \leqslant F\}$, defined by

$$\Phi(F) = F/E, \quad \text{for every } F \in \mathscr{E}_E(A), \tag{2.15}$$

is a lattice isomorphism, i.e., it is surjective and

$$F \leqslant G \iff \Phi(F) \leqslant \Phi(G), \quad \text{for all } F, G \in \mathscr{E}_E(A). \tag{2.16}$$

For fuzzy automaton $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and fuzzy quasi-order R on A, the fuzzy automaton $\mathscr{A}/R = (A/R, \delta^{A/R}, \sigma^{A/R}, \tau^{A/R})$ where the set of states $A/R = \{R_a \mid a \in A\}$, the fuzzy transition function $\delta^{A/R} : A/R \times X \times A/R \to L$ is given by

$$\delta^{A/R}(R_a, x, R_b) = \bigvee_{a',b' \in A} R(a,a') \otimes \delta^A(a', x, b') \otimes R(b', b), \tag{2.17}$$

or equivalently

$$\delta^{A/R}(R_a, x, R_b) = (R \circ \delta^A_x \circ R)(a,b) = R_a \circ \delta^A_x \circ R^b, \tag{2.18}$$

for all $R_a, R_b \in A/R$ and $x \in X$, the fuzzy set $\sigma^{A/R} \in L^{A/R}$ of initial states is defined by

$$\sigma^{A/R}(R_a) = \bigvee_{a' \in A} \sigma^A(a') \otimes R(a', a) = (\sigma^A \circ R)(a) = \sigma^A \circ R^a, \quad a \in A; \tag{2.19}$$

the fuzzy set $\tau^{A/R} \in L^{A/R}$ of terminal states is defined by

$$\tau^{A/R}(R_a) = \bigvee_{a'\in A} R(a,a')\otimes\tau^A(a') = (R\circ\tau^A)(a) = R_a\circ\tau^A, \quad a\in A, \tag{2.20}$$

is called the *afterset fuzzy automaton* of $\mathscr{A}$ w.r.t. R.

The fuzzy language $[\![\mathscr{A}/R]\!]$ recognized by the afterset fuzzy automaton $\mathscr{A}/R$ is given by

$$\begin{aligned} &[\![\mathscr{A}/R]\!](e) = \sigma^A\circ R\circ\tau^A, \\ &[\![\mathscr{A}/R]\!](u) = \sigma^A\circ R\circ\delta^A_{x_1}\circ R\circ\delta^A_{x_2}\circ R\circ\cdots\circ R\circ\delta^A_{x_n}\circ R\circ\tau^A, \end{aligned} \tag{2.21}$$

for every $u = x_1x_2\dots x_n\in X^*$, where $x_1,x_2,\dots,x_n\in X$.

Let us note that previous equation follows directly from the definition of the afterset automaton $\mathscr{A}/R$ and the fact that $R\circ R = R$ for every fuzzy quasi-order R.

Analogously, for a fuzzy automaton $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$, the *foreset fuzzy automaton* of $\mathscr{A}$ w.r.t. R is a fuzzy automaton $\mathscr{A}\backslash R = (A\backslash R,\delta^{A\backslash R},\sigma^{A\backslash R},\tau^{A\backslash R})$ with:

the set of states $A\backslash R = \{R^a \mid a\in A\}$, the fuzzy transition function $\delta^{A\backslash R}$ defined by

$$\begin{aligned} \delta^{A\backslash R}(R^a,x,R^b) &= \bigvee_{a',b'\in A} R(a,a')\otimes\delta^A(a',x,b')\otimes R(b',b) \\ &= (R\circ\delta^A_x\circ R)(a,b) = R_a\circ\delta^A_x\circ R^b, \end{aligned} \tag{2.22}$$

for all for any $R^a,R^b\in A\backslash R$ and $x\in X$, a fuzzy set $\sigma^{A\backslash R}\in L^{A\backslash R}$ of initial states is given by

$$\sigma^{A\backslash R}(R^a) = \bigvee_{a'\in A}\sigma^A(a')\otimes R(a',a) = (\sigma^A\circ R)(a) = \sigma^A\circ R^a, \quad R^a\in A\backslash R, \tag{2.23}$$

and the fuzzy set $\tau^{A\backslash R}\in L^{A\backslash R}$ of terminal states is given by

$$\tau^{A\backslash R}(R^a) = \bigvee_{a'\in A} R(a,a')\otimes\tau^A(a') = (R\circ\tau^A)(a) = R_a\circ\tau^A, \quad R^a\in A\backslash R. \tag{2.24}$$

It can easily be shown that:

Theorem 2.3. *For any fuzzy quasi-order R on a fuzzy automaton $\mathscr{A}$ the afterset fuzzy automaton $\mathscr{A}/R$ and the foreset fuzzy automaton $\mathscr{A}\backslash R$ are isomorphic.*

In view of the previous theorem, in the rest of this section we will consider only afterset fuzzy automata. We will see in the following example that the factor fuzzy automata $\mathscr{A}/E_R$ of $\mathscr{A}$, w.r.t. the natural fuzzy equivalence E_R of R, is not necessary isomorphic to fuzzy recognizers $\mathscr{A}/R$ and $\mathscr{A}\backslash R$, but by (b) of Theorem 1.7, it has the same cardinality as $\mathscr{A}/R$ and $\mathscr{A}\backslash R$, and if $[\![\mathscr{A}]\!] = [\![\mathscr{A}/R]\!](= [\![\mathscr{A}\backslash R]\!])$, then we also have that $[\![\mathscr{A}]\!] = [\![\mathscr{A}/E_R]\!]$.

Example 2.1. Let $\mathscr{L}$ be the Boolean structure, and let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton over $\mathscr{L}$, where $A = \{1,2,3\}$, $X = \{x,y\}$, and fuzzy transition relations σ, δ_x, δ_y and τ are given by

$$\sigma = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \quad \delta_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \delta_y = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

The fuzzy quasi order R on A and its natural fuzzy equivalence E_R are given by

$$R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad E_R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

The afterset fuzzy automaton $\mathscr{A}/R$ is not isomorphic to the factor fuzzy automaton $\mathscr{A}/E_R$, since:

$$R \circ \delta_y \circ R = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad E_R \circ \delta_y \circ E_R = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

For a fuzzy automaton $\mathscr{A} = (A,\delta,\sigma,\tau)$ and a fuzzy quasi-order R on A we can define the fuzzy automaton $\mathscr{A}|R = (A,\sigma^{A|R},\delta^{A|R},\tau^{A|R})$, where the set of states, the input alphabet, the fuzzy set of initial states and the fuzzy set of final states are the same as in the original automaton, and the fuzzy transition function is defined as follows:

$$\delta^{A|R}(a,x,b) = (R \circ \delta_x \circ R)(a,b), \quad \text{for all } a,b \in A \text{ and } x \in X.$$

The following theorem is a result from [117] and it presents a version of the well-known Second Isomorphism Theorem, concerning fuzzy automata and fuzzy quasi-orders on them.

Theorem 2.4. *Let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton and let R and S be fuzzy quasi-orders on A such that $R < S$. Then a fuzzy relation S/R on $\mathscr{A}/R$ defined by*

$$S/R(R_a,R_b) = S(a,b), \text{ for all } a,b \in A, \tag{2.25}$$

is a fuzzy quasi-order on $\mathscr{A}/R$ and fuzzy recognizers $\mathscr{A}/S$, $(\mathscr{A}/R)/(S/R)$ and $(\mathscr{A}/R)/S$ are isomorphic.

Let us note that if $\mathscr{A}$ is a fuzzy automaton, A is its set of states, and R, S and T are fuzzy quasi-orders on A such that $R \leqslant S$ and $R \leqslant T$, then

$$S \leqslant T \Leftrightarrow S/R \leqslant T/R, \tag{2.26}$$

and hence, a mapping $\Phi : \mathscr{Q}_R(A) = \{S \in \mathscr{Q}(A) | R \leqslant S\} \to \mathscr{Q}(A/R)$, given by $\Phi : S \to S/R$, is injective (in fact, it is an order isomorphism of $\mathscr{Q}_R(A)$ onto a subset of $\mathcal{Q}(A/R)$). In particular, for a fuzzy quasi-order R on A, the fuzzy relation R/R on A/R will be denoted by $\tilde{R}$. It can be easily verified that $\tilde{R}$ is a fuzzy quasi-order on A/R, and if E is a fuzzy equivalence on A, then $\tilde{E}$ is a fuzzy equality on A/E.

As we maintained in Section 2.1., for a given fuzzy automaton $\mathscr{A} = (A,\delta,\sigma,\tau)$ over X and $\mathscr{L}$ the fuzzy language recognized by $\mathscr{A}$ is:

$$[[\mathscr{A}]](e) = \sigma^A \circ \tau^A,$$
$$[[\mathscr{A}]](u) = \sigma^A \circ \delta^A_{x_1} \circ \delta^A_{x_2} \circ \cdots \circ \delta^A_{x_n} \circ \tau^A,$$

for any $u = x_1x_2\cdots x_n \in X^*$.

According to this and equality (2.21), we obtain that the fuzzy automaton $\mathscr{A}$ and the afterset automaton $\mathscr{A}/R$ are equivalent, i.e. they recognize the same language if and only if the fuzzy quasi-order R is a solution to a system of fuzzy relation equations:

$$\begin{aligned}&\sigma^A \circ \tau^A = \sigma^A \circ R \circ \tau^A,\\ &\sigma^A \circ \delta^A_{x_1} \circ \delta^A_{x_2} \circ \cdots \circ \delta^A_{x_n} \circ \tau^A = \sigma^A \circ R \circ \delta^A_{x_1} \circ R \circ \delta^A_{x_2} \circ R \circ \cdots \circ R \circ \delta^A_{x_n} \circ R \circ \tau^A,\end{aligned} \tag{2.27}$$

for all $n \in N$ and $x_1, x_2, \cdots, x_n \in X$.

The general system has at least one solution in $\mathcal{Q}(A)$, the equality relation on A. It will be called the trivial solution. To attain the best possible reduction of A , we have to find the greatest solution to the general system in $\mathcal{Q}(A)$, if it exists, or to find as big solution as possible. However, the general system does not necessary have the greatest solution and also, it may consist of infinitely many equations, and finding its nontrivial solutions may be a very difficult task. For that reason we will aim our attention to some instances of the general system. These instances have to be as general as possible, but they have to be easier to solve. From a practical point of view, these instances have to consist of finitely many equations. The next theorem shows some properties of the set of all solutions to the general system.

Theorem 2.5. *Let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton.*

The set of all solutions to the general system in $\mathcal{Q}(A)$ is an order ideal of the lattice $\mathcal{Q}(A)$.

Consequently, if a fuzzy quasi-order R on A is a solution to the general system, then its natural fuzzy equivalence E_R is also a solution to the general system.

The following example shows that there are fuzzy quasi-orders which are not solutions to the general system, but their natural fuzzy equivalences are solutions to this system.

Example 2.2. Let $\mathscr{L}$ be the Boolean structure, let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ be a fuzzy automaton over $\mathscr{L}$, where $A = \{1,2,3\}$, $X = \{x,y\}$, and δ^A_x, δ^A_y, σ^A and τ^A are given by

$$\delta_x^A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \delta_y^A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad \sigma^A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix},$$

and consider a fuzzy quasi-order R on A given by

$$R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then we have that

$$\sigma^A \circ R \circ \delta_x^A \circ R \circ \delta_y^A \circ R \circ \tau^A = 1 \neq 0 = \sigma^A \circ \delta_x^A \circ \delta_y^A \circ \tau^A,$$

so R is not a solution to the general system, but its natural fuzzy equivalence E_R is the equality relation on A, and hence, it is a solution to the general system.

The next example demonstrates one shortcoming of state reductions by means of fuzzy quasi-orders and fuzzy equivalences. Namely, we show that for some fuzzy automata no reduction will result in its minimal automaton.

Example 2.3. Let $\mathscr{L}$ be the Boolean structure and $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ a fuzzy automaton over $\mathscr{L}$, where $|A| = 4$, $X = \{x\}$, and δ^A, σ^A, and τ^A are given by

$$\delta_x^A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \sigma^A = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}, \qquad \tau^A = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

It is easy to check that for each $u \in X^*$ the following is true:

$$[\![\mathscr{A}]\!](u) = \begin{cases} 0 & \text{if } u = e \text{ or } u = x^n, \text{for } n \geqslant 2, \\ 1 & \text{if } u = x, \end{cases}$$

(in fact, $\mathscr{A}$ is a nondeterministic automaton and $[\![\mathscr{A}]\!]$ is an ordinary crisp language consisting only of the letter x). If $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ is a fuzzy automaton over $\mathscr{L}$ with $|B| = 2$, and

$$\delta_x^B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad \sigma^B = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad \tau^B = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

then $\mathscr{B}$ recognizes $[\![\mathscr{A}]\!]$, and it is a minimal fuzzy automaton of $[\![\mathscr{A}]\!]$, since $[\![\mathscr{A}]\!]$ can not be recognized by a fuzzy automaton with only one state.

Consider now an arbitrary fuzzy equivalence

$$E = \begin{bmatrix} 1 & a_{12} & a_{13} & a_{14} \\ a_{12} & 1 & a_{23} & a_{24} \\ a_{13} & a_{23} & 1 & a_{34} \\ a_{14} & a_{24} & a_{34} & 1 \end{bmatrix}$$

on A, and suppose that E is a solution to the general system corresponding to the fuzzy automaton $\mathscr{A}$. We will show that E can not reduce $\mathscr{A}$ to a fuzzy automaton with two states.

First, by $\sigma^A \circ E \circ \tau^A = a_{23} \vee a_{24}$ and $\sigma^A \circ E \circ \tau^A = \sigma^A \circ \tau^A = [\![\mathscr{A}]\!](e) = 0$ it follows $a_{23} = a_{24} = 0$. Next, reflexivity and transitivity of E yield $E \circ E = E$, what implies

$$\begin{aligned} a_{12} \wedge a_{13} &= 0, & & a_{12} = 0 \text{ or } a_{13} = 0 && (2.28)\\ a_{12} \wedge a_{14} &= 0, & & a_{12} = 0 \text{ or } a_{14} = 0 && (2.29)\\ a_{13} \vee (a_{14} \wedge a_{34}) &= a_{13}, & \text{i.e.,}\quad & a_{13} = 0 \text{ implies } a_{14} = 0 \text{ or } a_{34} = 0, && (2.30)\\ a_{14} \vee (a_{13} \wedge a_{34}) &= a_{14}, & & a_{14} = 0 \text{ implies } a_{13} = 0 \text{ or } a_{34} = 0, && (2.31)\\ a_{34} \vee (a_{13} \wedge a_{14}) &= a_{34}, & & a_{34} = 0 \text{ implies } a_{13} = 0 \text{ or } a_{14} = 0. && (2.32) \end{aligned}$$

If $a_{12} = 1$, then by (2.28) and (2.29) we obtain $a_{13} = a_{14} = 0$, and hence

$$E = \begin{bmatrix} 1&1&0&0 \\ 1&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix} \quad \text{or} \quad E = \begin{bmatrix} 1&1&0&0 \\ 1&1&0&0 \\ 0&0&1&1 \\ 0&0&1&1 \end{bmatrix}.$$

However, none of these two matrices is a solution to the general system. Therefore, we conclude that $a_{12} = 0$. According to (2.30), (2.31) and (2.32), we distinguish the following five cases

$$\begin{aligned} &a_{13} = a_{14} = a_{34} = 0, \\ &a_{13} = a_{14} = 0, \quad a_{34} = 1, \\ &a_{13} = a_{34} = 0, \quad a_{14} = 1, \\ &a_{14} = a_{34} = 0, \quad a_{13} = 1, \\ &a_{13} = a_{14} = a_{34} = 1, \end{aligned}$$

and we obtain that E has one of the following forms

$$E = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix}, E = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&1 \\ 0&0&1&1 \end{bmatrix}, E = \begin{bmatrix} 1&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \\ 1&0&0&1 \end{bmatrix}, E = \begin{bmatrix} 1&0&1&0 \\ 0&1&0&0 \\ 1&0&1&0 \\ 0&0&0&1 \end{bmatrix}, E = \begin{bmatrix} 1&0&1&1 \\ 0&1&0&0 \\ 1&0&1&1 \\ 1&0&1&1 \end{bmatrix}. \qquad (2.33)$$

In the first case, E is the equality relation, and it does not provide any reduction of $\mathscr{A}$, and in the second and fourth case, it can be easily verified that E is a solution to the general system, but it reduces $\mathscr{A}$ to a fuzzy automaton with three states. Finally, in the third and fifth case, E is not a solution to the general system, since

$$\sigma^A \circ E \circ \delta_x^A \circ E \circ \delta_x^A \circ E \circ \tau_x^A = 1 \neq 0 = \sigma^A \circ \delta_x^A \circ \delta_x^A \circ \tau_x^A.$$

Therefore, any state reduction of $\mathscr{A}$ by means of fuzzy equivalences does not provide fuzzy automaton with less than three states.

2.4. Simulations and bisimulations between fuzzy automata

Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata, and let $\gamma \in \mathscr{R}(A,B)$ be a non-empty fuzzy relation. We call γ a *forward simulation* if it satisfies

$$\sigma^A \leqslant \sigma^B \circ \gamma^{-1}, \tag{fs-1}$$
$$\gamma^{-1} \circ \delta_x^A \leqslant \delta_x^B \circ \gamma^{-1}, \quad \text{for every } x \in X, \tag{fs-2}$$
$$\gamma^{-1} \circ \tau^A \leqslant \tau^B, \tag{fs-3}$$

and a *backward simulation* if

$$\tau^A \leqslant \gamma \circ \tau^B, \tag{bs-1}$$
$$\delta_x^A \circ \gamma \leqslant \gamma \circ \delta_x^B, \quad \text{for every } x \in X, \tag{bs-2}$$
$$\sigma^A \circ \gamma \leqslant \sigma^B. \tag{bs-3}$$

Furthermore, we call γ a *forward bisimulation* if both γ and γ^{-1} are forward simulations, i.e., if γ satisfies

$$\sigma^A \leqslant \sigma^B \circ \gamma^{-1}, \qquad \sigma^B \leqslant \sigma^A \circ \gamma, \tag{fb-1}$$
$$\gamma^{-1} \circ \delta_x^A \leqslant \delta_x^B \circ \gamma^{-1}, \quad \gamma \circ \delta_x^B \leqslant \delta_x^A \circ \gamma, \quad \text{for every } x \in X, \tag{fb-2}$$
$$\gamma^{-1} \circ \tau^A \leqslant \tau^B, \qquad \gamma \circ \tau^B \leqslant \tau^A, \tag{fb-3}$$

and a *backward bisimulation*, if both γ and γ^{-1} are backward simulations, i.e., if γ satisfies

$$\tau^A \leqslant \gamma \circ \tau^B, \qquad \tau^B \leqslant \gamma^{-1} \circ \tau^A, \tag{bb-1}$$
$$\delta_x^A \circ \gamma \leqslant \gamma \circ \delta_x^B, \quad \delta_x^B \circ \gamma^{-1} \leqslant \gamma^{-1} \circ \delta_x^A, \quad \text{for every } x \in X, \tag{bb-2}$$
$$\sigma^A \circ \gamma \leqslant \sigma^B, \qquad \sigma^B \circ \gamma^{-1} \leqslant \sigma^A. \tag{bb-3}$$

Also, if γ is a forward simulation and γ^{-1} is a backward simulation, i.e., if γ satisfies

$$\sigma^A \leqslant \sigma^B \circ \gamma^{-1}, \qquad \tau^B \leqslant \gamma^{-1} \circ \tau^A, \tag{fbb-1}$$

$$\gamma^{-1} \circ \delta_x^A = \delta_x^B \circ \gamma^{-1}, \qquad \text{for every } x \in X, \tag{fbb-2}$$

$$\sigma^B \circ \gamma^{-1} \leqslant \sigma^A, \qquad \gamma^{-1} \circ \tau^A \leqslant \tau^B, \tag{fbb-3}$$

then γ is called a *forward-backward bisimulation*, and if γ is a backward simulation and γ^{-1} is a forward simulation, i.e., if

$$\sigma^B \leqslant \sigma^A \circ \gamma, \qquad \tau^A \leqslant \gamma \circ \tau^B, \tag{bfb-1}$$

$$\delta_x^A \circ \gamma = \gamma \circ \delta_x^B, \qquad \text{for every } x \in X, \tag{bfb-2}$$

$$\sigma^A \circ \gamma \leqslant \sigma^B \qquad \gamma \circ \tau^B \leqslant \tau^A. \tag{bfb-3}$$

then γ is called a *backward-forward bisimulation*.

For the sake of simplicity, we will call γ just a *simulation* if γ is either a forward or a backward simulation, and just a *bisimulation* if γ is any of the four types of bisimulations defined above. Moreover, forward and backward bisimulations will be called *homotypic*, whereas backward-forward and forward-backward bisimulations will be called *heterotypic*. Moreover, for any $w \in \{fs, bs, fb, bb, fbb, bfb\}$, a fuzzy relation γ which satisfies (w-1), (w-2) and (w-3) will be called simply a *simulation/bisimulation of type w*.

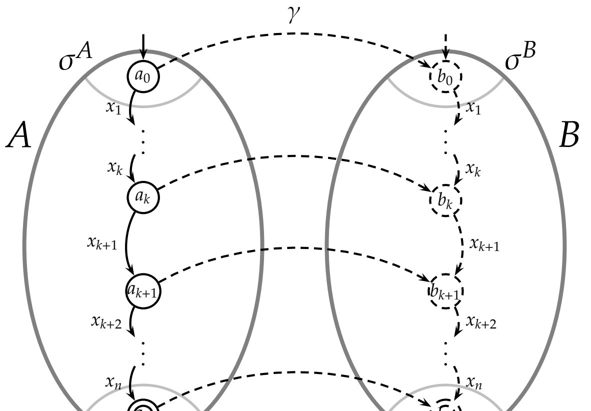


Fig. 2.1 Forward and backward simulation

The meaning of forward and backward simulations can be best explained in the case when $\mathscr{A}$ and $\mathscr{B}$ are nondeterministic (Boolean) automata. For this purpose we will use the diagram shown in Figure 1. Let γ be a forward simulation between $\mathscr{A}$ and $\mathscr{B}$ and let $a_0,a_1,\dots,a_n$ be an arbitrary successful run of the automaton $\mathscr{A}$ on a word $u = x_1x_2\cdots x_n$ ($x_1,x_2,\dots,x_n \in X$), i.e., a sequence of states of $\mathscr{A}$ such that $a_0 \in \sigma^A$, $(a_k,a_{k+1}) \in \delta^A_{x_{k+1}}$, for $0 \leqslant k \leqslant n-1$, and $a_n \in \tau^A$. According to (fs-1), there is an initial state $b_0 \in \sigma^B$ such that $(a_0,b_0) \in \gamma$. Suppose that for some k, $0 \leqslant k \leqslant n-1$, we have built a sequence of states $b_0,b_1,\dots,b_k$ such that $(b_{i-1},b_i) \in \delta^B_{x_i}$ and $(a_i,b_i) \in \gamma$, for each i, $1 \leqslant i \leqslant k$. Then $(b_k,a_{k+1}) \in \gamma^{-1} \circ \delta^A_{x_{k+1}}$, and by ($fs$-2) we obtain that $(b_k,a_{k+1}) \in \delta^B_{x_{k+1}} \circ \gamma^{-1}$, so there exists $b_{k+1} \in B$ such that $(b_k,b_{k+1}) \in \delta^B_{x_{k+1}}$ and $(a_{k+1},b_{k+1}) \in \gamma$. Therefore, we have successively built a sequence $b_0,b_1,\dots,b_n$ of states of $\mathscr{B}$ such that $b_0 \in \sigma^B$, $(b_k,b_{k+1}) \in \delta^B_{x_{k+1}}$, for every k, $0 \leqslant k \leqslant n-1$, and $(a_k,b_k) \in \gamma$, for each k, $0 \leqslant k \leqslant n$. Moreover, by ($fs$-3) we obtain that $b_n \in \tau^B$. Thus, the sequence $b_0,b_1,\dots,b_n$ is a successful run of the automaton $\mathscr{B}$ on the word u which simulates the original run $a_0,a_1,\dots,a_n$ of $\mathscr{A}$ on u. In contrast to forward simulations, where we build the sequence $b_0,b_1,\dots,b_n$ moving forward, starting with b_0 and ending with b_n, in the case of backward simulations we build this sequence moving backward, starting with b_n and ending with b_0. In a similar way we can understand forward and backward simulations between arbitrary fuzzy automata, taking into account degrees of possibility of transitions and degrees of relationship.

Given fuzzy automata $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$, a fuzzy relation $\gamma \in \mathscr{R}(A,B)$ is a backward simulation between fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ if and only if it is a forward simulation between the reverse fuzzy automata $\bar{\mathscr{A}}$ and $\bar{\mathscr{B}}$. Therefore, forward and backward simulations, forward and backward bisimulations, and backward-forward and forward-backward bisimulations, are mutually dual concepts, what means that for any statement on some of these concepts which is universally valid (valid for all fuzzy automata) there is the corresponding universally valid statement on its dual concept.

It is clear that bisimulation is a fuzzy relation which realizes simulation of a fuzzy automaton by another,and its inverse realizes there verse simulation. However, bisimulation is a more restrictive concept than the two-way simulation, by which we mean a pair of simulations of a fuzzy automaton to another,and viceversa, which are not necessarily mutually inverse. Thus,the following example demonstrates a pair of fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ such that there are forward simulations between $\mathscr{A}$ and $\mathscr{B}$ and between $\mathscr{B}$ and $\mathscr{A}$, but there is no any forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$.

Example 2.4. Let $\mathscr{L}$ be the Gödel structure, and let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata over $\mathscr{L}$ and $X = \{x,y\}$ with $|A| = 3$, $|B| = 2$ and

$$\sigma^A = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, \quad \delta^A_x = \begin{bmatrix} 1 & 0.3 & 0.4 \\ 0.5 & 1 & 0.3 \\ 0.4 & 0.6 & 0.7 \end{bmatrix}, \quad \delta^A_y = \begin{bmatrix} 0.5 & 0.6 & 0.2 \\ 0.6 & 0.3 & 0.4 \\ 0.7 & 0.7 & 1 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

$$\sigma^B = \begin{bmatrix} 0.7 & 1 \end{bmatrix}, \quad \delta_x^B = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 0.7 \end{bmatrix}, \quad \delta_y^A = \begin{bmatrix} 0.6 & 0.6 \\ 0.7 & 1 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

We have that fuzzy relations

$$\varphi = \begin{bmatrix} 1 & 0.7 \\ 1 & 0.7 \\ 0.6 & 1 \end{bmatrix}, \quad \phi = \begin{bmatrix} 1 & 1 & 0.7 \\ 0.6 & 0.6 & 1 \end{bmatrix}.$$

are, respectively,the greatest forward simulation between $\mathscr{A}$ and $\mathscr{B}$, and viceversa, but there is no forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$. Indeed, let α be an arbitrary fuzzy relation between $\mathscr{A}$ and $\mathscr{B}$ such that $\varphi \leqslant \alpha$, i.e.,

$$\alpha = \begin{bmatrix} 1 & a \\ 1 & b \\ c & 1 \end{bmatrix}, \quad \text{with} \quad a \geqslant 0.7, \ b \geqslant 0.7 \quad \text{and} \quad c \geqslant 0.6.$$

If $\alpha^{-1} \circ \delta_x^A \leqslant \delta_x^B \circ \alpha^{-1}$, then we easily obtain that $a \leqslant 0.7$, $b \leqslant 0.7$, and $c \leqslant 0.7$, what means that $a = b = 0.7$ and $0.6 \leqslant c \leqslant 0.7$. Moreover, if $\alpha^{-1} \circ \delta_y^A \leqslant \delta_y^B \circ \alpha^{-1}$, then we obtain that $c \leqslant 0.6$, and hence, $c = 0.6$. Therefore, we have proved that $\alpha = \varphi$, and so φ is the greatest forward simulation between $\mathscr{A}$ and $\mathscr{B}$. In a similar way we show that ϕ is the greatest forward simulation between $\mathscr{B}$ and $\mathscr{A}$. Suppose now that β is an arbitrary forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$. Then $\beta \leqslant \varphi$ and $\beta \leqslant \phi^{-1}$, i.e., $\beta \leqslant \varphi \wedge \phi^{-1}$, what implies

$$\sigma^A \circ \beta = \sigma^A \circ (\varphi \wedge \phi^{-1}) \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \circ \begin{bmatrix} 1 & 0.6 \\ 1 & 0.6 \\ 0.6 & 1 \end{bmatrix} = \begin{bmatrix} 0.6 & 1 \end{bmatrix} < \begin{bmatrix} 0.7 & 1 \end{bmatrix} = \sigma^B.$$

This contradicts our assumption that β is a forward bisimulation. Hence,there is not any forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$.

In numerous papers dealing with simulations and bisimulations, forward simulations and forward bisimulations have been mostly studied. They have been usually called just simulations and bisimulations, or strong simulations and strong bisimulations (cf. [72,73,91]). The greatest bisimulation equivalence relation has been usually called the bisimilarity. Automata between which there is a bisimulation have been often called bisimilar. Distinction between forward and backward simulations, and forward and backward bisimulations, has been made, for instance, in [12,35,68] (for various kinds of automata), but less or more these concepts differ from the concepts having the same name which are considered here. More similar to our concepts of forward and backward simulations and bisimulations are those studied in [11], and in [36,37] (for tree automata).

For more information about bisimulations we refer to [1, 24, 40, 44, 77, 82, 83, 103, 109].

Chapter 3
Computation of the greatest simulations and bisimulations between fuzzy automata

As it is mentioned earlier the simulations and bisimulations, introduced in [22] play significant role in the theory of bisimulations for fuzzy automata over complete residuated lattices. In [22] it has been proved that if there is at least one simulation/bisimulation of some of these types between the given fuzzy automata, then there is the greatest simulation/bisimulation of this kind. However, there has not been given any efficient algorithm for deciding whether there is a simulation/bisimulation of some of these types between the given fuzzy automata, and for computing the greatest one, if it exists. In [22] a theorem has been proved which can be used just for checking the existence of a uniform forward bisimulation (i.e., a complete and surjective forward bisimulation) between the given fuzzy automata. According to this theorem, there is a uniform forward bisimulation between fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ if and only if there is a special isomorphism between the factor fuzzy automata of $\mathscr{A}$ and $\mathscr{B}$ with respect to their greatest forward bisimulation fuzzy equivalences. Using the algorithm provided in [27], in numerous cases we can efficiently compute the greatest forward bisimulation fuzzy equivalences E on $\mathscr{A}$ and F on $\mathscr{B}$. Then we can construct factor fuzzy automata $\mathscr{A}/E$ and $\mathscr{B}/F$, and check whether there is an isomorphism between them that satisfies an additional condition stated in [22]. But, even when we are able to efficiently compute the greatest forward bisimulations E and F and construct the factor fuzzy automata $\mathscr{A}/E$ and $\mathscr{B}/F$, it may be difficult to determine whether there is an isomorphism between $\mathscr{A}/E$ and $\mathscr{B}/F$ that satisfies this additional condition. This problem comes down to the well-known *graph isomorphism problem*, which is one of the few important algorithmic problems whose rough computational complexity is still not known, and it is generally accepted that it lies between P and NP-complete if P≠NP (cf. [115]). Fortunately, although no worst-case polynomial-time algorithm is known, testing graph isomorphism is usually not very hard in practice.

In this chapter, for any of the types of simulations/bisimulations, introduced in the Chapter 2, we will provide an efficient algorithm for deciding whether there is a simulation/bisimulation of this type between the given fuzzy automata, and for computing the greatest one, whenever it exists. The algorithms are based on the method developed in [55], which comes down to the computing of the greatest post-fixed point, contained in a given fuzzy relation, of an isotone function on the lattice of fuzzy relations. Namely, for each type of simulations and bisimulations we will determine the corresponding isotone and image-localized function ϕ on the lattice of fuzzy relations, as well as the corresponding initial fuzzy relation π, and the computing of the greatest simulation/bisimulation of this type we will reduce to the the computing of the greatest post-fixed point of ϕ contained in π. This is an iterative procedure by which we will successively build a decreasing sequence of relations, starting from the relation π and using the function ϕ. If this sequence is finite, then it stabilizes and its smallest member is exactly the fuzzy relation which we are searching for, the greatest post-fixed point of ϕ contained in π. We will determine sufficient conditions under which this sequence is finite, when our algorithm terminates in the finite number of steps (cf. Theorem 3.4), as well as sufficient conditions under which the infimum of this sequence is exactly the fuzzy relation which we are searching for (cf. Theorem 3.6). Modifying the algorithms for computing the greatest simulations and bisimulations we will provide algorithms for computing the greatest crisp simulations and bisimulations between fuzzy automata (cf. Proposition 3.1). These algorithms always terminate in a finite number of steps, independently of the properties of the underlying structure of truth values, but we will show that there are fuzzy automata such that there is a simulation or bisimulation of a given type between them, and there is not any crisp simulation/bisimulation of this type (cf. Example 3.1).

The chapter will be composed of three sections. In Section 3.1. right and left residuals of fuzzy relation will be discussed. Using residuals for any of the above mentioned simulations and bisimulations we will define the corresponding function on the lattice of fuzzy relations. Section 3.2. will contain main results on the computing of the greatest simulations and bisimulations between fuzzy automata. In Section 3.3. we will present examples which demonstrate the application of the algorithms and clarify relationships between different types of simulations and bisimulations.

It is worth noting that algorithms for deciding whether there are simulations and bisimulations between nondeterministic automata, and the computing of the greatest simulations and bisimulations, if they exist, were given in [21, 70]. Various algorithms for the computing of the greatest bisimulation equivalences on labelled transition systems can be found in [40, 44, 67, 88, 102, 105].

The results from this chapter are published in [23].

The results in this chapter are original and are closely relates to the results of Ignjatović et all. in [57].

3.1. The residuals

Here, we introduce several notions and notation that will be used in the chapter.

For non-empty sets A and B and fuzzy subsets $\eta \in \mathscr{F}(A)$ and $\xi \in \mathscr{F}(B)$, fuzzy relations $\eta\backslash\xi \in \mathscr{R}(A,B)$ and $\eta/\xi \in \mathscr{R}(A,B)$ are defined as follows

$$(\eta\backslash\xi)(a,b) = (\eta(a) \to \xi(b)), \tag{3.1}$$
$$(\eta/\xi)(a,b) = (\xi(b) \to \eta(a)), \tag{3.2}$$

for arbitrary $a \in A$ and $b \in B$. Let us note that $\eta/\xi = (\xi\backslash\eta)^{-1}$.

We have the following.

Lemma 3.1. *Let A and B be non-empty sets and let $\eta \in \mathscr{F}(A)$ and $\xi \in \mathscr{F}(B)$.*

(a) *The set of all solutions to the inequality $\eta \circ \chi \leqslant \xi$, where χ is an unknown fuzzy relation between A and B, is the principal ideal of $\mathscr{R}(A,B)$ generated by the fuzzy relation η/ξ.*

(b) *The set of all solutions to the inequality $\chi \circ \xi \leqslant \eta$, where χ is an unknown fuzzy relation between A and B, is the principal ideal of $\mathscr{R}(A,B)$ generated by the fuzzy relation $\eta\backslash\xi$.*

Proof. These are the well-known results by E. Sanchez (cf. [106, 107, 108]). □

Note that $(\eta/\xi) \wedge (\eta\backslash\xi) = \eta|\xi$, where $\eta|\xi$ is a fuzzy relation between A and B defined by

$$(\eta|\xi)(a,b) = (\eta(a) \leftrightarrow \xi(b)), \tag{3.3}$$

for arbitrary $a \in A$ and $b \in B$.

Next, let A and B be non-empty sets and let $\alpha \in \mathscr{R}(A)$, $\beta \in \mathscr{R}(B)$ and $\gamma \in \mathscr{R}(A,B)$. The *right residual* of γ by α is a fuzzy relation $\alpha\backslash\gamma \in \mathscr{R}(A,B)$ defined by

$$(\alpha\backslash\gamma)(a,b) = \bigwedge_{a' \in A} (\alpha(a',a) \to \gamma(a',b)), \tag{3.4}$$

for all $a \in A$ and $b \in B$, and the *left residual* of γ by β is a fuzzy relation $\gamma/\beta \in \mathscr{R}(A,B)$ defined by

$$(\gamma/\beta)(a,b) = \bigwedge_{b' \in B} (\beta(b,b') \to \gamma(a,b')), \tag{3.5}$$

for all $a \in A$ and $b \in B$. We think of the right residual $\alpha\backslash\gamma$ as what remains of γ on the right after "dividing" γ on the left by α, and of the left residual γ/β as what remains of γ on the left after "dividing" γ on the right by β. In other words,

$$\alpha \circ \gamma' \leqslant \gamma \;\Leftrightarrow\; \gamma' \leqslant \alpha\backslash\gamma, \qquad \gamma' \circ \beta \leqslant \gamma \;\Leftrightarrow\; \gamma' \leqslant \gamma/\beta, \tag{3.6}$$

for all $\alpha \in \mathscr{R}(A)$, $\beta \in \mathscr{R}(B)$ and $\gamma', \gamma \in \mathscr{R}(A,B)$. In the case when $A = B$, these two concepts become the well-known concepts of right and left residuals of fuzzy relations on a set (cf. [55]).

We also have the following.

Lemma 3.2. *Let A and B be non-empty sets and let $\alpha \in \mathscr{R}(A)$, $\beta \in \mathscr{R}(B)$ and $\gamma \in \mathscr{R}(A,B)$.*

(a)*The set of all solutions to the inequality $\alpha \circ \chi \leqslant \gamma$, where χ is an unknown fuzzy relation between A and B, is the principal ideal of $\mathscr{R}(A,B)$ generated by the right residual $\alpha\backslash\gamma$ of γ by α.*

(b)*The set of all solutions to the inequality $\chi \circ \beta \leqslant \gamma$, where χ is an unknown fuzzy relation between A and B, is the principal ideal of $\mathscr{R}(A,B)$ generated by the left residual γ/β of γ by β.*

Proof. These are also results by E. Sanchez (cf. [106, 107, 108]). □

As we said in the introduction, the problem of deciding whether there is a simulation or bisimulation of a given type between fuzzy automata, and the problem of computing the greatest simulation/bisimulation of this type, will be reduced to the problem of computing the greatest post-fixed point, contained in a given fuzzy relation, of an appropriate isotone function on the lattice of fuzzy relations. For this purpose we define the following fuzzy relations and functions on the lattice of fuzzy relations.

Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata. We define fuzzy relations $\pi^w \in \mathscr{R}(A,B)$, for $w \in \{fs, bs, fb, bb, fbb, bfb\}$, in the following way:

$$\pi^{fs} = \tau^A \backslash \tau^B, \tag{3.7}$$

$$\pi^{bs} = \sigma^A \backslash \sigma^B, \tag{3.8}$$

$$\pi^{fb} = (\tau^A \backslash \tau^B) \wedge (\tau^A / \tau^B) = \tau^A | \tau^B, \tag{3.9}$$

$$\pi^{bb} = (\sigma^A \backslash \sigma^B) \wedge (\sigma^A / \sigma^B) = \sigma^A | \sigma^B, \tag{3.10}$$

$$\pi^{fbb} = (\tau^A \backslash \tau^B) \wedge (\sigma^A / \sigma^B), \tag{3.11}$$

$$\pi^{bfb} = (\sigma^A \backslash \sigma^B) \wedge (\tau^A / \tau^B). \tag{3.12}$$

Moreover, we define functions $\phi^w : \mathscr{R}(A,B) \to \mathscr{R}(A,B)$, for $w \in \{fs, bs, fb, bb, fbb, bfb\}$, as follows:

$$\phi^{fs}(\gamma) = \bigwedge_{x\in X}[(\delta_x^B\circ\gamma^{-1})/\delta_x^A]^{-1}, \tag{3.13}$$

$$\phi^{bs}(\gamma) = \bigwedge_{x\in X}\delta_x^A\backslash(\gamma\circ\delta_x^B), \tag{3.14}$$

$$\phi^{fb}(\gamma) = \bigwedge_{x\in X}[(\delta_x^B\circ\gamma^{-1})/\delta_x^A]^{-1}\wedge[(\delta_x^A\circ\gamma)/\delta_x^B] = \phi^{fs}(\gamma)\wedge[\phi^{fs}(\gamma^{-1})]^{-1}, \tag{3.15}$$

$$\phi^{bb}(\gamma) = \bigwedge_{x\in X}[\delta_x^A\backslash(\gamma\circ\delta_x^B)]\wedge[\delta_x^B\backslash(\gamma^{-1}\circ\delta_x^A)]^{-1} = \phi^{bs}(\gamma)\wedge[\phi^{bs}(\gamma)]^{-1}, \tag{3.16}$$

$$\phi^{fbb}(\gamma) = \bigwedge_{x\in X}[(\delta_x^B\circ\gamma^{-1})/\delta_x^A]^{-1}\wedge[\delta_x^B\backslash(\gamma^{-1}\circ\delta_x^A)]^{-1} = \phi^{fs}(\gamma)\wedge[\phi^{bs}(\gamma^{-1})]^{-1}, \tag{3.17}$$

$$\phi^{bfb}(\gamma) = \bigwedge_{x\in X}[\delta_x^A\backslash(\gamma\circ\delta_x^B)]\wedge[(\delta_x^A\circ\gamma)/\delta_x^B] = \phi^{bs}(\gamma)\wedge[\phi^{fs}(\gamma^{-1})]^{-1}, \tag{3.18}$$

for any $\gamma\in\mathscr{R}(A,B)$. Notice that in the expression "$\phi^w(\alpha^{-1})$" ($w\in\{fs,bs\}$) we denote by ϕ^w a function from $\mathscr{R}(B,A)$ into itself.

The next theorem provides equivalent forms of the second and third conditions in the definitions of simulations and bisimulations.

Theorem 3.1. *Let $\mathscr{A}=(A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B}=(B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata and $w\in\{fs,bs,fb,bb,fbb,bfb\}$. A fuzzy relation $\gamma\in\mathscr{R}(A,B)$ satisfies conditions (w-2) and (w-3) if and only if it satisfies*

$$\gamma\leqslant\phi^w(\gamma),\qquad \gamma\leqslant\pi^w. \tag{3.19}$$

Proof. We will prove only the case $w=fs$. The assertion concerning the case $w=bs$ follows by the duality, and according to equations (3.9)–(3.12) and (3.15)–(3.18), all other assertions can be obtained by the first two.

Consider an arbitrary $\gamma\in\mathscr{R}(A,B)$. According to Lemma 3.1(b), γ satisfies condition (fs-3) if and only if $\gamma^{-1}\leqslant\tau^B/\tau^A=(\tau^A\backslash\tau^B)^{-1}$, which is equivalent to $\gamma\leqslant\tau^A\backslash\tau^B=\pi^{fs}$. Therefore, γ satisfies (fs-3) if and only if $\gamma\leqslant\pi^{fs}$.

On the other hand, γ satisfies (fs-2) if and only if

$$\gamma^{-1}(b,a)\otimes\delta_x^A(a,a')\leqslant(\delta_x^B\circ\gamma^{-1})(b,a'),$$

for all $a,a'\in A$, $b\in B$ and $x\in X$. According to the adjunction property, this is equivalent to

$$\gamma^{-1}(b,a)\leqslant\bigwedge_{a'\in A}[\delta_x^A(a,a')\to(\delta_x^B\circ\gamma^{-1}(b,a'))]=((\delta_x^B\circ\gamma^{-1})/\delta_x^A)(b,a)$$

for all $a\in A$, $b\in B$ and $x\in X$, which is further equivalent to

$$\gamma(a,b)\leqslant\bigwedge_{x\in X}[(\delta_x^B\circ\gamma^{-1})/\delta_x^A]^{-1}(a,b)=(\phi^{fs}(\gamma))(a,b)$$

for all $a \in A$ and $b \in B$. Therefore, γ satisfies (fs-2) if and only if $\gamma \leqslant \phi^{fs}(\gamma)$.

Now we conclude that a fuzzy relation $\gamma \in \mathscr{R}(A,B)$ satisfies (fs-2) and (fs-3) if and only if it satisfies (3.19) (for $\text{-}w = fs$), which was to be proved. □

3.2. Computation of the greatest simulations and bisimulations

We will provide a method for computing the greatest simulations and bisimulations between fuzzy automata adapting the method developed in [55] for computing the greatest post-fixed points of an isotone function on the lattice of fuzzy relations on a single set.

Let A and B be non-empty sets and let $\phi : \mathscr{R}(A,B) \to \mathscr{R}(A,B)$ be an isotone function, i.e., let $\alpha \leqslant \beta$ implies $\phi(\alpha) \leqslant \phi(\beta)$, for all $\alpha, \beta \in \mathscr{R}(A,B)$. A fuzzy relation $\alpha \in \mathscr{R}(A,B)$ is called a *post-fixed point* of ϕ if $\alpha \leqslant \phi(\alpha)$. The well-known Knaster-Tarski fixed point theorem (stated and proved in a more general context, for complete lattices) asserts that the set of all post-fixed points of ϕ form a complete lattice (cf. [104]). Moreover, for any fuzzy relation $\pi \in \mathscr{R}(A,B)$ we have that the set of all post-fixed points of ϕ contained in π is also a complete lattice. According to Theorem 3.1, our main task is to find an efficient procedure for computing the greatest post-fixed point of the function ϕ^w contained in the fuzzy relation π^w, for each $w \in \{fs, bs, fb, bb, fbf, bfb\}$.

It should be noted that the set of all post-fixed points of an isotone function on a complete lattice is always non-empty, because it contains the least element of this complete lattice. However, this set may consist only of that single element. In our case, when we are working with a lattice of fuzzy relations, the empty relation may be the only post-fixed point, whereas we have defined simulations and bisimulations to be non-empty fuzzy relations. This requirement is necessary because the empty relation can not satisfy the condition (w-1), unless the fuzzy set of initial states or the fuzzy set of terminal states is also empty. Therefore, our task is actually to find an efficient procedure for deciding whether there is a non-empty post-fixed point of ϕ^w contained in π^w, and if it exists, then find the greatest one.

Let $\phi : \mathscr{R}(A,B) \to \mathscr{R}(A,B)$ be an isotone function and $\pi \in \mathscr{R}(A,B)$. We define a sequence $\{\gamma_k\}_{k\in\mathbb{N}}$ of fuzzy relations from $\mathscr{R}(A,B)$ by

$$\gamma_1 = \pi, \qquad \gamma_{k+1} = \gamma_k \wedge \phi(\gamma_k), \text{ for each } k \in \mathbb{N}. \tag{3.20}$$

The sequence $\{\gamma_k\}_{k\in\mathbb{N}}$ is obviously descending. If we denote by $\hat{\gamma}$ the greatest post-fixed point of ϕ contained in π, we can easily verify that

$$\hat{\gamma} \leqslant \bigwedge_{k\in\mathbb{N}} \gamma_k. \tag{3.21}$$

Now two very important questions arise. First, under what conditions the equality holds in (3.21)? Even more important question is: under what conditions the sequence $\{\gamma_k\}_{k\in\mathbb{N}}$ is finite? If this sequence is finite, then it is not hard to show that there exists $k \in \mathbb{N}$ such that $\gamma_k = \gamma_m$, for every $m \geqslant k$, i.e., there exists $k \in \mathbb{N}$ such that the sequence stabilizes on γ_k. We can recognize that the sequence has stabilized when we find the smallest $k \in \mathbb{N}$ such that $\gamma_k = \gamma_{k+1}$. In this case $\hat{\gamma} = \gamma_k$, and we have an algorithm which computes $\hat{\gamma}$ in a finite number of steps.

Some conditions under which equality holds in (3.21) or the sequence is finite were found in [55], for fuzzy relations on a single set. It is easy to show that the same results are also valid for fuzzy relations between two sets. In the sequel we present these results.

First we note that a sequence $\{\gamma_k\}_{k\in\mathbb{N}}$ of fuzzy relations from $\mathscr{R}(A,B)$ is finite if and only it it is *image-finite*, which means that the set $\bigcup_{k\in\mathbb{N}} \mathrm{Im}(\gamma_k)$ is finite. Next, the function $\phi : \mathscr{R}(A,B) \to \mathscr{R}(A,B)$ is called *image-localized* if there exists a finite $K \subseteq L$ such that for each fuzzy relation $\gamma \in \mathscr{R}(A,B)$ we have

$$\mathrm{Im}(\phi(\gamma)) \subseteq \langle K \cup \mathrm{Im}(\gamma)\rangle, \tag{3.22}$$

where $\langle K \cup \mathrm{Im}(\gamma)\rangle$ denotes the subalgebra of $\mathscr{L}$ generated by the set $K \cup \mathrm{Im}(\gamma)$. Such K will be called a *localization set* of the function ϕ.

Theorem analogous to the following theorem was proved in [55] for fuzzy relations on a single set, but its proof can be easily transformed into the proof of the corresponding theorem concerning fuzzy relations between two sets.

Theorem 3.2. *Let the function ϕ be image-localized, let K be its localization set, let $\pi \in \mathscr{R}(A,B)$, and let $\{\gamma_k\}_{k\in\mathbb{N}}$ be the sequence of fuzzy relations in $\mathscr{R}(A,B)$ defined by* (3.20). *Then*

$$\bigcup_{k\in\mathbb{N}} \mathrm{Im}(\gamma_k) \subseteq \langle K \cup \mathrm{Im}(\pi)\rangle. \tag{3.23}$$

If, moreover, $\langle K \cup \mathrm{Im}(\pi)\rangle$ is a finite subalgebra of $\mathscr{L}$, then the sequence $\{\gamma_k\}_{k\in\mathbb{N}}$ is finite.

Going back now to the functions ϕ^w, for $w \in \{fs, bs, fb, bb, fbb, bfb\}$, we prove the following.

Theorem 3.3. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be arbitrary automata.*

For any $w \in \{fs, bs, fb, bb, fbb, bfb\}$ the function ϕ^w is isotone and image-localized.

Proof. We will prove only the case $w = fs$. The assertion concerning the case $w = bs$ follows by the duality, and according to equations (3.9)–(3.12) and (3.15)–(3.18), all other assertions can be derived from the first two.

Let $\gamma_1, \gamma_2 \in \mathscr{R}(A,B)$ be fuzzy relations such that $\gamma_1 \leqslant \gamma_2$, and consider the following systems of fuzzy relation inequalities:

$$\chi^{-1} \circ \delta_x^A \leqslant \delta_x^B \circ \gamma_1^{-1}, \quad x \in X; \tag{3.24}$$

$$\chi^{-1} \circ \delta_x^A \leqslant \delta_x^B \circ \gamma_2^{-1}, \quad x \in X. \tag{3.25}$$

where $\chi \in \mathscr{R}(A,B)$ is an unknown fuzzy relation. Using Lemma 3.2 (b) and the definition of an inverse relation, it can be easily shown that the set of all solutions to system (3.24) (resp. (3.25)) form a principal ideal of $\mathscr{R}(A,B)$ generated by $\phi^{fs}(\gamma_1)$ (resp. $\phi^{fs}(\gamma_2)$). Since for each $x \in X$ we have that $\delta_x^B \circ \gamma_1^{-1} \leqslant \delta_x^B \circ \gamma_2^{-1}$, we conclude that every solution to (3.24) is a solution to (3.25). Consequently, $\phi^{fs}(\gamma_1)$ is a solution to (3.25), so $\phi^{fs}(\gamma_1) \leqslant \phi^{fs}(\gamma_2)$. Therefore, we have proved that ϕ^{fs} is an isotone function.

Next, let $K = \bigcup_{x\in X}(\mathrm{Im}(\delta_x^A) \cup \mathrm{Im}(\delta_x^B))$ and let $\gamma \in \mathscr{R}(A,B)$ be an arbitrary fuzzy relation. It is evident that $\mathrm{Im}(\phi^{fs}(\gamma)) \subseteq \langle K \bigcup \mathrm{Im}(\gamma)\rangle$, and since the input alphabet X is finite, then K is also finite. This confirms that the function ϕ^{fs} is image-localized. □

Now we are ready to prove the main result of this chapter, which provides algorithms for deciding whether there is a simulation or bisimulation of a given type between fuzzy automata, and for computing the greatest simulations and bisimulations, when they exist.

Theorem 3.4. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata, let $w \in \{fs,bs,fb,bb,fbb,bfb\}$, and let a sequence $\{\gamma_k\}_{k\in\mathbb{N}}$ of fuzzy relations from $\mathscr{R}(A,B)$ be defined by*

$$\gamma_1 = \pi^w, \qquad \gamma_{k+1} = \gamma_k \wedge \phi^w(\gamma_k), \text{ for each } k \in \mathbb{N}. \tag{3.26}$$

If $\langle \mathrm{Im}(\pi^w) \cup \bigcup_{x\in X}(\mathrm{Im}(\delta_x^A) \cup \mathrm{Im}(\delta_x^B))\rangle$ is a finite subalgebra of $\mathscr{L}$, then the following is true:

(a)*the sequence $\{\gamma_k\}_{k\in\mathbb{N}}$ is finite and descending, and there is the least natural number k such that $\gamma_k = \gamma_{k+1}$;*
(b)*γ_k is the greatest fuzzy relation in $\mathscr{R}(A,B)$ which satisfies (w-2) and (w-3);*
(c)*if γ_k satisfies (w-1), then it is the greatest fuzzy relation in $\mathscr{R}(A,B)$ which satisfies (w-1), (w-2) and (w-3);*
(d)*if γ_k does not satisfy (w-1), then there is no any fuzzy relation in $\mathscr{R}(A,B)$ which satisfies (w-1), (w-2) and (w-3).*

Proof. We will prove only the case $w = fs$. All other cases can be proved in a similar manner.

So, let $\langle \mathrm{Im}(\pi^{fs}) \cup \bigcup_{x\in X}(\mathrm{Im}(\delta_x^A) \cup \mathrm{Im}(\delta_x^A))\rangle$ be a finite subalgebra of $\mathscr{L}$.

(a) According to Theorems 3.3 and 3.2, the sequence $\{\gamma_k\}_{k\in N}$ is finite and descending, so there are $k,m \in \mathbb{N}$ such that $\gamma_k = \gamma_{k+m}$, whence $\gamma_{k+1} \leqslant \gamma_k = \gamma_{k+m} \leqslant \gamma_{k+1}$. Thus, there is $k \in \mathbb{N}$ such that $\gamma_k = \gamma_{k+1}$, and consequently, there is the least natural number having this property.

(b) By $\gamma_k = \gamma_{k+1} = \gamma_k \wedge \phi^{fs}(\gamma_k)$ we obtain that $\gamma_k \leqslant \phi^{fs}(\gamma_k)$, and also, $\gamma_k \leqslant \gamma_1 = \pi^{fs}$. Therefore, by Theorem 3.1 it follows that γ_k satisfies (fs-2) and (fs-3).

Let $\alpha \in \mathscr{R}(A,B)$ be an arbitrary fuzzy relation which satisfies (fs-2) and (fs-3). As we have already noted, α satisfies (fs-3) if and only if $\alpha \leqslant \pi^{fs} = \gamma_1$. Next, suppose that $\alpha \leqslant \gamma_n$, for some $n \in \mathbb{N}$. Then for every $x \in X$ we have that $\alpha^{-1} \circ \delta_x^A \leqslant \delta_x^B \circ \alpha^{-1} \leqslant \delta_x^B \circ \gamma_n^{-1}$, and according to Lemma 3.2 (b), $\alpha^{-1} \leqslant (\delta_x^B \circ \gamma_n^{-1})/\delta_x^A$, i.e., $\alpha \leqslant [(\delta_x^B \circ \gamma_n^{-1})/\delta_x^A]^{-1} = \phi^{fs}(\gamma_n)$. Therefore, $\alpha \leqslant \gamma_n \wedge \phi^{fs}(\gamma_n) = \gamma_{n+1}$. Now, by induction we obtain that $\alpha \leqslant \gamma_n$, for every $n \in \mathbb{N}$, and hence, $\alpha \leqslant \gamma_k$. This means that γ_k is the greatest fuzzy relation in $\mathscr{R}(A,B)$ satisfying (fs-2) and (fs-3).

(c) This follows immediately from (b).

(d) Suppose that γ_k does not satisfy (fs-1). Let $\gamma \in \mathscr{R}(A,B)$ be an arbitrary fuzzy relation which satisfies (fs-1), (fs-2) and (fs-3). According to (b) of this theorem, $\gamma \leqslant \gamma_k$, so we have that $\sigma^A \leqslant \sigma^B \circ \gamma^{-1} \leqslant \sigma^B \circ \gamma_k^{-1}$. But, this contradicts our starting assumption that γ_k does not satisfy (fs-1). Hence, we conclude that there is no any fuzzy relation in $\mathscr{R}(A,B)$ which satisfies (fs-1), (fs-2) and (fs-3). □

Algorithm 3.5 *Construction of the greatest simulation/bisimulation* The input of this algorithm are fuzzy finite automata $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$. The algorithm decides whether there is a simulation or bisimulation between $\mathscr{A}$ and $\mathscr{B}$ of a given type $w \in \{fs,bs,fb,bb,fbb,bfb\}$, and when it exists, the output of the algorithm is is the greatest simulation/bisimulation of the type w.

The procedure is to construct the sequence of fuzzy relations $\{\gamma_k\}_{k\in\mathbb{N}}$, in the following way:

(A1) In the first step we compute π^w and we set $\gamma_1 = \pi^w$.
(A2) After the kth step let a fuzzy relation γ_k have been constructed.
(A3) In the next step we construct the fuzzy relation γ_{k+1} by means of the formula $\gamma_{k+1} = \gamma_k \wedge \phi^w(\gamma_k)$.
(A4) Simultaneously, we check whether $\gamma_{k+1} = \gamma_k$.
(A5) When we find the smallest number k such that $\gamma_{k+1} = \gamma_k$, the procedure of constructing the sequence $\{\gamma_k\}_{k\in\mathbb{N}}$ terminates, and we check whether γ_k satisfies (w-1).
If γ_k satisfies (w-1), then it is the greatest simulation/bisimulation between $\mathscr{A}$ and $\mathscr{B}$ of type w, and if γ_k does not satisfy (w-1), then there is no any simulation/bisimulation between $\mathscr{A}$ and $\mathscr{B}$ of type w.

If the underlying structure of membership values $\mathscr{L}$ is locally finite, then the algorithm terminates in a finite number of steps, for any fuzzy finite automata over $\mathscr{L}$. On the other hand, if $\mathscr{L}$ is not locally finite, then the algorithm terminates in a finite number of steps under conditions determined by Theorems 3.2 and 3.4.

Note that the claim of Theorem 3.4 referring to forward bisimulations, i.e., the related part of Algorithm 5.3, is a straightforward generalization of a well-known result concerning forward bisimulations between ordinary nondeterministic automata, given in Kozen's book [70, page 106]. Theorem

3.4 also generalizes recent results given in [21], which refer to all four types of bisimulations between nondeterministic automata.

Next, we will consider the case when $\mathscr{L} = (L, \wedge, \vee, \otimes, \to, 0, 1)$ is a complete residuated lattice satisfying the following conditions:

$$x \vee \Big(\bigwedge_{i\in I} y_i\Big) = \bigwedge_{i\in I} (x \vee y_i), \tag{3.27}$$

$$x \otimes \Big(\bigwedge_{i\in I} y_i\Big) = \bigwedge_{i\in I} (x \otimes y_i), \tag{3.28}$$

for all $x \in L$ and $\{y_i\}_{i\in I} \subseteq L$. Let us note that if $\mathscr{L} = ([0,1], \wedge, \vee, \otimes, \to, 0, 1)$, where $[0,1]$ is the real unit interval and $\otimes$ is a left-continuous t-norm on $[0,1]$, then (3.27) follows immediately by linearity of $\mathscr{L}$, and $\mathscr{L}$ satisfies (3.28) if and only if $\otimes$ is a continuous t-norm, i.e., if and only if $\mathscr{L}$ is a *BL*-algebra (cf. [4, 10]). Therefore, conditions (3.27) and (3.28) hold for every *BL*-algebra on the real unit interval. In particular, the Łukasiewicz, Goguen (product) and Gödel structures fulfill (3.27) and (3.28).

Under these conditions we have the following.

Theorem 3.6. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata, let $w \in \{fs, bs, fb, bb, fbb, bfb\}$, let $\{\gamma_k\}_{k\in\mathbb{N}}$ be the sequence of fuzzy relations from $\mathscr{R}(A,B)$ defined by* (3.26)*, and let*

$$\gamma = \bigwedge_{k\in\mathbb{N}} \gamma_k. \tag{3.29}$$

If $\mathscr{L}$ is a complete residuated lattice satisfying (3.27) *and* (3.28)*, then the following is true:*

(a) *γ is the greatest fuzzy relation in $\mathscr{R}(A,B)$ which satisfies* (w-2) *and* (w-3)*;*
(b) *if γ satisfies* (w-1)*, then it is the greatest fuzzy relation in $\mathscr{R}(A,B)$ which satisfies* (w-1)*,* (w-2) *and* (w-3)*;*
(c) *if γ does not satisfy* (w-1)*, then there is no any fuzzy relation in $\mathscr{R}(A,B)$ which satisfies* (w-1)*,* (w-2) *and* (w-3)*.*

Proof. We will prove only the case $w = fs$. All other cases can be proved similarly.

So, let $\mathscr{L}$ be a complete residuated lattice satisfying (3.27) and (3.28). First, notice that it has been shown in [27] that if (3.27) holds, then for all descending sequences $\{x_k\}_{k\in\mathbb{N}}, \{y_k\}_{k\in\mathbb{N}} \subseteq L$ we have

$$\bigwedge_{k\in\mathbb{N}} (x_k \vee y_k) = \Big(\bigwedge_{k\in\mathbb{N}} x_k\Big) \vee \Big(\bigwedge_{k\in\mathbb{N}} y_k\Big). \tag{3.30}$$

(a) For arbitrary $x \in X$, $a \in A$ and $b \in B$ we have that

$$\begin{aligned}
\Big(\bigwedge_{k\in\mathbb{N}}(\delta_x^B\circ\gamma_k^{-1})\Big)(b,a) &= \bigwedge_{k\in\mathbb{N}}(\delta_x^B\circ\gamma_k^{-1})(b,a) = \bigwedge_{k\in\mathbb{N}}\Big(\bigvee_{b'\in B}\delta_x^B(b,b')\otimes\gamma_k^{-1}(b',a)\Big)\\
&= \bigvee_{b'\in B}\Big(\bigwedge_{k\in\mathbb{N}}\delta_x^B(b,b')\otimes\gamma_k^{-1}(b',a)\Big) && \text{(by (3.30))}\\
&= \bigvee_{b'\in B}\Big(\delta_x^B(b,b')\otimes\Big(\bigwedge_{k\in\mathbb{N}}\gamma_k^{-1}(b',a)\Big)\Big) && \text{(by (3.28))}\\
&= \bigvee_{b'\in B}\Big(\delta_x^B(b,b')\otimes\gamma^{-1}(b',a)\Big) = (\delta_x^B\circ\gamma^{-1})(b,a),
\end{aligned}$$

which means that

$$\bigwedge_{k\in\mathbb{N}}\delta_x^B\circ\gamma_k^{-1} = \delta_x^B\circ\gamma^{-1},$$

for every $x\in X$. The use of condition (3.30) is justified by the facts that B is finite, and that $\{\gamma_k^{-1}(b',a)\}_{k\in\mathbb{N}}$ is a descending sequence, so $\{\delta_x^B(b,b')\otimes\gamma_k^{-1}(b',a)\}_{k\in\mathbb{N}}$ is also a descending sequence.

Now, for all $x\in X$ and $k\in\mathbb{N}$ we have that

$$\gamma\leqslant\gamma_{k+1}\leqslant\phi^{fs}(\gamma_k) = [(\delta_x^B\circ\gamma_k^{-1})/\delta_x^A]^{-1},$$

which is equivalent to

$$\gamma^{-1}\circ\delta_x^A\leqslant\delta_x^B\circ\gamma_k^{-1}.$$

As the last inequality holds for every $k\in\mathbb{N}$, we have that

$$\gamma^{-1}\circ\delta_x^A\leqslant\bigwedge_{k\in\mathbb{N}}\delta_x^B\circ\gamma_k^{-1} = \delta_x^B\circ\gamma^{-1},$$

for every $x\in X$. Therefore, γ satisfies (fs-2). Moreover, $\gamma\leqslant\gamma_1=\pi^{fs}$, so γ also satisfies (fs-3).

Next, let $\alpha\in\mathscr{R}(A,B)$ be an arbitrary fuzzy relation satisfying (fs-2) and (fs-3). According to Theorem 3.1, $\alpha\leqslant\phi^{fs}(\alpha)$ and $\alpha\leqslant\pi^{fs}=\gamma_1$. By induction we can easily prove that $\alpha\leqslant\gamma_k$ for every $k\in\mathbb{N}$, and therefore, $\alpha\leqslant\gamma$. This means that γ is the greatest fuzzy relation in $\mathscr{R}(A,B)$ which satisfies (fs-2) and (fs-3).

The assertion (b) follows immediately from (a), whereas the assertion (c) can be proved in the same way as the assertion (d) of Theorem 3.4. □

In some situations we do not need simulations and bisimulations that are fuzzy relations, but those that are ordinary crisp relations. Moreover, in cases where our algorithms for computing the greatest simulations and bisimulations fail to terminate in a finite number of steps, we can search for the greatest crisp simulations and bisimulations. They can be understood as a kind of "approximations" of the greatest fuzzy simulations and bisimulations. Here we show that the above given algorithms for computing the greatest fuzzy simulations and bisimulations can be modified to compute the greatest crisp simulations and bisimulations. The new algorithms terminate in a finite number of steps, independently of the properties of the underlying structure of truth values. In the next section we will give an example which shows that the greatest crisp simulations and bisimulations can not be obtained simply by taking the crisp parts of the greatest fuzzy simulations and bisimulations. In fact, our example shows that there may be a fuzzy simulation/bisimulation of a given type between two fuzzy automata, but there is not any crisp simulation/bisimulation of this type between them.

Let A and B be non-empty finite sets, and let $\mathscr{R}^c(A,B)$ denote the set of all crisp relations from $\mathscr{R}(A,B)$. It is not hard to verify that $\mathscr{R}^c(A,B)$ is a complete sublattice of $\mathscr{R}(A,B)$, i.e., the meet and the join in $\mathscr{R}(A,B)$ of an arbitrary family of crisp relations from $\mathscr{R}^c(A,B)$ are also crisp relations (in fact, they coincide with the ordinary intersection and union of crisp relations). Moreover, for each fuzzy relation $\gamma \in \mathscr{R}(A,B)$ we have that $\gamma^c \in \mathscr{R}^c(A,B)$, where γ^c denotes the *crisp part* of a fuzzy relation γ (in some sources called the *kernel* of γ), i.e., a function $\gamma^c : A\times B \to \{0,1\}$ defined by $\gamma^c(a,b)=1$, if $\gamma(a,b)=1$, and $\gamma^c(a,b)=0$, if $\gamma(a,b)<1$, for arbitrary $a\in A$ and $b\in B$. Equivalently, γ^c is considered as an ordinary crisp relation between A and B given by $\gamma^c = \{(a,b)\in A\times B \mid \gamma(a,b)=1\}$.

For each function $\phi : \mathscr{R}(A,B)\to\mathscr{R}(A,B)$ we define a function $\phi^c : \mathscr{R}^c(A,B)\to \mathscr{R}^c(A,B)$ by

$$\phi^c(\gamma) = (\phi(\gamma))^c, \text{ for any } \gamma\in\mathscr{R}^c(A,B).$$

If ϕ is isotone, then it can be easily shown that ϕ^c is also an isotone function.

We have that the following is true.

Proposition 3.1. *Let A and B be non-empty finite sets, let $\phi : \mathscr{R}(A,B)\to\mathscr{R}(A,B)$ be an isotone function and let $\pi\in\mathscr{R}(A,B)$ be a given fuzzy relation. A crisp relation $\varrho\in\mathscr{R}^c(A,B)$ is the greatest crisp solution in $\mathscr{R}(A,B)$ to the system*

$$\chi \leqslant \phi(\chi), \qquad \chi\leqslant\pi, \tag{3.31}$$

if and only if it is the greatest solution in $\mathscr{R}^c(A,B)$ to the system

$$\xi \leqslant \phi^c(\xi), \qquad \xi\leqslant\pi^c, \tag{3.32}$$

where χ is an unknown fuzzy relation and ξ is an unknown crisp relation.

Furthermore, a sequence $\{\varrho_k\}_{k\in\mathbb{N}}\subseteq\mathscr{R}(A,B)$ defined by

$$\varrho_1 = \pi^c, \;\; \varrho_{k+1} = \varrho_k \wedge \phi^c(\varrho_k), \;\; \textit{for every } k\in\mathbb{N}, \tag{3.33}$$

is a finite descending sequence of crisp relations, and the least member of this sequence is the greatest solution to the system (3.32) in $\mathscr{R}^c(A,B)$.

Proof. The proof of this proposition can be obtained simply by translating the proof of Theorem 5.8. [55] to the case of relations between the two sets. □

Taking ϕ to be any of the functions ϕ^w, for $w \in \{fs, bs, fb, bb, fbb, bfb\}$, Proposition 3.1 gives algorithms for deciding whether there is a crisp simulation or bisimulation of a given type between fuzzy automata, and for computing the greatest crisp simulations and bisimulations, when they exist. As we have seen in Proposition 3.1, these algorithms always terminate in a finite number of steps, independently of the properties of the underlying structure of truth values. However, as we have already mentioned, in the next section we will give an example which shows that there may be a fuzzy simulation/bisimulation of a given type between two fuzzy automata, and there is not any crisp simulation/bisimulation of this type between them.

It is worth noting that functions $(\phi^w)^c$, for all $w \in \{fs, bs, fb, bb, fbb, bfb\}$, can be characterized as follows:

$$
\begin{aligned}
&(a,b) \in (\phi^{fs})^c(\varrho) \iff (\forall x \in X)(\forall a' \in A)\, \delta_x^A(a,a') \leqslant (\delta_x^B \circ \varrho^{-1})(b,a'),\\
&(a,b) \in (\phi^{bs})^c(\varrho) \iff (\forall x \in X)(\forall a' \in A)\, \delta_x^A(a',a) \leqslant (\varrho \circ \delta_x^B)(a',b),\\
&(\phi^{fb})^c(\varrho) = (\phi^{fs})^c(\varrho) \wedge [(\phi^{fs})^c(\varrho^{-1})]^{-1}, \ (\phi^{bb})^c(\varrho) = (\phi^{bs})^c(\varrho) \wedge [(\phi^{bs})^c(\varrho^{-1})]^{-1},\\
&(\phi^{fbb})^c(\varrho) = (\phi^{fs})^c(\varrho) \wedge [(\phi^{bs})^c(\varrho^{-1})]^{-1}, \ (\phi^{bfb})^c(\varrho) = (\phi^{bs})^c(\varrho) \wedge [(\phi^{fs})^c(\varrho^{-1})]^{-1},
\end{aligned}
$$

for all $\varrho \in \mathscr{R}^c(A,B)$, $a \in A$ and $b \in B$.

3.3. Computational examples

In this section we give examples which demonstrate the application of our algorithms and clarify relationships between different types of simulations and bisimulations.

The first example demonstrates the case when there are all types of simulations and bisimulations, but there is not any crisp bisimulation between two given automata.

Example 3.1. Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be automata over the Gödel structure and an alphabet $X = \{x, y\}$, with $|A| = 3$, $|B| = 2$, and fuzzy transition relations and fuzzy sets of initial and terminal states which are represented by the following fuzzy matrices and vectors:

$$\sigma^A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \quad \delta_x^A = \begin{bmatrix} 1 & 0.3 & 0.4 \\ 0.5 & 1 & 0.3 \\ 0.4 & 0.6 & 0.7 \end{bmatrix}, \quad \delta_y^A = \begin{bmatrix} 0.5 & 0.6 & 0.2 \\ 0.6 & 0.3 & 0.4 \\ 0.7 & 0.7 & 1 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{3.34}$$

$$\sigma^B = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad \delta_x^B = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 0.7 \end{bmatrix}, \quad \delta_y^B = \begin{bmatrix} 0.6 & 0.6 \\ 0.7 & 1 \end{bmatrix}, \quad \tau^B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{3.35}$$

Using algorithms based on Theorem 3.4 we obtain that there are all types of simulations and bisimulations between fuzzy automata $\mathscr{A}$ and $\mathscr{B}$, and the greatest simulations and bisimulations are given by matrices

$$\gamma^{fs} = \begin{bmatrix} 1 & 0.7 \\ 1 & 0.7 \\ 0.6 & 1 \end{bmatrix}, \ \gamma^{bs} = \begin{bmatrix} 1 & 0.7 \\ 1 & 0.7 \\ 0.7 & 1 \end{bmatrix},$$

$$\gamma^{fb} = \begin{bmatrix} 1 & 0.6 \\ 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}, \ \gamma^{bb} = \begin{bmatrix} 1 & 0.7 \\ 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}, \ \gamma^{fbb} = \begin{bmatrix} 1 & 0.7 \\ 1 & 0.7 \\ 0.6 & 1 \end{bmatrix}, \ \gamma^{bfb} = \begin{bmatrix} 1 & 0.6 \\ 1 & 0.6 \\ 0.7 & 1 \end{bmatrix}.$$

On the other hand, using the version of the algorithms for crisp simulations and bisimulations, we obtain that there is not any crisp bisimulation between fuzzy automata $\mathscr{A}$ and $\mathscr{B}$.

The second example demonstrates the case when there are is a forward bisimulation, but there is not any other type of bisimulations between two given automata.

Example 3.2. Let us change σ^A in (3.34) and σ^B in (3.35) to

$$\sigma^A = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \qquad \sigma^B = \begin{bmatrix} 1 & 0.5 \end{bmatrix}.$$

Then the greatest forward bisimulation between fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ is given by

$$\gamma^{fb} = \begin{bmatrix} 1 & 0.6 \\ 1 & 0.6 \\ 0.6 & 1 \end{bmatrix},$$

but there are no bisimulations of any other type between $\mathscr{A}$ and $\mathscr{B}$.

Due to the duality between forward and backward bisimulations, it is possible to construct automata between which there is a backward bisimulation, but there is not any other type of bisimulations.

We can also give an example which demonstrates the case when there are is a backward-forward bisimulation, but there is not any other type of bisimulations between two given automata.

Example 3.3. Let us change σ^A in (3.34) and σ^B in (3.35) to

$$\sigma^A = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, \qquad \sigma^B = \begin{bmatrix} 0.7 & 1 \end{bmatrix}.$$

Then the greatest backward-forward bisimulation between fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ is given by

$$\gamma^{bfb} = \begin{bmatrix} 1 & 0.6 \\ 1 & 0.6 \\ 0.7 & 1 \end{bmatrix},$$

but there are no bisimulations of any other type between $\mathscr{A}$ and $\mathscr{B}$.

Due to the duality between backward-forward and forward-backward bisimulations, it is possible to construct automata between which there is a backward bisimulation, but there is not any other type of bisimulations.

Next, we give an example which demonstrates the case when there is not any type of simulations and bisimulations between two given automata.

Example 3.4. Let us change σ^A in (3.34) and σ^B in (3.35) to

$$\sigma^A = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \qquad \sigma^B = \begin{bmatrix} 0.5 & 1 \end{bmatrix}.$$

Then there are no simulations and bisimulations of any type between $\mathscr{A}$ and $\mathscr{B}$.

The following example considers the case (over the product structure) when the sequence of fuzzy relations defined by (3.26) is infinite, and its intersection is the greatest forward bisimulation between the given fuzzy automata.

Example 3.5. Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be automata over the Goguen (product) structure and an alphabet $X = \{x\}$, with $|A| = 3$, $|B| = 2$, and fuzzy transition relations and fuzzy sets of initial and terminal states which are represented by the following fuzzy matrices and vectors:

$$\sigma^A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \quad \delta^A_x = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \sigma^B = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad \delta^B_x = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}, \quad \tau^B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Computing the sequence $\{\gamma_k\}_{k \in \mathbb{N}}$ for forward bisimulations by the formula (3.26) ($w = fb$) we obtain that

$$\gamma_k = \begin{bmatrix} 1 & \frac{1}{2^{k-1}} \\ 1 & \frac{1}{2^{k-1}} \\ \frac{1}{2^{k-1}} & 1 \end{bmatrix}, \text{ for each } k \in \mathbb{N}, \qquad \gamma = \bigwedge_{k \in \mathbb{N}} \gamma_k = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

According to Theorem 3.6, γ is the greatest forward bisimulation between fuzzy automata $\mathscr{A}$ and $\mathscr{B}$.

The last example shows that the finiteness of the subalgebra $\langle \mathrm{Im}(\pi^w) \cup \bigcup_{x \in X} (\mathrm{Im}(\delta^A_x) \cup \mathrm{Im}(\delta^B_x)) \rangle$ of $\mathscr{L}$, which appears as an assumption in Theorem 3.4, is sufficient for the finiteness of the sequence defined by (3.26), but it is not necessary.

Example 3.6. Let us change σ^A, σ^B, τ^A and τ^B in the previous example to

$$\sigma^A = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \qquad \sigma^B = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad \tau^A = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \qquad \tau^B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Computing the fuzzy relations γ_k, $k \in \mathbb{N}$, using the formula (3.26), we obtain that

$$\gamma_1 = \gamma_2 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

and it is the greatest greatest forward bisimulation between fuzzy automata $\mathscr{A}$ and $\mathscr{B}$.

On the other hand, we have that the subalgebra $\langle \mathrm{Im}(\pi^w) \cup \mathrm{Im}(\delta_x^A) \cup \mathrm{Im}(\delta_x^B) \rangle$ of $\mathscr{L}$ is not finite, since it contains $\frac{1}{2^k}$, for every $k \in \mathbb{N}$.

Chapter 4
Weak bisimulation for fuzzy automata

In the case of nondeterministic and fuzzy automata the problem of expressing the language equivalence in terms of relationships between states of given automata is very complicated, and we can only examine different approximations of the language-equivalence. The problem of language equivalence of fuzzy automata was first studied by Santos in [111, 112], where he considered equivalence of finite max-min fuzzy automata and max-product fuzzy automata. The equivalence problem for fuzzy automata with membership degrees in a bounded chain was studied by Peeva [95], where she considered various special equivalence problems and provided their algorithmic decidability. Later, equivalence of fuzzy automata over complete residuated lattice was discussed by Xing et al. in [128].

Although, the bisimulations, introduced in Chapter 2, are shown to be a very good means for approximating the language equivalence between two fuzzy automata, there exist fuzzy automata which are language equivalent but there is no any type of bisimulation between them. In order to more precisely describe the class of all relations between the states of fuzzy automata, which preserve the language equivalence, the more general class of bisimulation is introduced.

The fundamental goal of this chapter is to define two new kinds of bisimulations, weak forward and backward bisimulations, which provide better approximations of the language-equivalence than forward and backward bisimulations. Moreover the weak simulations and bisimulations provide better results in the reduction of the states of fuzzy automata.

Because of the fact that every bisimulation is primarily a simulation, in the Section 4.1. the notion of weak forward and backward simulation will be introduced and some fundamental properties will be discussed. It is important to mention, that weak forward(backward) simulation is generalization of the notion of forward(backward) simulation. It will be shown that the existence of weak forward (backward) simulation between two automata implies language inclusion between them. The procedures for deciding whether there exist weak forward and backward simulations, and for computing the greatest ones, whenever they exist will be presented(Theorem 4.2).

Afterwards, in Section 4.2. the notion of weak forward and backward bisimulations will be given and their main features will be considered. The weak bisimulations approximate the language equivalence better then weak simulations, that is, the existence of weak forward (backward) bisimulation between two automata implies their language equivalence. The example showing that weak forward bisimulations are better means for modeling the language equivalence between fuzzy automata than forward bisimulations will be presented(Example 4.2). In the same section, the procedure for computing the greatest weak bisimulations will be proposed(Theorem 4.5). Further, a weak forward bisimulation from a fuzzy automaton $\mathscr{A}$ into itself, called a weak forward bisimulation on $\mathscr{A}$ will be discussed. In the class of all weak forward bisimulations on $\mathscr{A}$, the special attention will be dedicate to these which are equivalences. The example showing fuzzy automata for which the reduction by means of the greatest weak forward bisimulation equivalence gives better results than one obtained by means of the greatest forward bisimulation equivalence will be given(Example 4.3).

In the Section 4.3., weak bisimulations will be studied in conjunction with the concept of a uniform fuzzy relation, which has been introduced in [20], and further developed in [22]. In particular, it will be shown that two fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ are weak forward bisimulation equivalent, i.e., there is a uniform weak forward bisimulation between them, if and only if there is a weak forward isomorphism between the factor fuzzy automata with respect to the greatest weak forward bisimulation equivalences on $\mathscr{A}$ and $\mathscr{B}$ (Theorem 4.14). An analogous theorem can be proved for weak backward bisimulation equivalent fuzzy automata. Also, uniform weak backward and forward bisimulations between two fuzzy automata in terms of isomorphisms between their Nerode and reverse Nerode automata will be characterized(Theorem 4.13).

In the Section 4.4. the weak forward bisimulation equivalence between fuzzy automata will be discussed.

The results from this chapter are published in [64].

All results in this chapter are original and are closely associated with the results of Ćirić, Ignjatović and Damljanović [20, 21, 22].

4.1. Weak simulation for fuzzy automata

Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be a fuzzy automaton. For all $u \in X^*$, we define fuzzy sets $\sigma_u^A, \tau_u^A \in \mathscr{F}(A)$ as follows:

$$\sigma_u^A(a) = \bigvee_{b \in A} \sigma^A(b) \otimes \delta_u^A(b,a), \qquad \tau_u^A(a) = \bigvee_{b \in A} \delta_u^A(b) \otimes \tau^A(b,a),$$

i.e.,

$$\sigma_u^A = \sigma^A \circ \delta_u^A, \qquad \tau_u^A = \delta_u^A \circ \tau^A.$$

It is worth noting that fuzzy sets σ_u^A ($u \in X^*$) play the key role in the determinization of the fuzzy automaton $\mathscr{A}$ (cf., e.g., [53, 56]), and consequently, fuzzy sets τ_u^A ($u \in X^*$) are used in the determinization of the reverse fuzzy automaton of $\mathscr{A}$.

Moreover, for each $a \in A$ the *left fuzzy language of the state a* and the *right fuzzy language of the state a* are a fuzzy languages $\sigma_a, \tau_a : X^* \to L$ defined by:

$$\sigma_a^A(u) = \bigvee_{b\in A} \sigma^A(b) \circ \delta_u^A(b,a), \qquad \tau_a(u) = \bigvee_{b\in A} \delta_u^A(a,b) \otimes \tau(b) \qquad u \in X^*.$$

It can easily be shown that, for every $u \in X^*$ and $a \in A$:

$$\sigma_a(u) = \sigma_u(a), \qquad \tau_a(u) = \tau_u(a). \tag{4.1}$$

Further, let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata. A fuzzy relation $\varphi \in \mathscr{R}(A,B)$ which is a solution to the system of fuzzy relation inequalities:

$$\sigma^A \leqslant \sigma^B \circ \varphi^{-1} \tag{4.2}$$

$$\varphi^{-1} \circ \tau_u^A \leqslant \tau_u^B \qquad u \in X^* \tag{4.3}$$

is called a *weak forward simulation*, and if φ is a solution to

$$\tau^A \leqslant \varphi \circ \tau^B \tag{4.4}$$

$$\sigma_u^A \circ \varphi \leqslant \sigma_u^B \qquad u \in X^* \tag{4.5}$$

then it is called a *weak backward simulation*.

Weak forward and weak backward simulations are generalization of the notion of forward and backward simulations introduce in [22].

For the sake of simplicity, we will call φ just a weak simulation if it is either a weak forward or a weak backward simulation.

Lemma 4.1. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata and $\varphi \in \mathscr{R}(A,B)$ a fuzzy relation, then the following holds:*

If φ is a forward(resp. backward) simulation, then φ is a weak forward(resp. backward) simulation.

Proof. Let φ be a forward simulation, i.e. φ satisfies (*fs*-1)-(*fs*-3). Thus, (4.2) holds and for $u = \varepsilon$ (4.3) holds. Now, let (4.3) hold, for all words $u = x_1 \dots x_n \in X^*$ of length $n \in \mathbb{N}$. For $k = n+1$ consider a word $u \in X^*$ of length k, that is $u = vx$ where $x \in X$ and $v \in X^*$ is a word of length n. We have

$$\varphi^{-1} \circ \tau_u^A = \varphi^{-1} \circ \delta_x^A \circ \tau_v^A \leqslant \delta_x^B \circ \varphi^{-1} \circ \tau_v^A \leqslant \delta_x^B \circ \tau_v^B = \tau_{xv}^B = \tau_u^B.$$

Hence, by induction we have shown that φ is a weak forward simulation. □

Theorem 4.1. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata and $\varphi \in \mathscr{R}(A,B)$ a fuzzy relation. If φ is a weak simulation, then $\|A\| \leqslant \|B\|$.*

Proof. Let φ be a weak forward simulation between the automata $\mathscr{A}$ and $\mathscr{B}$.Then φ satisfies (4.2) and (4.3), so for every $u \in X^*$, we have

$$\|A\|(u) = \sigma^A \circ \delta^A_u \circ \tau^A = \sigma^A \circ \tau^A_u \leqslant \sigma^B \circ \varphi^{-1} \circ \tau^A_u \leqslant \sigma^B \circ \tau^B_u = \sigma^B \circ \delta^B_u \circ \tau^B = \|B\|(u)$$

Therefore, $\|A\|(u) \leqslant \|B\|(u)$ for any $u \in X^*$. □

The following lemma can be easily proved using the definition of weak forward and backward simulations and the reverse fuzzy automaton.

Lemma 4.2. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata. A fuzzy relation $\varphi \in \mathscr{R}(A,B)$ is a weak backward simulation between automata $\mathscr{A}$ and $\mathscr{B}$ if and only if φ is a weak forward simulation between the reverse fuzzy automata $\bar{\mathscr{A}}$ and $\bar{\mathscr{B}}$.*

From this lemma, we conclude that for any statement on weak forward simulations which is universally valid (for all fuzzy automata) there is the corresponding universally valid statement on weak backward simulations. Accordingly, we will deal only with weak forward simulations.

It can be easily shown that the following holds.

Lemma 4.3. *The composition of two weak forward simulations and the union of an arbitrary family of weak forward simulations is also weak forward simulation.*

The following theorem gives a way to decide whether there is a weak forward simulation between two automata, and whenever it exists, provides a method to construct the greatest one.

Theorem 4.2. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata, and let $\varphi \in \mathscr{R}(A,B)$ be a fuzzy relation defined by:*

$$\varphi(a,b) = \bigwedge_{u\in X^*} \tau^A_u(a) \to \tau^B_u(b)$$

for every $a \in A, b \in B$. If φ satisfies (4.2), then it is the greatest weak forward simulation from $\mathscr{A}$ to $\mathscr{B}$, otherwise, if φ does not satisfy (4.2), then there is no weak forward simulation from $\mathscr{A}$ to $\mathscr{B}$.

Proof. Let φ satisfies (4.2) and let $\psi \in \mathscr{R}(A,B)$ be arbitrary solution to (4.3). Then,

$$\bigvee_{a\in A} \psi^{-1}(b,a) \otimes \tau^A_u(a) \leqslant \tau^B_u(b) \qquad u \in X^*$$

that is,

$$\psi^{-1}(b,a) \otimes \tau^A_u(a) \leqslant \tau^B_u(b) \qquad u \in X^*$$

for each $a \in A$ and $b \in B$. By adjunction property, we have

$$\psi^{-1}(b,a) \leqslant \tau_u^A(a) \to \tau_u^B(b) \qquad u \in X^*$$

for each $a \in A$ and $b \in B$. So,

$$\psi^{-1}(b,a) \leqslant \bigwedge_{u \in X^*} \tau_u^A(a) \to \tau_u^B(b) = \varphi(a,b) \qquad a \in A, b \in B$$

i.e.,

$$\psi(a,b) \leqslant \varphi(a,b) \qquad a \in A, b \in B.$$

Thus, a fuzzy relation ψ is a solution to (4.3) if and only if $\psi \leqslant \varphi$. Accordingly, φ is the greatest solution to (4.3) and therefore it is the greatest weak forward simulation from $\mathscr{A}$ to $\mathscr{B}$.

Suppose that φ does not satisfy (4.2). And assume the opposite, that is, let ψ be an weak forward simulation from $\mathscr{A}$ to $\mathscr{B}$. But, then we have

$$\sigma^A \leqslant \sigma^B \circ \psi^{-1} \leqslant \sigma^B \circ \varphi^{-1}$$

which contradicts the fact φ does not satisfy (4.2). So, we conclude that there is no weak forward simulation from $\mathscr{A}$ to $\mathscr{B}$. □

By the Theorem 4.5, in order to compute the greatest weak forward simulation, we need to compute all fuzzy sets τ_u^A and τ_u^B, for $u \in X^*$. In other words, we need to determine all the states of the reverse Nerode automata $\bar{\mathscr{A}}_N$ and $\bar{\mathscr{B}}_N$, which can be exponential in the number of states of $\mathscr{A}$ and $\mathscr{B}$, or even infinite.

The next algorithm gives method for construction of the set $\{\tau_v, v \in X^*\}$:

Algorithm 4.3 (*Construction of the set* $\{\tau_v, v \in X^*\}$**)** The input of this algorithm is a fuzzy automaton $\mathscr{A} = (A, \delta, \sigma, \tau)$ over X and L, and the output is the set $\{\tau_v, v \in X^*\}$ of all the fuzzy sets τ_v, $v \in X^*$. It is constructed inductively in the following way:

(T1) First we put $\tau_\varepsilon = \tau$, and we set $T_0 = \{\tau_\varepsilon\}$.

(T2) After the i-th step let the set T_i have been constructed, and elements in S_i have been labeled either 'closed' or 'non-closed'. The meaning of these two terms will be made clear in the sequel,

(T3) In the succeeding step we make the set T_{i+1} by enriching the set T_i as follows: for every non-closed element τ_u occurring in T_i, where $u \in X^*$, and every $x \in X$ we add an element $\tau_{xu} = \delta_x \circ \tau_u$. In the case, τ_{xu} is an element that has already been constructed, we label τ_{xu} as *closed*. The procedure terminates when all elements are marked closed,

(T4) The resulting set is the set $\{\tau_v, v \in X^*\}$.

Although, this procedure works in exponential time, in many cases computation of the set $\{\tau_v, v \in X^*\}$ performs more quickly than its worst case complexity would suggest.

4.2. Weak bisimulation for fuzzy automata

Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata. A fuzzy relation $\varphi \in \mathscr{R}(A,B)$ is a *weak forward bisimulation* if both φ and φ^{-1} are weak forward simulations, i.e., if φ satisfies (4.2), (4.3) and

$$\sigma^B \leqslant \sigma^A \circ \varphi \tag{4.6}$$

$$\varphi \circ \tau^B_u \leqslant \tau^A_u \qquad u \in X^*, \tag{4.7}$$

and φ is called *weak backward bisimulation* if it satisfies (4.5), (4.4) and

$$\tau^B \leqslant \varphi^{-1} \circ \tau^A \tag{4.8}$$

$$\sigma^B_u \circ \varphi^{-1} \leqslant \sigma^A_u \qquad u \in X^* \tag{4.9}$$

i.e., if both φ and φ^{-1} are weak backward simulations.

These concepts generalize the notions of bisimulations for fuzzy automata [22].

The notions of weak simulations and bisimulations were used in a different context in the study of labeled transition systems with ε-transitions (or silent transitions). However, these concepts differ from our weak simulations and bisimulations. In fact, a weak simulation (bisimulation) between transition systems $\mathscr{A}$ and $\mathscr{B}$ is an ordinary (forward) simulation (bisimulation) between $\mathscr{A}$ and the transition system obtained from $\mathscr{B}$ by removing ε-transitions.

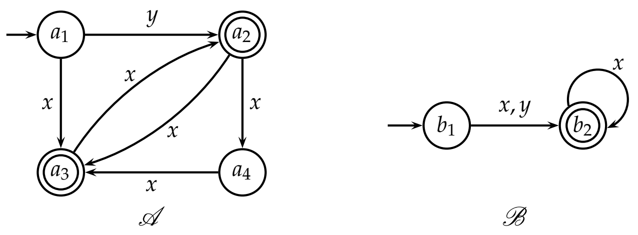
To simplify, we will call φ just a weak bisimulation if it is either a weak forward or a weak backward bisimulation.

The proof of the following lemma is similar to the proof of the Lemma 4.1 and it will be omitted.

Lemma 4.4. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata and $\varphi \in \mathscr{R}(A,B)$ a fuzzy relation, then if φ is a forward(resp. backward) bisimulation, then φ is a weak forward(resp. backward) bisimulation.*

However, the converse does not hold in general.

Example 4.1. Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be automata over Boolean structure with $|A| = 4$ and $|B| = 2$ and $X = \{x,y\}$, given by the following graphs:



In terms of Boolean matrices and vectors these automata are given by:

$$\sigma^A = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \quad \delta_x^A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \delta_y^A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

$$\sigma^B = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \delta_x^B = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad \delta_y^B = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \tau^B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Automata $\mathscr{A}$ and $\mathscr{B}$ are language equivalent, moreover $\|A\| = \|B\| = (x+y)x^*$. The fuzzy relation μ defined by the following matrix:

$$\mu = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

satisfies conditions (4.2), (4.3), (4.6) and (4.7).Hence μ is a weak forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$. Since μ does not satisfies (fb-2), it is not a forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$.

It is important to mention that according to the Lemma 4.2, for every assertion on weak forward bisimulations which is globally valid there is the corresponding globally valid assertion on weak backward bisimulations. Therefore, we will deal only with weak forward bisimulations.

Theorem 4.4. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata and $\varphi \in \mathscr{R}(A,B)$ a fuzzy relation. If φ is a weak bisimulation, then $\|A\| = \|B\|$.*

The proof of the previous Theorem is analogue to the proof of the Theorem 4.1.

It can be easily shown that the following holds.

Lemma 4.5. *The composition of two weak forward bisimulations and the union of an arbitrary family of weak forward bisimulations is also weak forward bisimulations.*

Moreover, the inverse of a weak forward bisimulation is a weak forward bisimulation.

According to this result, we have the following.

Lemma 4.6. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata such that there exists at least one weak forward bisimulation from $\mathscr{A}$ to $\mathscr{B}$. Then there exists the greatest weak forward bisimulation from $\mathscr{A}$ to $\mathscr{B}$, which is a partial fuzzy function.*

Proof. Let $\{\varphi_i\}_{i\in I} \in \mathscr{R}(A,B)$ be family of all weak forward bisimulations from $\mathscr{A}$ to $\mathscr{B}$. Denote $\varphi = \bigvee_{i\in I} \varphi_i$. According to the previous lemma, φ is also weak forward bisimulation, and it is the greatest one. In order to prove that φ is a partial fuzzy function, we will show $\varphi \circ \varphi^{-1} \circ \varphi \leqslant \varphi$. Denote $\eta = \varphi \circ \varphi^{-1} \circ \varphi$. Then for every $u \in X^*$ we have

$$\eta \circ \tau_u^B = \varphi \circ \varphi^{-1} \circ \varphi \circ \tau_u^B \leqslant \varphi \circ \varphi^{-1} \circ \tau_u^A \leqslant \varphi \circ \tau_u^B \leqslant \tau_u^A$$

And similarly, (4.2),(4.3) and (4.6) holds. Therefore, η is a weak forward bisimulation from $\mathscr{A}$ to $\mathscr{B}$. Hence, $\eta \leqslant \varphi$, and φ is a partial fuzzy function. □

The following theorem gives the procedure for computing the greatest weak forward bisimulations, whenever it exists.

Theorem 4.5. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata, and let $\varphi \in \mathscr{R}(A,B)$ be a fuzzy relation defined by:*

$$\varphi(a,b) = \bigwedge_{u \in X^*} \tau_u^A(a) \leftrightarrow \tau_u^B(b), \tag{4.10}$$

for every $a \in A, b \in B$. If φ satisfies (4.2) and (4.6), then it is the greatest weak forward bisimulation from $\mathscr{A}$ to $\mathscr{B}$, and it is a partial fuzzy function, otherwise, there is no weak forward bisimulation from $\mathscr{A}$ to $\mathscr{B}$.

Proof. This theorem can be proved similarly as Theorems 4.6 and 4.2. □

According to (4.1) and (4.11), the greatest weak forward bisimulation can be presented as:

$$\varphi(a,b) = \bigwedge_{u \in X^*} \tau_a^A(u) \leftrightarrow \tau_b^B(u). \tag{4.11}$$

Therefore, the greatest weak forward bisimulation between two fuzzy automata can be interpreted as a measure of the degrees of equality of the right languages.

Although the greatest weak bisimulations provide better approximation of the language-equivalence and better reductions than the ordinary bisimulations, their computation can be computationally hard. But in a lot of instances this method terminates in acceptable number of steps.

Example 4.2. Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be automata over the Boolean structure with $X = \{x\}$, $|A| = 4$ and $|B| = 2$. The fuzzy transition relations and fuzzy sets of initial and terminal states are given by the following Boolean matrices and vectors:

$$\sigma^A = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}, \quad \delta_x^A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \sigma^B = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \delta_x^B = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad \tau^B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

In order to compute the greatest weak forward bisimulation using the formula (4.10) we compute $\tau_u^A \leftrightarrow \tau_u^B$, for every $u \in X^*$:

$$\tau_e^A \leftrightarrow \tau_e^B = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \tau_{x^i}^A \leftrightarrow \tau_{x^i}^B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix},$$

for every $i \in N$. Therefore, we obtain:

$$\varphi = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Moreover, we can easily check that φ satisfies both (4.2) and (4.6). Now, by Theorem 4.5, φ is the greatest weak forward bisimulation between automata $\mathscr{A}$ and $\mathscr{B}$. On the other hand, using Algorithm 3.5 in Chapter 3, we obtain that there is no forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$. Since φ is complete and surjective (i.e., it is a uniform fuzzy relation), we have that $\mathscr{A}$ and $\mathscr{B}$ are WFB-equivalent, but they are not FB-equivalent. If we change σ^A and σ^B to

$$\sigma^A = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, \quad \sigma^B = \begin{bmatrix} 1 & 1 \end{bmatrix},$$

then the fuzzy relation φ does not satisfy (4.6). In this case there is no weak forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$. Nevertheless, for any $u \in X^*$ we have

$$\|A\|(u) = \|B\|(u) = \begin{cases} 1 & \text{if } u = \varepsilon, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ are language-equivalent.

In the previous example, for given fuzzy automata, there exists a weak forward bisimulation between them, i.e. the fuzzy automata are language equivalent, whereas there is no forward bisimulation. In other words, weak forward bisimulations are better means for modeling the language equivalence between fuzzy automata than forward bisimulations.

Next, consider an arbitrary fuzzy automaton $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$. A weak forward bisimulation from $\mathscr{A}$ into itself will be called a *weak forward bisimulation on $\mathscr{A}$* (analogously we define *weak backward bisimulations on $\mathscr{A}$*). Since the equality relation is a weak forward bisimulation on $\mathscr{A}$, the set of all weak forward bisimulations on $\mathscr{A}$ is non-empty. Moreover, according to Theorem 4.5 there is the greatest weak forward bisimulation on A, and we can easily show that it is a fuzzy equivalence.

Weak forward bisimulations on $\mathscr{A}$ which are equivalences will be called weak forward bisimulation equivalences. The set of all weak forward bisimulation equivalences on $\mathscr{A}$ we denote by $\mathscr{E}^{wfb}(\mathscr{A})$.

Theorem 4.6. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be a fuzzy automaton. A fuzzy equivalence E on A is a weak forward bisimulation on $\mathscr{A}$ if and only if:*

$$E \circ \tau_u^A \leqslant \tau_u^A, \qquad u \in X^*, \tag{4.12}$$

or equivalently,

$$E \circ \tau_u^A = \tau_u^A, \qquad u \in X^*.$$

Proof. Let E be a fuzzy equivalence on A, which satisfies (4.12). Obviously, E satisfies (4.7). E is a reflexive and therefore it satisfies (4.2) and (4.6). Moreover, since E is a symmetric fuzzy relation we can easily show that (4.3) holds. Therefore, E is a weak forward bisimulation on $\mathscr{A}$.

Conversely, if E is a weak forward bisimulation equivalence then (4.12) holds. □

In an analogous way as in the previous theorem, we can show that a fuzzy equivalence E on A is a weak backward bisimulation on $\mathscr{A}$ if and only if the following holds:

$$\sigma_u^A \circ E \leqslant \sigma_u^A, \qquad u \in X^*, \tag{4.13}$$

or equivalently,

$$\sigma_u^A \circ E = \sigma_u^A, \qquad u \in X^*. \tag{4.14}$$

Theorem 4.7. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be a fuzzy automaton.*

The set $\mathscr{E}^{wfb}(\mathscr{A})$ of all weak forward bisimulation fuzzy equivalence relations on $\mathscr{A}$ forms a principal ideal of the lattice $\mathscr{E}(A)$ of all fuzzy equivalences on A generated by the relation E^{wfb} on A defined by

$$E^{wfb}(a,b) = \bigwedge_{u \in X^*} \tau_u^A(a) \leftrightarrow \tau_u^A(b), \qquad a, b \in A. \tag{4.15}$$

Proof. Evidently, E^{wfb} is a fuzzy equivalence. Also, according to Theorem 4.5, E^{wfb} is the greatest forward bisimulation from $\mathscr{A}$ into itself. Next, let $E \in \mathscr{E}(A)$ such that $E \leqslant E^{wfb}$. Then for every $u \in X^*$, we have that $E \circ \tau_u^A \leqslant E^{wfb} \circ \tau_u^A \leqslant \tau_u^A$. Hence, $E \in \mathscr{E}^{wfb}(\mathscr{A})$.

Conversely, let $E \in \mathscr{E}^{wfb}(\mathscr{A})$. Hence, E satisfies (4.12). In a similar way, as in the proof of Theorem 4.2, we show that (4.12) is equivalent to $E \leqslant E^{wfb}$. Therefore, $E \leqslant E^{wfb}$ if and only if $E \in \mathscr{E}^{wfb}(\mathscr{A})$. □

The next example shows that there are automata whose greatest weak forward bisimulation equivalence is strictly greater than the greatest forward bisimulation equivalence and the reduction by means of this equivalence gives better results than one obtained by means of the greatest forward bisimulation equivalence.

Example 4.3. Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be an automaton over the Boolean structure with $X = \{x, y\}$ and $|A| = 5$. The fuzzy transition relation and fuzzy sets of initial and terminal states are given by the following Boolean matrices and vectors:

$$\sigma^A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad \delta_x^A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad \delta_y^A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

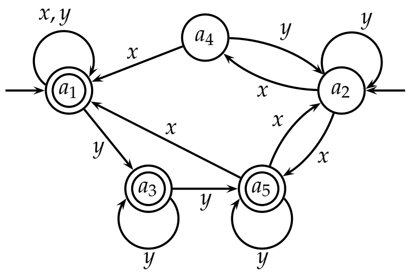
Computing the fuzzy relation $E \in \mathscr{R}(A)$ using formula (4.15), we obtain:

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix},$$

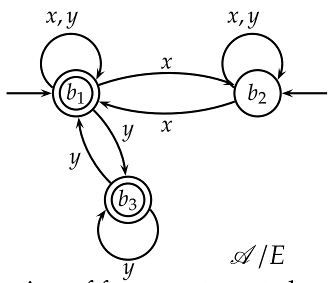
and by Theorem 4.7, E is the greatest weak forward bisimulation equivalence on $\mathscr{A}$. However, the greatest forward bisimulation equivalence on $\mathscr{A}$ is a fuzzy relation $F \in \mathscr{R}(A)$ given by:

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Clearly, E is strictly larger than F.



Thus, the first and the fifth state of $\mathscr{A}$, as well as the second and the fourth state, are equivalent w.r.t. the greatest weak forward bisimulation E, but there are no states equivalent w.r.t. the greatest forward bisimulation F. This means that the greatest weak forward bisimulation reduce the number of states of $\mathscr{A}$ to three states, whereas the greatest forward bisimulation does not reduce the number of states.



$\mathscr{A}/E$

Although the reduction of fuzzy automata by means of the greatest weak forward bisimulations is in general better than one obtained by means of the greatest forward bisimulations, this reductions does not produce a minimal fuzzy automata.

Example 4.4. Consider automaton $\mathscr{A}$ from Example 4.1. The greatest weak bisimulation on $\mathscr{A}$ is given by the following matrix:

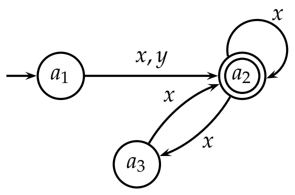
$$\varphi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The reduction of the automaton $\mathscr{A}$ by means of φ gives the factor fuzzy automaton $\mathscr{A}/\varphi$ that has three states.

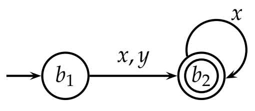
Consider now, the equivalence relation E on $\mathscr{A}$ defined by:

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

The reduction of the automaton $\mathscr{A}$ by means of this equivalence gives the factor fuzzy automaton $\mathscr{A}/E$ that has two states and $\|A\| = L(\mathscr{A}/E)$ ($\mathscr{A}/E$ is equal to automaton $\mathscr{B}$ in Example 4.1).



$\mathscr{A}/\varphi$



$\mathscr{A}/E$

As previous example suggests there exist fuzzy automata where reduction by means of some other fuzzy equivalences gives better result then one obtained by means of the greatest weak forward bisimulation.

4.3. Uniform weak bisimulations

In this section, we will consider weak forward and backward bisimulations which are uniform fuzzy relations.

Theorem 4.8. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata, and let $\varphi \in \mathscr{R}(A, B)$ be a uniform fuzzy relation. Then, φ is a weak forward bisimulation if and only if following holds:*

$$\sigma^A \circ \varphi \circ \varphi^{-1} = \sigma^B \circ \varphi^{-1}, \qquad \sigma^B \circ \varphi^{-1} \circ \varphi = \sigma^A \circ \varphi, \tag{4.16}$$

$$\varphi^{-1} \circ \tau_u^A = \tau_u^B, \qquad \varphi \circ \tau_u^B = \tau_u^A, \quad u \in X^*. \tag{4.17}$$

Proof. Let φ be a weak forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$. By (4.2) and (4.6) we have that:

$$\sigma^A \circ \varphi \circ \varphi^{-1} \leqslant \sigma^B \circ \varphi^{-1} \circ \varphi \circ \varphi^{-1} = \sigma^B \circ \varphi^{-1} \leqslant \sigma^A \circ \varphi \circ \varphi^{-1},$$

so $\sigma^A \circ \varphi \circ \varphi^{-1} = \sigma^B \circ \varphi^{-1}$. Analogously, $\sigma^B \circ \varphi^{-1} \circ \varphi = \sigma^A \circ \varphi$. Moreover, as $\varphi^{-1} \circ \varphi$ is reflexive, for each $u \in X^*$ we have that

$$\tau_u^A \leqslant \varphi \circ \varphi^{-1} \circ \tau_u^A \leqslant \varphi \circ \tau_u^B \leqslant \tau_u^A,$$

and hence, $\varphi \circ \tau_u^B = \tau_u^A$. In a similar way we show that $\varphi^{-1} \circ \tau_u^A = \tau_u^B$, for each $u \in X^*$.

Conversely, let (4.16) and (4.17) hold. According to reflexivity of $\varphi \circ \varphi^{-1}$ and $\varphi^{-1} \circ \varphi$ we have

$$\sigma^A \leqslant \sigma^A \circ \varphi \circ \varphi^{-1} = \sigma^B \circ \varphi^{-1}, \qquad \sigma^B \leqslant \sigma^B \circ \varphi^{-1} \circ \varphi = \sigma^A \circ \varphi.$$

Therefore, (4.2) and (4.6) hold. Further, directly from (4.17), we conclude that (4.3) and (4.7) hold. Thus φ is a weak forward bisimulation. □

Lemma 4.7. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be a fuzzy automaton, let E be a fuzzy equivalence on $\mathscr{A}$, and $\mathscr{A}/E = (A/E, \delta^{A/E}, \sigma^{A/E}, \tau^{A/E})$ be the factor fuzzy automaton of $\mathscr{A}$ w.r.t E. If E is a weak forward bisimulation, then*

$$\tau_u^{A/E}(E_a) = \tau_u^A(a), \qquad u \in X^*, a \in A. \tag{4.18}$$

Proof. This follows immediately from the definition of $\tau_u^{A/E}$ and the fact that E is a weak forward bisimulation equivalence on $\mathscr{A}$. □

Theorem 4.9. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ be a fuzzy automaton, E a fuzzy equivalence on $\mathscr{A}$, φ_E the natural function from A to A/E and $\mathscr{A}/E = (A/E, \delta^{A/E}, \sigma^{A/E}, \tau^{A/E})$ the factor fuzzy automaton of $\mathscr{A}$ w.r.t E.*

Then, E is a weak forward bisimulation equivalence on $\mathscr{A}$ if and only if φ_E is a weak forward bisimulation.

Proof. Let E be a weak forward bisimulation equivalence on $\mathscr{A}$. According to the previous lemma, for arbitrary $a \in A$ and $u \in X^*$ we have:

$$\varphi_E \circ \tau_u^{A/E}(a) = \bigvee_{b \in A} \varphi_E(a, E_b) \otimes \tau_u^{A/E}(E_b) = \bigvee_{b \in A} E(a,b) \otimes \tau_u^A(b) = E \circ \tau_u^A(a) = \tau_u^A(a),$$

and also,

$$\begin{aligned}\sigma^A(a) \leqslant \sigma^A \circ E \circ E(a) &= \bigvee_{b \in A} \sigma^A \circ E(b) \otimes E(b,a) = \bigvee_{b \in A} \sigma^{A/E}(E_b) \otimes \varphi_E(a, E_b) = \\ &= \bigvee_{b \in A} \sigma^{A/E}(E_b) \otimes \varphi_E^{-1}(E_b, a) = \sigma^{A/E} \circ \varphi_E^{-1}(a).\end{aligned}$$

In a similar way we prove that $\varphi_E^{-1} \circ \tau_u^A \leqslant \tau_u^{A/E}$, for each $u \in X^*$, and $\sigma^{A/E} \leqslant \sigma^A \circ \varphi_E$. Thus, φ_E is a weak forward bisimulation.

Conversely, let φ_E be a weak forward bisimulation. Now, for arbitrary $u \in X^*$ and $a \in A$ we have that

$$E \circ \tau_u^A(a) = \tau_u^{A/E}(E_a) = \varphi_E(a, E_a) \otimes \tau_u^{A/E}(E_a) \leqslant \varphi_E \circ \tau_u^{A/E}(a) = \tau_u^A(a).$$

Therefore, E is a weak forward bisimulation equivalence on $\mathscr{A}$. □

Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata. A bijective function ϕ of A onto B is called a *weak forward isomorphism* of fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ if

$$\sigma^A(a) = \sigma^B(\phi(a)), \qquad a \in A, \tag{4.19}$$

$$\tau_u^A(a) = \tau_u^B(\phi(a)), \qquad a \in A, u \in X^*, \tag{4.20}$$

and a *weak backward isomorphism* of fuzzy automata $\mathscr{A}$ and $\mathscr{B}$ if

$$\sigma_u^A(a) = \sigma_u^B(\phi(a)), \qquad a \in A, u \in X^*, \tag{4.21}$$

$$\tau^A(a) = \tau^B(\phi(a)), \qquad a \in A. \tag{4.22}$$

The notion of a weak forward(backward) isomorphism generalizes the notion of an isomorphism between fuzzy automata, that is, the following is true:

Lemma 4.8. *Any isomorphism between two fuzzy automata is also a weak forward and a weak backward isomorphism between these automata.*

Proof. Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$, and let $\phi : A \to B$ be an isomorphism between $\mathscr{A}$ and $\mathscr{B}$. Then ϕ satisfies (4.19). Next, according to (2.7) and the fact that ϕ is a bijection from A to B, for every $u \in X^x$ and $a \in A$, we have:

$$\tau_u^A(a) = \delta_u^A \circ \tau^A(a) = \bigvee_{c \in A} (\delta_u^A(c,a) \otimes \tau^A(a)) = \bigvee_{c \in A} (\delta_u^B(\phi(c), \phi(a)) \otimes \tau^B(\phi(a))) =$$
$$= \bigvee_{\phi(c) \in B} (\delta_u^B(\phi(c), \phi(a)) \otimes \tau^B(\phi(a))) = \delta_u^B \circ \tau^B(\phi(a)) = \tau_u^B(\phi(a)).$$

Hence, ϕ satisfies (4.20) and consequently ϕ is a weak forward isomorphism from $\mathscr{A}$ to $\mathscr{B}$. In an analogous way, we prove that ϕ is a weak backward isomorphism. □

Moreover, we have:

Lemma 4.9. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$. If there exists a weak forward isomorphism of fuzzy automata $\mathscr{A}$ and $\mathscr{B}$, then $\|A\| = \|B\|$.*

Proof. Let $\phi : A \to B$ be a weak forward isomorphism. Then for every $u \in X^*$ we have

$$L(A)(u) = \sigma^A \circ \tau_u^A = \bigvee_{a \in A} \sigma^A(a) \otimes \tau_u^A(a) = \bigvee_{a \in A} \sigma^B(\phi(a)) \otimes \tau_u^B(\phi(a)) =$$
$$= \bigvee_{b \in B} \sigma^B(b) \otimes \tau_u^B(b) = \sigma^B \circ \tau_u^B = L(B)(u),$$

which was to be proved. □

We can easily prove the following lemma.

Lemma 4.10. *The composition of weak forward isomorphisms is a weak forward isomorphism, and a inverse of the weak forward isomorphism is a weak forward isomorphism.*

Lemma 4.11. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$, $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata. If $\mathscr{A}$ and $\mathscr{B}$ are weak forward isomorphic, then there exists a uniform weak forward bisimulation from $\mathscr{A}$ to $\mathscr{B}$.*

Proof. Let $\phi : A \to B$ be an weak forward isomorphism from $\mathscr{A}$ to $\mathscr{B}$. Define a fuzzy relation $\varphi \in \mathscr{R}(A, B)$ by:

$$\varphi(a,b) = \begin{cases} 1, & \text{if } b = \phi(a) \\ 0, & \text{otherwise.} \end{cases}$$

It can be easily shown that φ is a surjective $\mathscr{L}$-function. Next, for every $a \in A$ we have the following:

$$\sigma^A(a) = \sigma^B(\phi(a)) = \sigma^B(\phi(a)) \otimes \varphi^{-1}(\phi(a),a) \leqslant \sigma^B \circ \varphi^{-1}(a),$$

and thus, φ satisfies (4.2). According to the definition of φ, and (1.7), for every $u \in X^*$ and $b \in B$, the following holds:

$$\varphi^{-1} \circ \tau_u^A(b) = \bigvee_{a\in A} (\varphi^{-1}(b,a) \otimes \tau_u^A(a)) = \varphi^{-1}(b,\phi^{-1}(b)) \otimes \tau_u^A(\phi^{-1}(b)) =$$
$$= \tau_u^A(\phi^{-1}(b)) = \tau_u^B(\phi(\phi^{-1}(b))) = \tau_u^B(b),$$

and hence, φ satisfies (4.3). Similarly we prove that (4.6) and (4.7) hold. Therefore, φ is a weak forward bisimulation. Now, by Theorem 4.6, there exists the greatest weak forward bisimulation ξ from $\mathscr{A}$ to $\mathscr{B}$, which is a partial fuzzy function. Since φ is a surjective $\mathscr{L}$-function and $\varphi \leqslant \xi$, then ξ is also a surjective $\mathscr{L}$-function. Whence, ξ is a uniform weak forward bisimulation. □

Theorem 4.10. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$, $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata, and $\varphi \in \mathscr{R}(A,B)$ be a uniform fuzzy relation. Then φ is a weak forward bisimulation if and only if following is true:*

(1)*E_A^φ is a weak forward bisimulation equivalence on the fuzzy automaton $\mathscr{A}$;*
(2)*E_B^φ is a weak forward bisimulation equivalence on the fuzzy automaton $\mathscr{B}$;*
(3)*$\tilde{\varphi}$ is an weak forward isomorphism of factor fuzzy automata $\mathscr{A}/E_A^\varphi$ and $\mathscr{B}/E_B^\varphi$.*

Proof. For the sake of simplicity set $E = E_A^\varphi$ and $F = E_B^\varphi$. Also, let $\psi \in CR(\varphi)$ be an arbitrary crisp description of φ.

Let φ be a uniform weak forward bisimulation. By (iv) and (v) of Theorem 1.9, we have that $E = \varphi \circ \varphi^{-1}$ and $F = \varphi^{-1} \circ \varphi$. Now,

$$\tau_u^A \leqslant \varphi \circ \varphi^{-1} \circ \tau_u^A \leqslant \varphi \circ \tau_u^B \leqslant \tau_u^A,$$

so $E \circ \tau_u^A = \tau_u^A$ for each $u \in X^*$, i.e., E is a weak forward bisimulation fuzzy equivalence. Similarly we prove that $F \circ \tau_u^B = \tau_u^B$, for each $u \in X^*$, therefore (2) also holds. According to Lemma 1.10, $\tilde{\varphi}$ is a bijective function of A/E onto B/F. By Theorem 4.8, we have

$$\sigma^{A/E}(E_a) = \sigma^A \circ E(a) = \sigma^A \circ \varphi \circ \varphi^{-1}(a) = \sigma^B \circ \varphi^{-1}(a)$$
$$= \bigvee_{b\in B} \sigma(b) \otimes \varphi(a,b) = \bigvee_{b\in B} \sigma^B(b) \otimes F(\psi(a),b) = \sigma^{B/F}(F_{\psi(a)}),$$

and hence, $\sigma^{A/E}(E_a) = \sigma^{B/F}(F_{\psi(a)}) = \sigma^{B/F}(\tilde{\varphi}(E_a))$. Also, the following holds

$$\tau_u^{A/E}(E_a) = \tau_u^A(a) = \varphi \circ \tau_u^B(a) = \bigvee_{b\in B} F(\psi(a),b) \otimes \tau_u^B(b) =$$
$$= F_{\psi(a)} \circ \tau_u^B = \tau_u^{B/F}(F_{\psi(a)}) = \tau_u^{B/F}(\tilde{\varphi}(E_a)).$$

Therefore, $\tilde{\varphi}$ is a weak forward isomorphism of factor fuzzy automata $\mathscr{A}/E_A^{\varphi}$ and $\mathscr{B}/E_B^{\varphi}$.

Conversely, let (1),(2) and (3) hold. Then,

$$\begin{aligned}\sigma^A(a) &\leqslant \sigma^A \circ E(a) = \sigma^{A/E}(E_a) = \sigma^{B/F}(\tilde{\varphi}(E_a)) = \sigma^{B/F}(F_{\psi(a)}) = \\ &= \bigvee_{b\in B} \sigma^B(b) \otimes F(\psi(a),b) = \bigvee_{b\in B} \sigma^B(b)\otimes\varphi(a,b) = \sigma^B\circ\varphi^{-1}(a),\end{aligned}$$

and similarly, $\sigma^B \leqslant \sigma^A \circ \varphi$.

Now, for arbitrary $u\in X^*$ and $a\in A$ we have

$$\begin{aligned}\varphi\circ\tau_u^B(a) &= \bigvee_{b\in B}\varphi(a,b)\otimes\tau_u^B(b) = \bigvee_{b\in B} F(\psi(a),b)\otimes\tau_u^B(b) = \\ &= \tau_u^{B/F}(F_{\psi(a)}) = \tau_u^{B/F}(\tilde{\varphi}(E_a)) = \tau_u^{A/E}(E_a) = \tau_u^A(a).\end{aligned}$$

Therefore, $\varphi\circ\tau_u^B = \tau_u^A$, which yields $\varphi^{-1}\circ\tau_u^A = \varphi^{-1}\circ\varphi\circ\tau_u^B = F\circ\tau_u^B = \tau_u^B$, and hence, φ is a weak forward bisimulation. □

Theorem 4.11. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$, $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata, and let E be a weak forward bisimulation fuzzy equivalence on $\mathscr{A}$ and F a weak forward bisimulation fuzzy equivalence on $\mathscr{B}$.*
Then there exists a uniform weak forward bisimulation $\varphi\in\mathscr{R}(A,B)$ such that

$$E_A^{\varphi} = E \qquad \textit{and} \qquad E_B^{\varphi} = F, \tag{4.23}$$

if and only if there exists a weak forward isomorphism $\phi : \mathscr{A}/E \to \mathscr{B}/F$ such that for every $a_1,a_2\in A$ we have

$$\tilde{E}(E_{a_1},E_{a_2}) = \tilde{F}(\phi(E_{a_1}),\phi(E_{a_1})). \tag{4.24}$$

Proof. The proof of this theorem is similar to the proof of Theorem 6.4 in ([22]), and it will be omitted. □

Theorem 4.12. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ be a fuzzy automaton, let E be a weak forward bisimulation equivalence on $\mathscr{A}$ and F a fuzzy equivalences on $\mathscr{A}$ such that $E\leqslant F$. Then F is a weak forward bisimulation equivalence on $\mathscr{A}$ if and only if F/E is a weak forward bisimulation equivalence on $\mathscr{A}/E$.*

Proof. Let E be a weak forward bisimulation equivalence on A. Then, according to the definition of F/E and Lemma 4.7, for every $a\in A$ and $u\in X^*$ we have

$$F\circ\tau_u^A(a) = F/E\circ\tau_u^{A/E}(E_a)\leqslant\tau_u^{A/E}(E_a) = \tau_u^A(a).$$

Therefore, we obtain that $(F/E)\circ\tau_u^{A/E}\leqslant\tau_u^{A/E}$ if and only if $F\circ\tau_u^A\leqslant\tau_u^A$, which was to be proved. □

Corollary 4.1. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ be an automaton, and let E and F be weak forward bisimulation equivalences on A such that $E \leqslant F$. Then F is the greatest weak forward bisimulation equivalence on A if and only if F/E is the greatest weak forward bisimulation equivalence on A/E.*

Proof. This is a direct consequence of the previous theorem and Theorem 2.2. □

Recall now, that the Nerode automaton of a given fuzzy automaton $\mathscr{A} = (A,\delta,\sigma,\tau)$ is a fuzzy automaton $\mathscr{A}_N = (A_N,\delta^N,\sigma_\varepsilon^A,\tau^N)$.

The *reverse Nerode automaton* of a fuzzy automaton $\mathscr{A}$ is an automaton $\tilde{\mathscr{A}}_N = (\tilde{A}_N,\delta^{\tilde{A}_N},\tau_\varepsilon^A,\tau^{\tilde{A}_N})$, where $\tilde{A}_N = \{\tau_u^A | u \in X^*\}$, and $\delta^{\tilde{A}_N} : A_N \times X \longrightarrow A_N$ and $\tau^{\tilde{A}_N} \in \mathscr{F}(A_N)$ are defined with

$$\delta^{\tilde{A}_N}(\tau_u^A,x) = \tau_{ux}^A, \qquad \tau^{A_N}(\tau_u^A) = \sigma^A \circ \tau_u^A,$$

for every $u \in X^*$ and $x \in X$. The automaton $\tilde{\mathscr{A}}_N$ is a crisp-deterministic fuzzy automaton which is isomorphic to the Nerode automaton of the reverse automaton $\tilde{\mathscr{A}}$ of $\mathscr{A}$.

The next theorem gives a characterization of uniform weak forward bisimulations in terms of the reverse Nerode automata. Moreover, in a similar way, we can characterize uniform weak backward bisimulations in terms of the Nerode automata.

Theorem 4.13. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata, and let $\varphi \in \mathscr{R}(A,B)$ be a uniform fuzzy relation. Then φ is a weak forward bisimulation from $\mathscr{A}$ to $\mathscr{A}$ if and only if it satisfies (4.2) and (4.6), and functions*

$$\tau_u^A \mapsto \varphi^{-1} \circ \tau_u^A, \quad \tau_u^B \mapsto \varphi \circ \tau_u^B, \quad u \in X^*,$$

are mutually inverse isomorphisms between reverse Nerode automata $\bar{\mathscr{A}}_N$ and $\bar{\mathscr{B}}_N$.

Proof. Consider mappings $\eta : \bar{A}_N \to \bar{B}_N$ and $\phi : \bar{B}_N \to \bar{A}_N$, defined by $\eta(\tau_u^A) = \varphi^{-1} \circ \tau_u^A$ and $\phi(\tau_u^B) = \varphi \circ \tau_u^B$ for every $u \in X^*$.

Let φ be a weak forward bisimulation from $\mathscr{A}$ to $\mathscr{B}$. Clearly, 4.2) and (4.6) hold. According to Theorem 4.8, for every $u \in X^*$, $\eta(\tau_u^A) = \tau_u^B \in \bar{B}_N$ and $\phi(\tau_u^B) = \tau_u^A \in \bar{A}_N$. Thus, η maps $\bar{A}_N$ into $\bar{B}_N$, and ϕ maps $\bar{B}_N$ into $\bar{A}_N$. Also, by the same theorem, for every $u \in X^*$, $\eta(\phi(\tau_u^B)) = \varphi^{-1} \circ \varphi \circ \tau_u^B = \tau_u^B$ and $\phi(\eta(\tau_u^B)) = \varphi \circ \varphi^{-1} \circ \tau_u^A = \tau_u^A$. So, η and φ are mutually inverse bijective mappings between $\bar{A}_N$ and $\bar{B}_N$.

Moreover, directly from the definition of η, we have $\eta(\tau^A) = \tau^B$.

Also, for every $u \in X^*$ and $x \in X$ we have that:

$$\eta(\delta_{\bar{A}_\sigma}(\tau_u^A,x)) = \eta(\tau_{xu}^A) = \varphi^{-1} \circ \tau_{xu}^A = \tau_{xu}^B = \delta_{\bar{B}_\sigma}(\tau_u^B,x) = \delta_{\bar{B}_\sigma}(\eta(\tau_u^A),x).$$

Next, according to Lemma 4.9, the following holds

$$\tau_{\bar{A}_\sigma}(\tau_u^A) = \sigma^A \circ \tau_u^A = L(A)(u) = L(B)(u) = \sigma^B \circ \tau_u^B = \tau_{\bar{B}_\sigma}(\tau_u^B) = \tau_{\bar{B}_\sigma}(\eta(\tau_u^A)).$$

Hence, η is an isomorphism of the reverse Nerode automata $\bar{\mathscr{A}}_N$ and $\bar{\mathscr{B}}_N$. Similarly we prove that ϕ is an isomorphism from $\bar{\mathscr{B}}_N$ to $\bar{\mathscr{A}}_N$.

Conversely, let (4.2) and (4.6) hold, and let η and ϕ be mutually inverse isomorphisms between $\bar{\mathscr{A}}_N$ and $\bar{\mathscr{B}}_N$.

Since τ^A and τ^B are the unique initial states of $\bar{\mathscr{A}}_N$ and $\bar{\mathscr{B}}_N$, we have that $\eta(\tau^A) = \tau^B$ and $\phi(\tau^A) = \tau^B$. Thus, $\varphi^{-1} \circ \tau^A = \tau^B$ and $\varphi \circ \tau^B = \tau^A$.

Suppose that $\eta(\tau_u^A) = \tau_u^B$, for every $u \in X^*$, of length n. Consider now, $v \in X^*$, of length $n+1$. Assume that $v = xu$, where $u \in X^*$ is a word of length n, and $x \in X$. Then,

$$\eta(\tau_{xu}^A) = \eta(\delta_{\bar{A}_\sigma}(\tau_u^A, x)) = \delta_{\bar{B}_\sigma}(\eta(\tau_u^A), x) = \delta_{\bar{B}_\sigma}(\tau_u^B, x) = \tau_{xu}^B.$$

Hence, by induction we have $\eta(\tau_u^A) = \tau_u^B$, for every $u \in X^*$. Similarly, $\phi(\tau_u^B) = \tau_u^A$, for every $u \in X^*$, which means that (4.17) holds. Therefore, according to Theorem 4.8 we conclude that φ is a weak forward bisimulation. □

4.4. Weak forward bisimulation equivalent automata

Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata. If there exists a complete and surjective weak forward bisimulation from $\mathscr{A}$ to $\mathscr{B}$, then we say that $\mathscr{A}$ and $\mathscr{B}$ are weak forward bisimulation equivalent, or briefly WFB-equivalent, and we write $\mathscr{A} \sim_{WFB} \mathscr{B}$. Note that surjectivity and completeness of this forward bisimulation means that every state of A is equivalent to some state of B, and vice versa. It is also worth noting, that if there exists an weak forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$, which is complete and surjective, then the greatest weak forward bisimulation between $\mathscr{A}$ and $\mathscr{B}$ have the same property, and according to Theorem 4.6, it is a uniform weak forward bisimulation.

For arbitrary fuzzy automata $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$ we have the following:

$$\begin{gathered} \mathscr{A} \sim_{WFB} \mathscr{A}; \quad \mathscr{A} \sim_{WFB} \mathscr{B} \Rightarrow \mathscr{B} \sim_{WFB} \mathscr{A}; \\ (\mathscr{A} \sim_{WFB} \mathscr{B} \wedge \mathscr{B} \sim_{WFB} \mathscr{C}) \Rightarrow \mathscr{A} \sim_{WFB} \mathscr{C}. \end{gathered} \tag{4.25}$$

Analogously, $\mathscr{A}$ and $\mathscr{B}$ are weak backward bisimulation equivalent, briefly WBB-equivalent, in notation $\mathscr{A} \sim_{WBB} \mathscr{B}$, if there exists a complete and surjective weak backward bisimulation from $\mathscr{A}$ to $\mathscr{B}$.

Next, we prove the following useful lemma.

Lemma 4.12. *Let $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ and $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ be fuzzy automata and let ϕ be an weak forward isomorphism from $\mathscr{A}$ to $\mathscr{B}$. Let E and F be the greatest weak forward bisimulation equivalences on $\mathscr{A}$ and $\mathscr{B}$, respectively.*

Then for every $a, b \in A$ the following holds:

$$E(a,b) = F(\phi(a),\phi(b)).$$

Proof. By the definition of the greatest weak forward bisimulation equivalences, and a weak forward isomorphism we have:

$$E(a,b) = \bigwedge_{u\in X^*} \tau_u^A(a) \leftrightarrow \tau_u^A(b) = \bigwedge_{u\in X^*} \tau_u^B(\phi(a)) \leftrightarrow \tau_u^B(\phi(b)) = F(\phi(a),\phi(b)),$$

for every $a,b \in X^*$. □

Theorem 4.14. *Let $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ and $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ be fuzzy automata, and let E and F be the greatest weak forward bisimulation equivalences on $\mathscr{A}$ and $\mathscr{B}$. Then $\mathscr{A}$ and $\mathscr{B}$ are WFB-equivalent if and only if there exists a weak forward isomorphism between factor automata $\mathscr{A}/E$ and $\mathscr{B}/F$.*

Proof. Let $\mathscr{A}$ and $\mathscr{B}$ be WFB-equivalent automata, that is, let there exists a complete and surjective weak forward bisimulation ϕ from $\mathscr{A}$ to $\mathscr{B}$. Now, by Theorem 4.6, there exists the greatest weak forward bisimulation φ from $\mathscr{A}$ to $\mathscr{B}$, which is a partial fuzzy function. Since ϕ is complete and surjective, and $\phi \leqslant \varphi$, then φ is also complete and surjective. Hence, φ is a uniform weak forward bisimulation.

Now, according to Theorem 4.10, E_A^φ and E_B^φ are weak forward bisimulation equivalences on $\mathscr{A}$ and $\mathscr{B}$, respectively, and $\tilde{\varphi}$ is a weak forward isomorphism of factor automata $\mathscr{A}/E_A^\varphi$ and $\mathscr{B}/E_B^\varphi$.

Let P and Q be respectively the greatest weak forward bisimulation equivalences on $\mathscr{A}/E_A^\varphi$ and $\mathscr{B}/E_B^\varphi$. Define a function $\xi : (\mathscr{A}/E_A^\varphi)/P \to (\mathscr{B}/E_B^\varphi)/Q$, by $\xi(P_a) = Q_{\tilde{\varphi}(a)}$ for every $a \in E_A^\varphi$. Using Lemma 4.12, it is easy to prove that ξ is a well-defined bijective function and according to (4.18), (4.25) and the fact that $\tilde{\varphi}$ is a weak forward isomorphism, we obtain that ξ is a weak forward isomorphism.

By Corollary 4.1 it follows that $P = E/E_A^\varphi$ and $Q = F/E_B^\varphi$, and according to Theorem 2.1, $\mathscr{A}/E$ is isomorphic to $(\mathscr{A}/E_A^\varphi)/P$ and $\mathscr{B}/F$ is isomorphic to $(\mathscr{B}/E_B^\varphi)/Q$, so $\mathscr{A}/E$ is isomorphic to $\mathscr{B}/F$. According to Lemma 4.8 we obtain that $\mathscr{A}/E$ is weak forward isomorphic to $\mathscr{B}/F$.

Conversely, if there exists an weak forward isomorphism from $\mathscr{A}/E$ to $\mathscr{B}/F$, then according to Lemma 4.11, $\mathscr{A}/E$ is WFB-equivalent to $\mathscr{B}/F$. Also, by Theorem 4.9, $\mathscr{A}$ and $\mathscr{A}/E$ are WFB-equivalent and $\mathscr{B}$ and $\mathscr{B}/F$ are WFB-equivalent. Now, according to (4.25), $\mathscr{A}$ is WFB-equivalent to $\mathscr{B}$. □

Corollary 4.2. *Let $\mathscr{A}$ be a fuzzy automaton, let E be the greatest weak forward bisimulation equivalence on $\mathscr{A}$, and let $\mathbb{WFB}(\mathscr{A})$ be the class of all automata which are WFB-equivalent to $\mathscr{A}$. Then $\mathscr{A}/E$ is a minimal automaton in $\mathbb{WFB}(\mathscr{A})$. Moreover, if $\mathscr{B}$ is any minimal automaton in $\mathbb{WFB}(\mathscr{A})$, then there exists a weak forward isomorphism between $\mathscr{A}/E$ and $\mathscr{B}$.*

Proof. Let $\mathscr{B}$ be an arbitrary minimal automaton in $\mathbb{WFB}(\mathscr{A})$, and let F be the greatest weak forward bisimulation equivalence on $\mathscr{B}$. According to the previous theorem, there exists a weak forward isomorphism between $\mathscr{A}/E$ and $\mathscr{B}/F$, and by Theorem 4.9 and (4.25) it follows that $\mathscr{B}/F \in \mathbb{WFB}(\mathscr{A})$. Now, by minimality of $\mathscr{B}$ we have that F is the equality relation on B. Thus, we obtain $\mathscr{B}/F \cong B$. Hence, there is a weak forward isomorphism between $\mathscr{A}/E$ and $\mathscr{B}$.

Therefore, $\mathscr{A}/E$ is also a minimal automaton in $\mathbb{WFB}(\mathscr{A})$. □

Chapter 5
Paige-Tarjan type algorithms

The known state minimization algorithms for non-deterministic finite automata, like those in [14, 71, 86, 87, 114], are not significant in practical use. Therefore, NFA state reduction methods are widely studied. The state reduction methods do not necessarily give a minimal NFAs, but they give "reasonably" small one that can be efficiently constructed.

Minimization algorithms for DFAs are based on computing and merging indistinguishable states. As an attempt to adopt this method to a non-deterministic case, Ilie and Yu have introduced the concept of a right invariant equivalence on an NFA in [59, 60]. Right invariant equivalences have been studied in [17, 16, 28, 60, 62, 63]. Furthermore, the same concept was studied under the name "bisimulation equivalence" in many areas of computer science and mathematics, such as modal logic, model checking, set theory, formal verification, etc., and numerous algorithms have been proposed to compute the greatest bisimulation equivalence on a given labeled graph or a labeled transition system (cf. [77, 81, 82, 83, 89, 103]). The faster algorithms are based on the crucial equivalence between the greatest bisimulation equivalence and the relational coarsest partition problem (see [40, 44, 67, 102, 88]).

Ilie and Yu also introduced the dual concept to the concept of right invariant equivalence, called a left invariant equivalence on an NFA, and showed that even smaller NFAs can be obtained alternating reductions by means of right invariant and left invariant equivalences [59, 60, 62, 63]. Another approach in the state reduction was proposed by Champarnaud and Coulon in [15, 16]. They used quasi-orders (preorders) instead of equivalences and showed that the method based on quasi-orders gives better reductions than the method based on equivalences. They gave an algorithm for computing the greatest right invariant and left invariant quasi-orders on an NFA working in a polynomial time, which was later improved in [62, 63].

Since fuzzy finite automata are generalizations of NFAs, in the work with fuzzy automata, the analogue minimization and reduction problems are also presented. Various researches considered the state reduction problem for fuzzy automata and they provided several algorithms which are also based on the idea for computing and merging indistinguishable states [2, 25, 74, 85, 84, 94]. It is worth mentioning, that although these algorithm are called the minimization algorithms, in the general case they do not produce the minimal fuzzy automaton in the set of all fuzzy automata recognizing a given fuzzy language, so these are just reduction algorithms.

Contrary to the deterministic case, where we can effectively detect and merge indistinguishable states, in the non-deterministic case is difficult to decide whether two states are distinguishable or not. In the case of fuzzy automata, this problem is even bigger, due to the fact we work with fuzzy sets of states. However, in the non-deterministic case indistinguishability can be successfully modeled by equivalences and quasi-orders. In [26, 27] it is shown that in the fuzzy case the indistinguishability can be modeled by fuzzy equivalences, and in [117] it is shown that this can be done by fuzzy quasi-orders, too. Namely, in [117], the reduction of fuzzy automata by means of the right and left invariant fuzzy quasi-orders as well as the reduction by means of the right and left invariant fuzzy equivalences is considered. In the same paper, it is shown that right invariant fuzzy quasi-orders on the given automaton $\mathscr{A}$ form a complete lattice whose greatest element gives the best reduction of $\mathscr{A}$ by means of fuzzy quasi-orders of this type. Furthermore, we propose the procedure for computing the greatest right invariant fuzzy quasi-order on a given fuzzy automaton.

This chapter will also be dedicated to the reduction of fuzzy automata by means of right and left invariant fuzzy quasi-orders and right and left invariant fuzzy equivalences, but here we will propose the new algorithm for computing the greatest right invariant fuzzy quasi-order on the given fuzzy automaton, which is based on the famous Paige-Tarjan's coarsest partition problem [88]. Afterwards, the modification of previous algorithm, which computes the greatest right invariant equivalence on the given non-deterministic automaton will be proposed. This modified version performs good complexity. Namely, the complexity of the first algorithm is $O(n^5m)$, where n is the number of the states of the automaton $\mathscr{A}$ and m is the size of the alphabet, whereas the complexity of the modified version is $O(n^3m)$.

5.1. Right and left invariant fuzzy quasi-orders

Let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton over X and $\mathscr{L}$. The fuzzy quasi-order $R \in \mathscr{R}(A)$ is called *right stabile* or just *r-stabile* if it satisfies the following system of fuzzy relational inequalities:

$$R \circ \delta_x \circ R \leqslant \delta_x \circ R, \quad x \in X. \tag{5.1}$$

Similarly, the fuzzy quasi-order $R \in \mathscr{R}(A)$ is called *left stabile*(for short *l-stabile*) if it satisfies the following system of fuzzy relational inequalities:

$$R \circ \delta_x \circ R \leqslant R \circ \delta_x, \quad x \in X. \tag{5.2}$$

Further, the fuzzy quasi-order $R \in \mathscr{R}(A)$ is said to be *right invariant* on $\mathscr{A}$ if it is r-stabile and satisfies the following inequality:

$$R \circ \tau \leqslant \tau. \tag{5.3}$$

Analogously, it is called *left invariant* on $\mathscr{A}$ if it is l-stabile and satisfies the following inequality:

$$\sigma \circ R \leqslant \sigma. \tag{5.4}$$

According to the fact that the fuzzy quasi-order R is a solution to (5.3) if and only if:

$$R \leqslant \tau \backslash \tau = \pi^{fs},$$

we conclude that the r-stabile fuzzy quasi-order on the given fuzzy automaton $\mathscr{A}$ is the right invariant fuzzy quasi-order on $\mathscr{A}$ if and only if it is contained in π^{fs}. And similarly, since fuzzy quasi-order R is a solution to (5.4) if and only if:

$$R \leqslant \sigma \backslash \sigma = \pi^{bs},$$

we obtain that the l-stabile fuzzy quasi-order on the given fuzzy automaton $\mathscr{A}$ is the left invariant quasi-order on $\mathscr{A}$ if and only if it is contained in π^{bs}.

It is worth mentioning that the notion of r-stability and l-stability of fuzzy quasi-order on fuzzy automaton $\mathscr{A} = (A, \delta, \sigma, \tau)$, which is introduced here actually coincides with the notion of right invariant and left invariant fuzzy quasi-order on a given fuzzy transition system $\mathscr{A} = (A, \delta)$. Therefore, the problem of finding the greatest right invariant fuzzy quasi-order on a fuzzy automaton is equivalent to the problem of finding the greatest right invariant fuzzy quasi-order on the corresponding fuzzy transition system contained in a given fuzzy quasi-order R.

Let us note that, since every fuzzy-quasi order is a reflexive relation, it is easy to prove that fuzzy quasi-order R is right invariant on $\mathscr{A}$ if and only if it satisfies the following system of fuzzy relational inequalities:

$$R \circ \delta_x \circ R = \delta_x \circ R, \quad x \in X \tag{5.5}$$

$$R \circ \tau = \tau, \tag{5.6}$$

and that it is left invariant on $\mathscr{A}$ if it satisfies the following:

$$R \circ \delta_x \circ R = R \circ \delta_x, \quad x \in X \tag{5.7}$$

$$\sigma \circ R = \sigma. \tag{5.8}$$

A crisp quasi-order on A which is a solution to (5.5) and (5.6) is called a right invariant quasi-order on $\mathscr{A}$, and a crisp quasi-order which is a solution to (5.7) and (5.8) is called a left invariant quasi-order on $\mathscr{A}$. Let us note that a fuzzy quasi-order on $\mathscr{A}$ is both r-stabile and l-stabile if and only if it is a solution to the system

$$\delta_x \circ R = R \circ \delta_x, \quad x \in X, \tag{5.9}$$

and then it is called an stabile fuzzy quasi-order.

It is clear that all right and left invariant fuzzy quasi-orders on a fuzzy automaton $\mathscr{A}$ are solutions of the general system (2.21), and hence, the corresponding afterset fuzzy transition systems are equivalent to $\mathscr{A}$.

The following theorem presents a version of the important result from [117] which gives the characterization of the set of all right invariant fuzzy quasi-orders

Theorem 5.1. *Let $\mathscr{A} = (A,\sigma,\delta,\tau)$ be a fuzzy automaton.*

(1) Then the set $\mathscr{Q}^{rs}(\mathscr{A})$ of all right stabile fuzzy quasi-orders forms a complete lattice, which is a complete join-subsemilattice of the lattice $\mathscr{Q}(A)$ of all fuzzy quasi-orders on the set A.
(2) Then the set $\mathscr{Q}^{ri}(\mathscr{A})$ of all right invariant fuzzy quasi-orders is a principal ideal of the lattice $\mathscr{Q}^{rs}(\mathscr{A})$.

The following lemma gives a characterization of r-stabile fuzzy quasi-orders:

Lemma 5.1. *Let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton. A fuzzy quasi-order $R \in \mathscr{Q}(A)$ is r-stabile on $\mathscr{A}$ if and only if:*

$$R \leqslant \bigwedge_{x\in X}\bigwedge_{c\in A}(\delta_x \circ R^c)/(\delta_x \circ R^c). \tag{5.10}$$

Proof. Consider the inequality (5.1), i.e.:

$$\bigvee_{b\in A} R(a,b) \otimes (\delta_x \circ R)((b,c) \leqslant (\delta_x \circ R)(a,c), \quad x \in X, \quad a,c \in A. \tag{5.11}$$

This is equivalent to:

$$R(a,b) \otimes (\delta_x \circ R)(b,c) \leqslant (\delta_x \circ R)(a,c), \quad x \in X, \quad a,b,c \in A,$$

and according to the adjunction property this is equivalent to the following:

$$R(a,b) \leqslant \bigwedge_{c\in A}(\delta_x \circ R)(b,c) \to (\delta_x \circ R)(a,c), \quad x \in X, \quad a,b \in A.$$

Consequently, a fuzzy quasi-order $R \in \mathscr{R}(A)$ is r-stable if and only if it satisfies the following inequality

$$R \leqslant \bigwedge_{c\in A} (\delta_x \circ R^c)/(\delta_x \circ R^c), \quad x \in X,$$

which was to be proved. □

As the consequence of the previous lemma we have:

Lemma 5.2. *Let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton over X and $P,R \in \mathscr{Q}(A)$, where $P \leqslant R$. If P is r-stable then :*

$$P \leqslant (\delta_x \circ R^a)/(\delta_x \circ R^a), \quad x \in X, \quad a \in A.$$

Proof. Let $a \in A$. According to the fact that P is r-stable, i.e. $P \circ \delta_x \circ P \leqslant \delta_x \circ P$, for every $x \in X$ we have

$$P \circ \delta_x \circ P \circ R \leqslant \delta_x \circ P \circ R, \quad x \in X.$$

From $P \circ R = R \circ P = R$ there holds

$$P \circ \delta_x \circ R \leqslant \delta_x \circ R, \quad x \in X,$$

or equivalently,

$$\bigvee_{c\in A} P(b,c) \otimes (\delta_x \circ R)(c,a) \leqslant (\delta_x \circ R)(b,a), \quad a,b \in A, \quad x \in X.$$

Therefore, it is clear that the following inequality holds

$$P(b,c) \leqslant (\delta_x \circ R^a)/(\delta_x \circ R^a)(b,c), \quad x \in X,$$

for every $b,c \in A$. □

Let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton, let R be a fuzzy quasi-order on $\mathscr{A}$. With $\mathscr{L}(\delta,\tau,R)$ we will denote the subalgebra of $\mathscr{L}$ generated by all membership values taken by δ and R.

The next theorem provides a method for calculating the greatest right invariant quasi order:

Theorem 5.2. *Let $\mathscr{A} = (A,\delta,\sigma,\tau)$ be a fuzzy automaton over X and $U \in \mathscr{R}(A)$ universal relation on A.*

Define the sequences $\{Q_k\}_{k\in N}$ and $\{R_k\}_{k\in N}$ of fuzzy quasi-orders on A as follows: Initially for $k = 1$

$$\begin{aligned} &R_1 = U, \\ &Q_1 = \pi^{fs} \wedge \Big(\bigwedge_{x\in X} (\delta_x \circ R_1^a)/(\delta_x \circ R_1^a)\Big), \end{aligned} \tag{5.12}$$

where $a \in A$ is an arbitrary element.

Further, for each $k \geqslant 2$ repeat the following step: Find $a \in A$, such that $R_k^a \neq Q_k^a$ and set

$$R_{k+1} = R_k \wedge (Q_k^a / Q_k^a), \tag{5.13}$$

$$Q_{k+1} = Q_k \wedge \Big(\bigwedge_{x \in X} \Big(\bigwedge_{b \in S_k^a} (\delta_x \circ R_{k+1}^b)/(\delta_x \circ R_{k+1}^b) \Big) \Big), \tag{5.14}$$

($S_k^a = \{b \in A \,|\, Q_k^a(b) \neq 0\}$), until $R_k = Q_k$. Then:

(a) Sequences $\{Q_k\}_{k \in N}$ and $\{R_k\}_{k \in N}$ are descending;
(b) $Q_k \leqslant R_k$ for every $k \in N$;
(c) For every $k \in N$, for all $c \in A$ the following holds:

$$Q_k \leqslant \bigwedge_{x \in X} (\delta_x \circ R_k^c)/(\delta_x \circ R_k^c); \tag{5.15}$$

(d) Q_k and R_k are fuzzy quasi-orders for every $k \in N$;
(e) If $R_k = Q_k$, for some $k \in N$, then Q_k is the greatest right invariant fuzzy quasi-order on $\mathscr{A}$;
(f) If $\mathscr{A}$ is finite and $\mathscr{L}(\delta, \tau, R)$ satisfies DCC, then the sequences $\{Q_k\}_{k \in N}$ and $\{R_k\}_{k \in N}$ are finite, that is, there exists $k \in N$ such that $R_k = Q_k$;

Proof. (a) Follows directly from the definition of Q_k and R_k;

(b) We prove (b) by induction on $k \in N$.

For $k = 1$, evidently holds $Q_1 \leqslant R_1$.

Suppose for $k = m$, $Q_m \leqslant R_m$, and prove $Q_{m+1} \leqslant R_{m+1}$.

According to Lemma 1.4 we have $Q_m \leqslant Q_m^a / Q_m^a$, by induction assumption $Q_m \leqslant R_m$, therefore $Q_m \leqslant R_m \wedge (Q_m^a / Q_m^a) = R_{m+1}$, and since $\{Q_k\}_{k \in N}$ is descending, we have that $Q_{m+1} \leqslant R_{m+1}$, and that was to be proved.

(c) We prove (c) also by induction on $k \in N$.

Primarily, for $k = 1$ all the foresets R_1^c of R_1 are equal to each other, because for any $c \in A$, R_1^c is defined by $R_1^c(b) = 1$, for every $b \in A$. By the definition of Q_1 we have :

$$Q_1 \leqslant \bigwedge_{x \in X} (\delta_x \circ R_1^c)/(\delta_x \circ R_1^c),$$

for every $c \in A$, and hence for $k = 1$ inequality (5.26) holds.

Suppose that for $k = m$, inequality (5.26) holds and let us prove it for $k = m + 1$.

At first, let us note that directly from the definition of Q_{m+1} we have:

$$Q_{m+1} \leqslant \bigwedge_{x \in X} \Big((\delta_x \circ R_{m+1}^c)/(\delta_x \circ R_{m+1}^c) \Big), \quad c \in S_m^a(c). \tag{5.16}$$

Next, consider $c \in A$ where $Q_m^a(c) = 0$, i.e. $c \in (A - S_m^a)$. Now we will show that $R_{m+1}^c = R_m^c$. For all $d \in A$ we have

$$\begin{aligned}R^c_{m+1}(d) = R_{m+1}(d,c) &= R_m(d,c) \wedge (Q^a_m/Q^a_m)(d,c) = \\ &= R^c_m(d) \wedge (Q^a_m(c) \to Q^a_m(d)) = R^c_m(d) \wedge (0 \to Q^a_m(d)) = \\ &= R^c_m(d) \wedge 1 = R^c_m(d).\end{aligned}$$

Now, according to induction assumption we have that

$$Q_m \leqslant \bigwedge_{x\in X} (\delta_x \circ R^c_m)/(\delta_x \circ R^c_m), \ \text{ for every } c \in A,$$

and we conclude the inequality holds also for all $c \in (A - S^a_m)$. Since $Q_{m+1} \leqslant Q_m$ we obtain:

$$Q_{m+1} \leqslant \bigwedge_{x\in X} \Big((\delta_x \circ R^c_{m+1})/(\delta_x \circ R^c_{m+1}) \Big), \quad c \in (A - S^a_m). \tag{5.17}$$

From (5.16) and (5.17) we obtain that inequality (5.26) holds for $k = m+1$, which was to be proved.

(d) By Lemma 1.1 and the fact that for every fuzzy subset f on A relation defined by equation (1.47) is fuzzy quasi order it follows that Q_k and R_k are fuzzy quasi orders for every $k \in N$.

(e) If $R_k = Q_k$, for some $k \in N$, then according to (c) the following holds:

$$Q_k \leqslant \bigwedge_{x\in X} (\delta_x \circ Q^c_k)/(\delta_x \circ Q^c_k), \quad \text{for all } c \in A,$$

which means that Q_k is r-stable. From that and the fact that $Q_k \leqslant \pi^{fs}$, we conclude Q_k is right invariant fuzzy quasi-order on $\mathscr{A}$. In order to show that Q_k is the greatest one, let us consider an arbitrary right invariant fuzzy quasi order P on $\mathscr{A}$.

We will prove that $P \leqslant Q_n$ for every $n \in N$, by induction on n.

For $n = 1$, since P, U are fuzzy quasi-orders such that $P \leqslant U = R_1$, according to Lemma 5.2 we have that $P \leqslant (\delta \circ R^a_1)/(\delta \circ R^a_1)$. Because of the fact P is right invariant fuzzy quasi-order on $\mathscr{A}$, P have to satisfy the inequality (5.3), which means $P \leqslant \pi^{fs}$. Thus $P \leqslant Q_1$ holds.

Suppose that assumption $P \leqslant Q_m$ holds for $n = m$, and prove $P \leqslant Q_{m+1}$.

Since $P \leqslant Q_m$, using (b), we obtain $P \leqslant R_m$ and by Lemma 1.4 it follows $P \leqslant R_{m+1}$. Then again according to Lemma 5.2 we obtain $P \leqslant Q_{m+1}$, which was to be proved.

(f) Let $\mathscr{A}$ be a finite fuzzy transition system and let $\mathscr{L}(\delta,\tau,R)$ satisfy DCC. Then fuzzy relations $\{R_k\}_{k\in\mathbb{N}}$ can be considered as fuzzy matrices with entries in $\mathscr{L}(\delta,\tau,R)$, and for any pair $(a,b) \in A \times A$, the (a,b)-entries of these matrices form a decreasing sequence $\{R_k(a,b)\}_{k\in\mathbb{N}}$ of elements of $\mathscr{L}(\delta,\tau,R)$. By the hypothesis, all these sequences stabilize, and since there is a finite number of these sequences, there exists $s \in \mathbb{N}$ such that after s steps all these sequences are stabilized. This means that the sequence $\{R_k\}_{k\in\mathbb{N}}$ of fuzzy quasi-orders also stabilizes after s steps, i.e., $R_k = R_{k+1}$.

Next, we will prove that if $R_k = R_{k+1}$ then $R_k = Q_k$. If $R_k = R_{k+1}$ then

$$R_k = R_{k+1} = R_k \wedge (Q_k^a / Q_k^a),$$

and thus, $R_k \leqslant Q_k^a / Q_k^a$. Consequently, $R_k^a \leqslant Q_k^a$, and since $Q_k^a \leqslant R_k^a$ we obtain $R_k^a = Q_k^a$. This means that there is no foreset R_k^a of R_k such that $R_k^a \neq Q_k^a$, or equivalently $R_k = Q_k$. $\square$

The previous theorem can be transformed into the following algorithm.

Algorithm 5.3 *The input is a fuzzy automaton $\mathscr{A} = (A, X, \delta^A, \sigma^A, \tau^A)$. The algorithm computes the greatest right invariant fuzzy quasi-order R^{ri} on $\mathscr{A}$.*

The procedure constructs the sequences of fuzzy quasi-orders $\{R_k\}_{k\in\mathbb{N}}$ and $\{Q_k\}_{k\in\mathbb{N}}$, in the following way:

(A1) In the first step we set

$$R_1 = U$$
$$Q_1 = \pi^{fs} \wedge \Big(\bigwedge_{x\in X} (\delta_x \circ R_1^a)/(\delta_x \circ R_1^a) \Big).$$

(A2) After the kth step let R_k and Q_k be fuzzy quasi-orders that have been constructed.

(A3) In the next step we find first $a \in A$ such that $R_k^a \neq Q_k^a$ if such exists.

(A4) If a has been found in the previous step, construct the fuzzy quasi-order R_{k+1} by means of the formula (B.55) and the fuzzy quasi-order Q_{k+1} by (B.56), otherwise the procedure of constructing the sequence $\{R_k\}_{k\in\mathbb{N}}$ and $\{Q_k\}_{k\in\mathbb{N}}$ terminates and Q_{k+1} is the greatest right invariant fuzzy quasi-order on $\mathscr{A}$.

According to Theorem 5.2, if the subalgebra $\mathscr{L}(\delta, \tau, R)$ of $\mathscr{L}$, generated by all values taken by fuzzy transition relations of $\mathscr{A}$ and R, satisfies DCC, the algorithm terminates in a finite number of steps.

Let us analize the computational time of this algorithm. Let n denote the number of states of $\mathscr{A}$ and m the number of letters in the input alphabet X, and let $c_\vee$, $c_\wedge$, $c_\otimes$ and $c_\to$ be the computational costs of the operations $\vee$, $\wedge$, $\otimes$ and $\to$ in $\mathscr{L}$, respectively. In particular, if $\mathscr{L}$ is linearly ordered, then the operations $\vee$ and $\wedge$ can be computed in constant time, so $c_\vee$ and $c_\wedge$ can be omitted, and similarly, when $\mathscr{L}$ is the Gödel structure, we can also omit $c_\otimes$ and $c_\to$.

In the step (A1) in order to compute Q_1 we first compute π^{fs}, which can be done in time $O(n^2 c_\to)$. Next, we compute the composition of fuzzy relations δ_x^A and R^a, for first $a \in A$ and if this computation is performed according to the definition of composition od fuzzy relations, its computational time is $O(n^2(c_\otimes + c_\vee))$. Then we compute $\bigwedge_{x\in X}(\delta_x \circ R_1^a)/(\delta_x \circ R_1^a)$, and the computational time of this part is $O(mn^2(c_\to + c_\wedge))$. Thus, the total computational time of (A1) is $O(mn^2(c_\to + c_\wedge + c_\otimes + c_\vee))$.

In (A3) computational time to find $a \in A$ such that $R_k^a \neq Q_k^a$, if it exists, is $O(n^2)$.

In (A4) computing of R_{k+1} using formula (B.55) can be done in time $O(n^2(c_{\to} + c_{\wedge}))$ and the computing of Q_{k+1} using formula (B.56) can be done in $O(mn^3(c_{\to} + c_{\wedge} + c_{\otimes} + c_{\vee}))$.

The hardest problem is to estimate the number of steps, in the case when it is finite.We need only to consider R_k because the both sequences have the same number of elements if they are finite. Consider fuzzy relations R_k as fuzzy matrices. After each step in the construction of the sequence $\{R_k\}_{k\in\mathbb{N}}$ we check whether some entry has changed its value, and the algorithm terminates after the first step in which there was no change. Suppose that $\mathscr{L}(\delta,\tau,R)$ satisfies DCC. Then $\{\{R_k(a,b)\}_{k\in\mathbb{N}} \mid (a,b) \in A^2\}$ is a finite collection of the finite sequences, so there exists $l \in \mathbb{N}$ such that the number of different elements in each of these sequences is less than or equal to l. As the sequence $\{R_k\}_{k\in\mathbb{N}}$ is descending, each entry can change its value at most $l-1$ times, and the total number of changes is less than or equal to $(l-1)(n^2-n)$ (the diagonal values must always be 1). Therefore, the algorithm terminates after at most $(l-1)(n^2-n)+2$ steps (changes are happening between the second and second to last step).

Summing up, we get that the total computation time for the whole algorithm is $O(lmn^5(c_{\to} + c_{\wedge} + c_{\otimes} + c_{\vee}))$, and hence, the algorithm is polynomial-time.

Let us note that if the subalgebra $\mathscr{L}(\delta,\tau,R)$ is finite, then we can assume that l is the number of elements of this subalgebra. In particular, if $\mathscr{L}$ is the Gödel structure, then the only values that can be taken by fuzzy relations $\{R_k\}_{k\in\mathbb{N}}$ are 1 and those taken by fuzzy relations $\{\delta_x\}_{x\in X}$ and R, and we can assume that $l = j+1$, where j is the number of different values taken by fuzzy relations $\{\delta_x\}_{x\in X}$ and R. Therefore, in this case the algorithm terminates after at most $j(n^2-n)+2$ steps, or after at most $(m+1)n^2(n^2-n)+2$ steps, since $j \leqslant mn^2+n^2$, and the computation time of our algorithm is $O(jmn^5)$, or $O(m^2n^7)$.

Finally, if $\mathscr{L}$ is the Boolean structure, then $l = 2$, and the algorithm terminates after at most n^2-n+2 steps, which means that the computational time of the algorithm is $O(mn^5)$.

According to (c) of Theorem 5.2, if the structure $\mathscr{L}$ is locally finite, then for every fuzzy transition system $\mathscr{A}$ over $\mathscr{L}$ we have that sequences of fuzzy quasi-orders defined by (B.55) and (B.56) are finite. However, this does not necessary hold if $\mathscr{L}$ is not locally finite, as the following example shows:

Example 5.1. Let $\mathscr{L}$ be the Goguen (product) structure and $\mathscr{A} = (A,\delta,\sigma,\tau)$ a fuzzy automaton over $\mathscr{L}$, where $A = \{1,2\}$, $X = \{x\}$, and δ_x^A is given by

$$\sigma = \begin{bmatrix} 1 & 1 \end{bmatrix}, \delta_x = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.1 \end{bmatrix}, \tau = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

and let R be the universal relation on A. Applying to R the procedure from Theorem 5.2, we obtain a sequences $\{R_k\}_{k\in\mathbb{N}}$ and $\{Q_k\}_{k\in\mathbb{N}}$ of fuzzy quasi-orders given by

$$R_k = \begin{bmatrix} 1 & 1 \\ \frac{1}{2^{k-1}} & 1 \end{bmatrix}, \; Q_k = \begin{bmatrix} 1 & 1 \\ \frac{1}{2^k} & 1 \end{bmatrix}, \; k \in \mathbb{N},$$

and thus $R_k \neq Q_k$, i.e., this sequences is infinite.

In the case fuzzy set of transition states is

$$\tau = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

using the procedure from Theorem 5.2 we obtain

$$R_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \; Q_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \; R_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \; Q_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

And hence the greatest right invariant equivalence on $\mathscr{A}$ is relation R_2.

For a fuzzy automaton $\mathscr{A} = (A, \delta, \sigma, \tau)$ over X and $\mathscr{L}$, the greatest left invariant fuzzy quasi-order R^{li} on A can be computed in a similar way as R^{ri}.

Namely, we define a sequences $\{R^k\}_{k\in\mathbb{N}}$ and $\{Q^k\}_{k\in\mathbb{N}}$ of fuzzy quasi-orders on A as follows:

Initially, $k = 1$:

$$\begin{aligned} R^1 &= U, \\ Q^1 &= \pi^{bs} \wedge \Big(\bigwedge_{x\in X} (R_a^1 \circ \delta_x) \backslash (R_a^1 \circ \delta_x) \Big), \end{aligned}$$

where $a \in A$ is arbitrary element.

Further, for each $k \geqslant 2$ repeat the following step: Find $a \in A$, such that $R_k^a \neq Q_k^a$ and set

$$R^{k+1} = R^k \wedge (Q_a^k \backslash Q_a^k), \tag{5.18}$$

$$Q^{k+1} = Q^k \wedge \Big(\bigwedge_{x\in X} \Big(\bigwedge_{b\in S_a^k} (R_b^{k+1} \circ \delta_x) \backslash (R_b^{k+1} \circ \delta_x) \Big) \Big), \tag{5.19}$$

until $Q^k = R^k$.

If $\mathscr{L}$ is locally finite, then this sequence are necessary finite and R^{li} equals the least element.

It is important to maintain that the greatest right and left invariant fuzzy quasi-orders are calculated using iterative procedures, but these calculations are not approximative. If these procedures terminate in a finite number of steps, exact solutions to the considered systems of fuzzy relation equations are obtained.

For a fuzzy automaton $\mathscr{A} = (A, \delta, \sigma, \tau)$ over X and $\mathscr{L}$ we give also a procedure for computing the greatest right invariant fuzzy equivalence on $\mathscr{A}$. This procedure is similar to the procedure given in Theorem 5.2 for fuzzy quasi-orders, and it also works for all fuzzy automata over a locally finite complete residuated lattice.

First of all, note that fuzzy equivalence E is a solution to inequality (5.3) if and only if $E \leqslant \tau | \tau = \pi^{fb}$.

The following lemma will be needed for proving the main result:

Lemma 5.3. *Let $E, F \in \mathscr{E}(A)$ and $E \leqslant F$. If E is r-stable then the following holds:*

$$E \leqslant (\delta_x \circ F^a) | (\delta_x \circ F^a), \quad x \in X,\ a \in A. \tag{5.20}$$

Proof. In the analogue way as in the proof of the Lemma 5.2, we show that:

$$E \leqslant (\delta_x \circ F^a) / (\delta_x \circ F^a), \quad x \in X,\ a \in A.$$

Using the fact that E is symmetric for every $b, c \in A$:

$$\begin{aligned} E(b,c) = E(c,b) &\leqslant \Big((\delta_x \circ F^a)/(\delta_x \circ F^a)\Big)(c,b) = \\ &= (\delta_x \circ F^a)(b) \to (\delta_x \circ F^a)(c) = \Big((\delta_x \circ F^a)\backslash(\delta_x \circ F^a)\Big)(b,c). \end{aligned}$$

Therefore, E satisfies (5.20). □

Next theorem gives a method to compute the greatest right invariant fuzzy equivalence on the given automaton $\mathscr{A}$.

Theorem 5.4. *Let $\mathscr{A} = (A, \delta, \sigma, \tau)$ be a fuzzy automaton over X, and $U \in \mathscr{R}(A)$ universal relation on A.*

Define the sequences $\{E_k\}_{k \in N}$ and $\{X_k\}_{k \in N}$ of equivalences on A as follows: Initially for $k = 1$

$$\begin{aligned} &X_1 = U, \\ &E_1 = \pi^{fb} \wedge \Big((\delta \circ X_1^a) | (\delta \circ X_1^a)\Big), \quad \textit{for some } a \in A. \end{aligned} \tag{5.21}$$

Further, for each $k \in N$ repeat the following step: Find $a \in A$, such that $X_k^a \neq E_k^a$ and set

$$\begin{aligned} &X_{k+1} = X_k \wedge (E_k^a | E_k^a), \\ &E_{k+1} = E_k \wedge \Big((\delta \circ E_k^a) | (\delta \circ E_k^a)\Big) \wedge \Big(\bigwedge_{b \in T_k^a} (\delta \circ X_{k+1}^b) | (\delta \circ X_{k+1}^b)\Big), \end{aligned} \tag{5.22}$$

($T_k^a = \{b \in A \,|\, E_k^a(b) \neq 1\}$) until $X_k = E_k$. Then:

(a) Sequences $\{E_k\}_{k \in N}$ and $\{X_k\}_{k \in N}$ are descending;
(b) For every $k \in N$, $E_k \leqslant X_k$;

(c) For every $k \in N$, for all $c \in A$ the following holds :

$$E_k \leqslant \bigwedge_{x\in X}\big((\delta_x \circ X_k^c)|(\delta_x \circ X_k^c)\big); \tag{5.23}$$

(d) For every $k \in N$, X_k and E_k are fuzzy equivalence relations;
(e) If $X_k = E_k$, for some $k \in N$, then E_k is the greatest right invariant fuzzy equivalence on $\mathscr{A}$;
(f) If $\mathscr{A}$ is finite and $\mathscr{L}(\delta,\tau,R)$ satisfies DCC, then the sequences $\{E_k\}_{k\in N}$ and $\{X_k\}_{k\in N}$ are finite, that is, there exists $k \in N$ such that $X_k = E_k$;

Proof. (a) Follows directly from the definition of E_k and X_k;

(b), (d) and (e) can be proved in the similar way as Theorem 5.2 (b), (d) and (e), respectively.

(c) We will prove this by induction on the number of states.

For $k = 1$ it can be proved in the similar way as Theorem 5.2(c).

Further, let $k > 1$. Let us show, first, that if $E_{k-1}^a(b) = 1$ then $X_k^b = E_{k-1}^a$. Using the fact that if $E_{k-1}^a(b) = 1$, then $E_{k-1}^a = E_{k-1}^b$, and for all $c \in A$ we have

$$\begin{aligned} X_k^b(c) &= X_k(c,b) = X_{k-1}(c,b) \wedge (E_{k-1}^a|E_{k-1}^a)(c,b) = \\ &= X_{k-1}^b(c) \wedge (E_{k-1}^a(b) \leftrightarrow E_{k-1}^a(c)) = X_{k-1}^b(c) \wedge (1 \leftrightarrow E_{k-1}^a(c)) \\ &= X_{k-1}^b(c) \wedge E_{k-1}^a(c) = E_{k-1}^a(c). \end{aligned}$$

From the definition of relation E_k, it follows that E_k is stable w.r.t. X_k^b, such that $E_{k-1}^a(b) \neq 1$ and that it is also stable w.r.t. E_{k-1}^a, that is w.r.t. all X_k^b, such that $E_{k-1}^a(b) = 1$. Therefore for $k \geqslant 1$ inequality (5.23) holds. □

The following example shows the case where the sequences of equivalences obtained using (5.21) and (B.59) are infinite, whereas the sequences of fuzzy quasi orders obtained using (5.12) and (B.56) are finite:

Example 5.2. Let $\mathscr{L}$ be the Goguen (product) structure and $\mathscr{A} = (A,\delta^A)$ a fuzzy transition system over $\mathscr{L}$, where $A = \{1,2,3\}$, $X = \{x\}$, and δ_x^A is given by

$$\delta_x^A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}.$$

If we start from the universal relation on A, according to (5.21) and (B.59) we obtain infinite sequence $\{X_k\}_{k\in\mathbb{N}}$ and $\{E_k\}_{k\in\mathbb{N}}$ of fuzzy equivalences on A, where

$$X_k = \begin{bmatrix} 1 & 1 & \frac{1}{2^{k-1}} \\ 1 & 1 & \frac{1}{2^{k-1}} \\ \frac{1}{2^{k-1}} & \frac{1}{2^{k-1}} & 1 \end{bmatrix}, \quad E_k = \begin{bmatrix} 1 & 1 & \frac{1}{2^k} \\ 1 & 1 & \frac{1}{2^k} \\ \frac{1}{2^k} & \frac{1}{2^k} & 1 \end{bmatrix}, \quad k \in \mathbb{N}.$$

On the other hand, if we also start from the universal relation, according to (5.12) and (B.56) gives a finite sequence $\{R_k\}_{k\in\mathbb{N}}$ of fuzzy quasi-orders on A, where

$$R_1 = \begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}, \quad Q_1 = \begin{bmatrix}1&1&1\\1&1&1\\\frac{1}{2}&\frac{1}{2}&1\end{bmatrix}, \quad R_2 = Q_2 = \begin{bmatrix}1&1&1\\1&1&1\\\frac{1}{2}&\frac{1}{2}&1\end{bmatrix}.$$

The next example shows the case where R is a right invariant fuzzy quasi-order and a E_R is not a right invariant fuzzy equivalence.

Example 5.3. Let $\mathscr{L}$ be the Boolean structure, and let $\mathscr{A} = (A, \delta^A)$ be a fuzzy transition system over $\mathscr{L}$, where $A = \{1,2,3\}$, $X = \{x,y\}$, and fuzzy transition relations δ_x^A and δ_y^A are given by

$$\delta_x^A = \begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix}, \quad \delta_y^A = \begin{bmatrix}1&0&0\\1&1&0\\1&0&0\end{bmatrix}.$$

The greatest right invariant fuzzy quasi order R^{ri} on $\mathscr{A}$, its natural fuzzy equivalence $E_{R^{\mathrm{ri}}}$, and the greatest right invariant fuzzy equivalence E^{ri} on $\mathscr{A}$ are given by

$$R^{\mathrm{ri}} = \begin{bmatrix}1&1&1\\0&1&1\\0&1&1\end{bmatrix}, \quad E_{R^{\mathrm{ri}}} = \begin{bmatrix}1&0&0\\0&1&1\\0&1&1\end{bmatrix}, \quad E^{\mathrm{ri}} = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}.$$

Thus, E^{ri} do not reduce the number of states of $\mathscr{A}$, but R^{ri} reduces $\mathscr{A}$ to a fuzzy transition system with two states.

Moreover, R^{ri} is a right invariant fuzzy quasi-order, but its natural fuzzy equivalence $E_{R^{\mathrm{ri}}}$ is not a right invariant fuzzy equivalence, because $E^{\mathrm{ri}} < E_{R^{\mathrm{ri}}}$. We also have that the afterset fuzzy transition system $\mathscr{A}/R^{\mathrm{ri}}$ is not isomorphic to the factor fuzzy transition system $\mathscr{A}/E_{R^{\mathrm{ri}}}$, since

$$R^{\mathrm{ri}} \circ \delta_y^A \circ R^{\mathrm{ri}} = \begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}, \quad E_{R^{\mathrm{ri}}} \circ \delta_y^A \circ E_{R^{\mathrm{ri}}} = \begin{bmatrix}1&0&0\\1&1&1\\1&1&1\end{bmatrix}.$$

5.2. Computing the right invariant equivalences on non-deterministic automata

In this section, we present the modification of the algorithm for computing the greatest right invariant fuzzy equivalence. This method computes the greatest right invariant equivalences for fuzzy automata over Boolean structure, i.e. classical non-deterministic automata.

Theorem 5.5. *Let $\mathscr{A} = (A, \delta, \sigma, \tau)$ be a nondeterministic automaton and $U = A \times A$ universal relation on A.*

Define the sequences $\{E_k\}_{k\in N}$ and $\{R_k\}_{k\in N}$ of equivalences on A as follows: Initially for $k = 1$

$$R_1 = U, \qquad E_1 = \pi^{fb} \cap \Big(\bigcap_{x\in X} (\delta_x \circ U^a) | (\delta_x \circ U^a)\Big),$$

where a is arbitrary element in A.

Further, for each $k \in N$ repeat the following step: Find $a \in A$, such that $R_k^a \neq E_k^a$ and set

$$R_{k+1} = R_k \cap (E_k^a | E_k^a), \tag{5.24}$$

$$E_{k+1} = E_k \cap \Big(\bigcap_{x\in X} \Big((\delta_x \circ (R_k^a - E_k^a)) | (\delta_x \circ (R_k^a - E_k^a))\Big) \cap ((\delta_x \circ E_k^a) | (\delta_x \circ E_k^a))\Big), \tag{5.25}$$

until $R_k = E_k$. Then:

(a) Sequences $\{E_k\}_{k\in N}$ and $\{R_k\}_{k\in N}$ are descending;
(b) For every $k \in N$, $E_k \subseteq R_k$;
(c) For every $k \in N$, for all $c \in A$ the following holds :

$$E_k \subseteq \bigcap_{x\in X} (\delta_x \circ R_k^c) | (\delta_x \circ R_k^c); \tag{5.26}$$

(d) For every $k \in N$, R_k and E_k are equivalence relations;
(e) The procedure terminates at most $|A| - 1$ steps and if it terminates after n steps then E_n is the greatest right invariant equivalence on $\mathscr{A}$.

Proof. (a), (b) and (d) can be proved in the analogue way as the proof of the Theorem 5.4.

(c) We also prove (c) by induction on $k \in N$.

In the case $k = 1$ directly from the definition of E_1, we obtain (5.26) holds.

For $k = 2$, $R_2 = R_1 \cap (E_1^a | E_1^a)$ for some $a \in A$. Since $R_1 = U$ it has only one equivalence class and we will call it R_1^a. According to the definition of R_2, we conclude that R_2 is also an equivalence relation and it has exactly two equivalence classes E_1^a and $R_1^a - E_1^a$. Now, according to the definition of E_2 we have:

$$E_2 = E_1 \cap \Big(\bigcap_{x\in X} \Big((\delta_x \circ (R_1^a - E_1^a)) | (\delta_x \circ (R_1^a - E_1^a))\Big) \cap ((\delta_x \circ E_1^a) | (\delta_x \circ E_1^a))\Big)$$

which means,

$$E_2 \subseteq (\delta_x \circ E_1^a) | (\delta_x \circ E_1^a), \quad x \in X \tag{5.27}$$

and

$$E_2 \subseteq (\delta_x \circ (R_1^a - E_1^a)) | (\delta_x \circ (R_1^a - E_1^a)), \quad x \in X. \tag{5.28}$$

From (5.27) and (5.28) we have:

$$E_2 = E_2 \cap \Big(\bigcap_{x\in X}\big((\delta_x \circ (R_1^a - E_1^a))|(\delta_x \circ (R_1^a - E_1^a))\big) \cap ((\delta_x \circ E_1^a)|(\delta_x \circ E_1^a))\Big),$$

and hence for $k = 2$ inequality (5.26) holds.

Suppose that for $k = m$ inequality (5.26) is true, that is, for every $b \in A$,

$$E_m = E_m \cap \Big(\bigcap_{x\in X}((\delta_x \circ R_m^b)|(\delta_x \circ R_m^b))\Big).$$

Consider equivalence relation $R_{m+1} = R_m \cap (E_m^a|E_m^a)$, for some $a \in A$ such that $R_m^a \neq E_m^a$. According to (b) $E_m^a \subset R_m^a$. This means that except of the equivalence class R_m^a, equivalence relations R_m and R_{m+1} have all equivalence classes the same. Moreover, R_m has the equivalence class R_m^a, whereas R_{m+1} has two equivalence classes E_m^a and $R_m^a - E_m^a$. Therefore, for any equivalence class R_{m+1}^b, $b \in A$, such that $R_{m+1}^b \neq E_m^a$ and $R_{m+1}^b \neq (R_m^a - E_m^a)$, according to the induction assumption we have:

$$E_{m+1} \subseteq E_m = E_m \cap \Big(\bigcap_{x\in X}((\delta_x \circ R_m^b)|(\delta_x \circ R_m^b))\Big) =$$
$$= E_m \cap \Big(\bigcap_{x\in X}((\delta_x \circ R_{m+1}^b)|(\delta_x \circ R_{m+1}^b))\Big) \subseteq \bigcap_{x\in X}((\delta_x \circ R_{m+1}^b)|(\delta_x \circ R_{m+1}^b)),$$

and hence,

$$E_{m+1} = E_{m+1} \cap \Big(\bigcap_{x\in X}((\delta_x \circ R_{m+1}^b)|(\delta_x \circ R_{m+1}^b))\Big).$$

Therefore, for every $b \in A$, such that $R_{m+1}^b \neq E_m^a$ and $R_{m+1}^b \neq (R_m^a - E_m^a)$, (5.26) holds. For equivalence classes E_m^a, according to the definition of E_{m+1} the following holds:

$$E_{m+1} \subseteq E_m \cap \Big(\bigcap_{x\in X}((\delta_x \circ E_m^a)|(\delta_x \circ E_m^a))\Big) \subseteq \bigcap_{x\in X}((\delta_x \circ E_m^a)|(\delta_x \circ E_m^a)),$$

which means,

$$E_{m+1} = E_{m+1} \cap \Big(\bigcap_{x\in X}((\delta_x \circ E_m^a)|(\delta_x \circ E_m^a))\Big).$$

In analogue way we prove that:

$$E_{m+1} = E_{m+1} \cap \Big(\bigcap_{x\in X}\big((\delta_x \circ (R_m^a - E_m^a))|(\delta_x \circ (R_m^a - E_m^a))\big)\Big).$$

Hence, for $k = m+1$ inequality (5.26) is satisfied, which was to be proved.

(e) According to (a), the sequence $\{E_k\}_{k\in N}$ is deceasing. So, since the first equivalence E_1 in the worst case can have only one class, and the last equivalence can have at most $|A|$ equivalence classes (identical relation on A), we obtain that this algorithm terminates after at most $|A|-1$ steps. The other part of the assertion can be proved in analogue way as Theorem 5.4(e). □

According to the previous theorem we have the following procedure for computing the greatest right invariant equivalence on a given non- deterministic automaton.

Algorithm 5.6 *The input of this algorithm is a nondeterministic automaton $\mathscr{A} = (A, X, \delta^A, \sigma^A, \tau^A)$. The algorithm computes the greatest right invariant equivalence E^{ri} on $\mathscr{A}$.*

The procedure constructs the sequence of equivalences $\{R_k\}_{k\in\mathbb{N}}$ and $\{E_k\}_{k\in\mathbb{N}}$, in the following way:

(A1) In the first step we set

$$R_1 = U$$
$$E_1 = \pi^{fb} \cap \Big(\bigcap_{x\in X}((\delta_x \circ R_1^a)|(\delta_x \circ R_1^a))\Big).$$

(A2) After the kth step let R_k and E_k be equivalences that have been constructed.
(A3) In the next step we find first $a \in A$ such that $R_k^a \neq E_k^a$ if it exists.
(A4) If in the previous step a was found then construct the equivalence R_{k+1} by means of the formula (5.24) and the equivalence E_{k+1} by means of the formula (B.62), otherwise the procedure of constructing the sequence $\{R_k\}_{k\in\mathbb{N}}$ and $\{E_k\}_{k\in\mathbb{N}}$ terminates and E_{k+1} is the greatest right invariant equivalence on $\mathscr{A}$.

Observe now the computational time of this algorithm. Let n denote the number of states of $\mathscr{A}$ and m the number of letters in the input alphabet X, and let $c_\cup$, $c_\cap$, $c_\otimes$ and $c_\to$ be respectively computational costs of the operations $\cup$, $\cap$, $\otimes$ and $\to$ in $\mathscr{L}$. Since $\mathscr{L}$ is Boolean structure we can assume that $c_\cup = c_\cap = c_\otimes = c_\to = 1$ and they can be omitted.

In the step (A1) in order to compute E_1 we firs compute π^{fb}, which can be done in time $O(n^2 c_\to)$. Next, we compute the composition of fuzzy relations δ_x^A and R^a, for first $a \in A$ and if this computation is performed according to the definition of composition od fuzzy relations, its computational time is $O(n^2(c_\otimes + c_\cup))$. Then we compute $\bigcap_{x\in X}(\delta_x \circ R_1^a)|(\delta_x \circ R_1^a)$, and the computational time of this part is $O(mn^2(c_\to + c_\cap))$. Thus, the total computational time of (A1) is $O(mn^2(c_\to + c_\cap + c_\otimes + c_\cup))$.

In (A3) computational time to find $a \in A$ $R_k^a \neq Q_k^a$, if it exists, is $O(n^2)$.

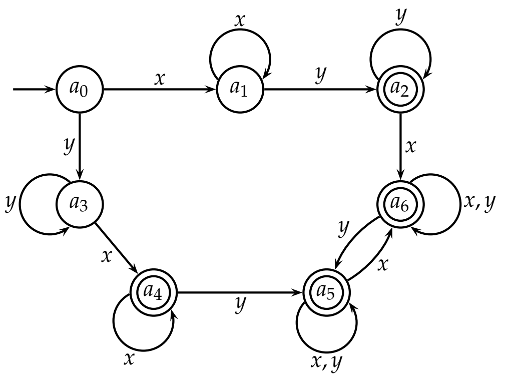
In (A4) computing of R_{k+1} using formula (5.24) can be done in time $O(n^2(c_\to + c_\cap))$ and the computing of E_{k+1} using formula (B.62) can be done in $O(mn^2(c_\to + c_\cap + c_\otimes + c_\cup))$.

As we have shown in the Theorem 5.5(e), the number of steps of the previous algorithm is at most $n-1$ step, i.e. the computational time is $O(n)$.

Summing up, we get that the total computation time for the whole algorithm is $O(mn^3(c_\to + c_\cap + c_\otimes + c_\cup))$, and since we have the Boolean structure we obtain that computational time of the whole algorithm is $O(mn^3)$, the algorithm is polynomial-time.

The following example shows the work of the algorithm.

Example 5.4. Let $\mathscr{A} = (A, \sigma, \delta, \tau)$ be a non-deterministic automaton given by the following figure:



Using the Algorithm 5.6(A1) we obtain:

$$R_1 = U = \begin{bmatrix} 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1 \end{bmatrix}, \quad E_1 = \pi^{fb} \cap \Big(\bigcap_{x\in X} (\delta_x \circ R_1^a) | (\delta_x \circ R_1^a)\Big) = \begin{bmatrix} 1&1&0&1&0&0&0\\ 1&1&0&1&0&0&0\\ 0&0&1&0&1&1&1\\ 1&1&0&1&0&0&0\\ 0&0&1&0&1&1&1\\ 0&0&1&0&1&1&1\\ 0&0&1&0&1&1&1 \end{bmatrix}.$$

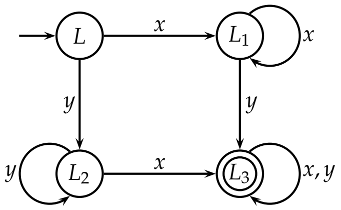
Now since in (A2), $R_1^1 \neq E_1^1$ we choose the first class E_1^1 and compute:

$$R_2 = R_1 \cap (E_1^1 \leftrightarrow E_1^1) \begin{bmatrix} 1&1&0&1&0&0&0\\ 1&1&0&1&0&0&0\\ 0&0&1&0&1&1&1\\ 1&1&0&1&0&0&0\\ 0&0&1&0&1&1&1\\ 0&0&1&0&1&1&1\\ 0&0&1&0&1&1&1 \end{bmatrix}, \quad E_2 = E_1 \cap \Big(\bigcap_{x\in X} (\delta_x \circ E_1^a) | (\delta_x \circ E_1^a)\Big). \begin{bmatrix} 1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&1&0&1&1&1\\ 0&0&0&1&0&0&0\\ 0&0&1&0&1&1&1\\ 0&0&1&0&1&1&1\\ 0&0&1&0&1&1&1 \end{bmatrix}.$$

Now again $R_2^1 \neq E_2^1$ and we choose the first class to compute R_3 and E_3:

$$E_3 = R_3 = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&0&1&0&1&1&1 \\ 0&0&0&1&0&0&0 \\ 0&0&1&0&1&1&1 \\ 0&0&1&0&1&1&1 \\ 0&0&1&0&1&1&1 \end{bmatrix}.$$

The factor automaton $\mathscr{A}/E$ has four states and it is the minimal automaton equivalent to $\mathscr{A}$:



The previous example shows the case where for given non-deterministic automaton the reduction by means of the greatest right invariant equivalence produces the minimal one.

As we already mentioned, the Algorithm 5.3 for computing the greatest right invariant fuzzy quasi-order for fuzzy automata over a Boolean structure runs in $O(mn^5)$, so the procedure for computing the greatest right invariant fuzzy equivalence for fuzzy automata over a Boolean structure, presented in the previous section, also runs in $O(mn^5)$. Therefore, we conclude that the Algorithm 5.6 gives the significant result when we consider the speed of execution.

Appendix A
C♯ codes

Computing the greatest simulations and bisimulations

The following program performs implementation of the Algorithm 3.4 (Chapter 3) for computing the greatest forward and backward simulations and forward, backward, forward-backward and backward-forward bisimulations between two fuzzy automata.

MainWindows.xaml

```
<Window x:Class="releqgraph.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Relational Equations" Height="804" Width="870">
    <Grid Height="747" Margin="0,14,-210,14">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="305*"/>
            <ColumnDefinition Width="767*"/>
        </Grid.ColumnDefinitions>
        <Button Content="Load initial A" Height="23" HorizontalAlignment="Left"
Margin="14,211,0,0" Name="button1" VerticalAlignment="Top" Width="75"
Click="button1_Click" />
        <RadioButton Content="(1) Forward simulacija" Height="16"
HorizontalAlignment="Left" Margin="12,48,0,0" Name="radioButton1" VerticalAlignment="Top"
GroupName="equationType" IsChecked="True" />
        <RadioButton Content="(2) Backward simulacija" Height="16"
HorizontalAlignment="Left" Margin="12,70,0,0" Name="radioButton2" VerticalAlignment="Top"
GroupName="equationType" />
        <RadioButton Content="(3) Forward bisimulacija" Height="16"
HorizontalAlignment="Left" Margin="12,92,0,0" Name="radioButton3" VerticalAlignment="Top"
GroupName="equationType" IsEnabled="True" />
        <TextBox Height="86" HorizontalAlignment="Left" Margin="14,270,0,0"
Name="tbInitialA" VerticalAlignment="Top" Width="172" AcceptsTab="False"
AcceptsReturn="True" TextChanged="tbInitial_TextChanged" />
        <TextBox Height="96" HorizontalAlignment="Left" Margin="14,408,0,0"
Name="tbInitialB" VerticalAlignment="Top" Width="172" AcceptsTab="False"
AcceptsReturn="True" TextChanged="tbInitial_TextChanged" />
        <Menu Height="28" HorizontalAlignment="Left" Name="menu1" VerticalAlignment="Top"
Width="848" Grid.ColumnSpan="2"></Menu>
        <TextBox Height="86" HorizontalAlignment="Left" Margin="276,270,0,0"
Name="tbTransitionA" VerticalAlignment="Top" Width="172" AcceptsReturn="True"
Grid.ColumnSpan="2" TextChanged="tbTransitionA_TextChanged" />
        <TextBox Height="86" HorizontalAlignment="Left" Margin="252,270,0,0"
Name="tbFinalA" VerticalAlignment="Top" Width="172" AcceptsReturn="True" Grid.Column="1"
/>
        <TextBox Height="96" HorizontalAlignment="Left" Margin="276,408,0,0"
Name="tbTransitionB" VerticalAlignment="Top" Width="172" AcceptsReturn="True"
Grid.ColumnSpan="2" />
        <TextBox Height="96" HorizontalAlignment="Left" Margin="252,408,0,0"
Name="tbFinalB" VerticalAlignment="Top" Width="172" AcceptsReturn="True" Grid.Column="1"
/>
        <Button Content="Load transition A" Height="23" HorizontalAlignment="Left"
Margin="276,211,0,0" Name="button2" VerticalAlignment="Top" Width="112"
Click="button2_Click" Grid.ColumnSpan="2" />
        <Button Content="Load final A" Height="23" HorizontalAlignment="Left"
Margin="252.045,211,0,0" Name="button3" VerticalAlignment="Top" Width="75"
Click="button3_Click" Grid.Column="1" />
        <Button Content="Run" Height="23" HorizontalAlignment="Left" Margin="12,124,0,0"
Name="button4" VerticalAlignment="Top" Width="75" Click="button4_Click" />
        <Label Height="28" HorizontalAlignment="Left" Margin="14,161,0,0" Name="message"
VerticalAlignment="Top" IsEnabled="True" FontSize="14" FontWeight="Bold"
Foreground="#FFD92426" />
        <TextBox Height="112" HorizontalAlignment="Left" Margin="14,606,0,0"
Name="tbResult" VerticalAlignment="Top" Width="797" VerticalScrollBarVisibility="Auto"
Grid.ColumnSpan="2" />
```

MainWindows.xaml

```
       <Label Content="Result" Height="28" HorizontalAlignment="Left" Margin="14,564,0,0"
Name="label4" VerticalAlignment="Top" />
        <RadioButton Content="(4) Backward bisimulacija" Height="16"
HorizontalAlignment="Left" Margin="175,48,0,0" Name="radioButton4"
VerticalAlignment="Top" GroupName="equationType" Grid.ColumnSpan="2"
Checked="radioButton4_Checked" />
        <RadioButton Content="(5) Forward-backward bisimulacija" Height="16"
HorizontalAlignment="Left" Margin="175,70,0,0" Name="radioButton5"
VerticalAlignment="Top" GroupName="equationType" Grid.ColumnSpan="2" />
        <RadioButton Content="(6) Backward-forward bisimulacija" Height="16"
HorizontalAlignment="Left" Margin="175,92,0,0" Name="radioButton6"
VerticalAlignment="Top" GroupName="equationType" Grid.ColumnSpan="2" />
        <Button Content="Exit" Height="23" HorizontalAlignment="Right"
Margin="0,724,37,0" Name="button5" VerticalAlignment="Top" Width="75"
Click="button5_Click" Grid.Column="1" />
        <Button Content="Save Result" Height="23" HorizontalAlignment="Left"
Margin="431,723,0,0" Name="button6" VerticalAlignment="Top" Width="75"
Click="button6_Click" DataContext="{Binding}" Grid.Column="1" />
        <Button Content="Clear Result" Height="23" HorizontalAlignment="Left"
Margin="93,124,0,0" Name="button7" VerticalAlignment="Top" Width="75"
Click="button7_Click" />
        <Button Content="Button" Height="23" HorizontalAlignment="Left"
Margin="428.045,215,0,0" Name="button8" VerticalAlignment="Top" Width="75"
Click="button8_Click" Grid.Column="1" />
        <Button Content="Load initial B" Height="23" HorizontalAlignment="Left"
Margin="14,380,0,0" Name="button9" VerticalAlignment="Top" Width="75"
Click="button9_Click" RenderTransformOrigin="0.427,7.217" />
        <Button Content="Load transition B" Height="23" HorizontalAlignment="Left"
Margin="276,380,0,0" Name="button10" VerticalAlignment="Top" Width="112"
Click="button10_Click" Grid.ColumnSpan="2" />
        <Button Content="Load final B" Height="23" HorizontalAlignment="Left"
Margin="252,380,0,0" Name="button11" VerticalAlignment="Top" Width="75"
Click="button11_Click" Grid.Column="1" />
    </Grid>
</Window>
```

MainWindows.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO;
using Microsoft.Win32;

namespace releqgraph
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    ///

    public partial class MainWindow : Window
    {
        public const int NUMBER_OF_ITERATIONS = 10;

        public MainWindow()
        {
            InitializeComponent();
        }
        //Load initial A
        private void button1_Click(object sender, RoutedEventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.DefaultExt = "txt";
            if (openFileDialog.ShowDialog().Value)
            {
                string fullPathname = openFileDialog.FileName;
                FileInfo src = new FileInfo(fullPathname);
                TextReader reader = src.OpenText();
                string line = reader.ReadLine();
                tbInitialA.Text = "";
                while (line != null)
                {
                    tbInitialA.Text += line + '\n';
                    line = reader.ReadLine();
                }
            }
        }
        //Load transition A
        private void button2_Click(object sender, RoutedEventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.DefaultExt = "txt";
            if (openFileDialog.ShowDialog().Value)
        {
```

MainWindows.cs

```
            string fullPathname = openFileDialog.FileName;
            FileInfo src = new FileInfo(fullPathname);
            TextReader reader = src.OpenText();
            string line = reader.ReadLine();
            while (line != null)
            {
                tbTransitionA.Text += line + '\n';
                line = reader.ReadLine();
            }
        }
    }
    //Load final A
    private void button3_Click(object sender, RoutedEventArgs e)
    {
        OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.DefaultExt = "txt";
        if (openFileDialog.ShowDialog().Value)
        {
            string fullPathname = openFileDialog.FileName;
            FileInfo src = new FileInfo(fullPathname);
            TextReader reader = src.OpenText();
            string line = reader.ReadLine();
            //tbInitial.Text = "";
            while (line != null)
            {
                tbFinalA.Text += line + '\n';
                line = reader.ReadLine();
            }
        }
    }

    //Load initial B
    private void button9_Click(object sender, RoutedEventArgs e)
    {
        OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.DefaultExt = "txt";
        if (openFileDialog.ShowDialog().Value)
        {
            string fullPathname = openFileDialog.FileName;
            FileInfo src = new FileInfo(fullPathname);
            TextReader reader = src.OpenText();
            string line = reader.ReadLine();
            tbInitialB.Text = "";
            while (line != null)
            {
                tbInitialB.Text += line + '\n';
                line = reader.ReadLine();
            }
        }
    }
    //Load transition B
    private void button10_Click(object sender, RoutedEventArgs e)
    {
        OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.DefaultExt = "txt";
        if (openFileDialog.ShowDialog().Value)
```

MainWindows.cs

```
                {
                 string fullPathname = openFileDialog.FileName;
                  FileInfo src = new FileInfo(fullPathname);
                  TextReader reader = src.OpenText();
                  string line = reader.ReadLine();
                  while (line != null)
                  {
                      tbTransitionB.Text += line + '\n';
                      line = reader.ReadLine();
                  }
              }
          }
          //Load final B
          private void button11_Click(object sender, RoutedEventArgs e)
          {
              OpenFileDialog openFileDialog = new OpenFileDialog();
              openFileDialog.DefaultExt = "txt";
              if (openFileDialog.ShowDialog().Value)
              {
                  string fullPathname = openFileDialog.FileName;
                  FileInfo src = new FileInfo(fullPathname);
                  TextReader reader = src.OpenText();
                  string line = reader.ReadLine();
                  //tbInitial.Text = "";
                  while (line != null)
                  {
                      tbFinalB.Text += line + '\n';
                      line = reader.ReadLine();
                  }
              }
          }


          private void button4_Click(object sender, RoutedEventArgs e)
          {
              try
              {
                  //parsing relation from textbox
                  Function SigmaA = new Function();
                  RelationArray DeltaArrayA = new RelationArray();
                  Function TauA = new Function();
                  Function SigmaB = new Function();
                  RelationArray DeltaArrayB = new RelationArray();
                  Function TauB = new Function();
                  SigmaA.loadFromStringArray(tbInitialA.Text);
                  DeltaArrayA.loadFromStringMatrix(tbTransitionA.Text);
                  TauA.loadFromStringArray(tbFinalA.Text);
                  SigmaB.loadFromStringArray(tbInitialB.Text);
                  DeltaArrayB.loadFromStringMatrix(tbTransitionB.Text);
                  TauB.loadFromStringArray(tbFinalB.Text);

                  tbResult.Text += '\n';
                  tbResult.Text += "================ Initial A =================";
                  SigmaA.writeToTextBox(tbResult);
                  tbResult.Text += "================ Transition A =================";
```

MainWindows.cs

```
                tbResult.Text += DeltaArrayA.toString();
                tbResult.Text += "================ Final A =================";
                TauA.writeToTextBox(tbResult);
                tbResult.Text += '\n';
                tbResult.Text += "================ Initial B =================";
                SigmaB.writeToTextBox(tbResult);
                tbResult.Text += "================ Transition B =================";
                tbResult.Text += DeltaArrayB.toString();
                tbResult.Text += "================ Final B =================";
                TauB.writeToTextBox(tbResult);

                if (radioButton1.IsChecked == true)
                    solve1(SigmaA, DeltaArrayA, TauA, SigmaB, DeltaArrayB, TauB,
tbResult);
                else if (radioButton2.IsChecked == true)
                    solve2(SigmaA, DeltaArrayA, TauA, SigmaB, DeltaArrayB, TauB,
tbResult);
                else if (radioButton3.IsChecked == true)
                    solve3(SigmaA, DeltaArrayA, TauA, SigmaB, DeltaArrayB, TauB,
tbResult);
                else if (radioButton4.IsChecked == true)
                    solve4(SigmaA, DeltaArrayA, TauA, SigmaB, DeltaArrayB, TauB,
tbResult);
                else if (radioButton5.IsChecked == true)
                    solve5(SigmaA, DeltaArrayA, TauA, SigmaB, DeltaArrayB, TauB,
tbResult);
                else if (radioButton6.IsChecked == true)
                    solve6(SigmaA, DeltaArrayA, TauA, SigmaB, DeltaArrayB, TauB,
tbResult);

            }
            catch (Exception ex)
            {
                message.Content = ex.Message + ex.StackTrace;
            }
        }

        private void solve1(Function SigmaA, RelationArray DeltaArrayA, Function TauA,
Function SigmaB, RelationArray DeltaArrayB, Function TauB, TextBox tb)
        {
            Relation X1 = new Relation(TauA.Rightarrow(TauB));
            Relation pom = new Relation(X1);
           Relation Inverzija = new Relation(X1.inverse());
           Function KompStart = new Function(SigmaB.fcomposition(Inverzija));
           Boolean startcond = SigmaA.Less(KompStart);
            Boolean forward = true;
            int numberOfIterations = 0;
            while (startcond && forward && numberOfIterations <= NUMBER_OF_ITERATIONS)
            {
                numberOfIterations++;
                tb.Text += '\n';
                tb.Text += "========== Iteration: " + numberOfIterations + " ==========";
                tb.Text += "========== X "+ numberOfIterations + "  ==========";
                pom.writeToTextBox(tb);
                Relation next = new Relation(pom.fs(DeltaArrayA, DeltaArrayB));
                tb.Text += "========== Iteration: " + numberOfIterations + " ==========";
                tb.Text += "========== next " + numberOfIterations + "  ==========";
                next.writeToTextBox(tb);
                startcond = SigmaA.Less(SigmaB.fcomposition(next.inverse()));
                if (next.Equals(pom))
```

MainWindows.cs

```
                forward = false;
                pom = new Relation(next);

            }
            if (startcond == false)
                tb.Text += "========== No forward simulation between A and B ==========";
        }

        private void solve2(Function SigmaA, RelationArray DeltaArrayA, Function TauA,
Function SigmaB, RelationArray DeltaArrayB, Function TauB, TextBox tb)
        {
            Relation prev = new Relation(SigmaA.Rightarrow(SigmaB));
            Boolean forward = true;
            Boolean startcond = TauA.Less(TauB.compositionf(prev));
            int numberOfIterations = 0;
            while (startcond && forward && numberOfIterations <= NUMBER_OF_ITERATIONS)
            {
                numberOfIterations++;
                tb.Text += '\n';
                tb.Text += "========== Iteration: " + numberOfIterations + " ==========";

                tb.Text += '\n';
                tb.Text += "========== X " + numberOfIterations + "  ==========";
                prev.writeToTextBox(tb);
                Relation next = new Relation(prev.bs(DeltaArrayA, DeltaArrayB));
                tb.Text += "========== Iteration: " + numberOfIterations + " ==========";
                tb.Text += "========== next " + numberOfIterations + "  ==========";
                next.writeToTextBox(tb);
                startcond = TauA.Less(TauB.compositionf(next));
                if (next == prev)
                    forward = false;
                prev = new Relation(next);
            }
            if (startcond == false)
                tb.Text += "========== No backward simulation between A and B
==========";
        }

        private void solve3(Function SigmaA, RelationArray DeltaArrayA, Function TauA,
Function SigmaB, RelationArray DeltaArrayB, Function TauB, TextBox tb)
        {
            Relation X11 = new Relation(TauA.Rightarrow(TauB));
            Relation pom1 =new Relation(TauA.Leftarrow(TauB));
            Relation X1 = new Relation(X11.infimum(pom1));
            Boolean forwardb = true;
            Boolean startcond1 = SigmaA.Less(SigmaB.fcomposition(X1.inverse()));
            Boolean startcond2 = SigmaB.Less(SigmaA.fcomposition(X1));
            Boolean startcond = startcond1 && startcond2;
            int numberOfIterations = 0;
            while (startcond && forwardb && numberOfIterations <= NUMBER_OF_ITERATIONS)
            {
                numberOfIterations++;
```

MainWindows.cs

```
                tb.Text += '\n';
                tb.Text += "========== Iteration: " + numberOfIterations + " ==========";
                tb.Text += '\n';
                tb.Text += "========== X " + numberOfIterations + " ==========";
                X1.writeToTextBox(tb);

                Relation X2 = new Relation(X1.fb(DeltaArrayA, DeltaArrayB));
                startcond1 = SigmaA.Less(SigmaB.fcomposition(X2.inverse()));
                startcond2 = SigmaB.Less(SigmaA.fcomposition(X2));
                startcond = startcond1 && startcond2;
                if (X2 == X1)
                    forwardb = false;
                X1 = new Relation(X2);
            }
            if (startcond == false)
                tb.Text += "========== No forward bisimulation between A and B
==========";
        }

        private void solve4(Function SigmaA, RelationArray DeltaArrayA, Function TauA,
Function SigmaB, RelationArray DeltaArrayB, Function TauB, TextBox tb)
        {
            Relation X11 = new Relation(SigmaA.Rightarrow(SigmaB));
            Relation X1 = new Relation(X11.infimum(SigmaA.Leftarrow(SigmaB)));
            Boolean startcond1 = TauA.Less(TauB.compositionf(X1));
            Boolean startcond2 = TauB.Less(TauA.compositionf(X1.inverse()));
            Boolean startcond = startcond1 && startcond2;
            Boolean forward = true;
            int numberOfIterations = 0;
            while (startcond && forward && numberOfIterations <= NUMBER_OF_ITERATIONS)
            {
                numberOfIterations++;
                tb.Text += '\n';
                tb.Text += "========== Iteration: " + numberOfIterations + " ==========";
                tb.Text += '\n';
                tb.Text += "========== X " + numberOfIterations + " ==========";
                X1.writeToTextBox(tb);
                Relation X2 = new Relation(X1.bb(DeltaArrayA, DeltaArrayB));
                startcond1 = TauA.Less(TauB.compositionf(X2));
                startcond2 = TauB.Less(TauA.compositionf(X2.inverse()));
                startcond = startcond1 && startcond2;
                if (X2 == X1)
                    forward = false;
                X1 = new Relation(X2);
            }
            if (startcond == false)
                tb.Text += "========== No backward bisimulation between A and B
==========";
        }

        private void solve5(Function SigmaA, RelationArray DeltaArrayA, Function TauA,
Function SigmaB, RelationArray DeltaArrayB, Function TauB, TextBox tb)
        {
            Relation X11 = new Relation(TauA.Rightarrow(TauB));
            Relation X1 = new Relation(X11.infimum(SigmaA.Leftarrow(SigmaB)));
```

MainWindows.cs

```
            Boolean startcond1 = SigmaA.Less(SigmaB.fcomposition(X1.inverse()));
            Boolean startcond2 = TauB.Less(TauA.compositionf(X1.inverse()));
            Boolean startcond = startcond1 && startcond2;
            Boolean forward = true;
            int numberOfIterations = 0;
            while (startcond && forward && numberOfIterations <= NUMBER_OF_ITERATIONS)
            {
                numberOfIterations++;

                tb.Text += '\n';
                tb.Text += "========== Iteration: " + numberOfIterations + " ==========";
                tb.Text += '\n';
                tb.Text += "========== X " + numberOfIterations + " ==========";
                X1.writeToTextBox(tb);
                Relation X2 = new Relation(X1.fbb(DeltaArrayA, DeltaArrayB));
                startcond1 = SigmaA.Less(SigmaB.fcomposition(X2.inverse()));
                startcond2 = TauB.Less(TauA.compositionf(X2.inverse()));
                startcond = startcond1 && startcond2;
                if (X2 == X1)
                    forward = false;
                X1 = new Relation(X2);
            }
            if (startcond == false)
                tb.Text += "========== No forward-backward bisimulation between A and B
==========";
        }



        private void solve6(Function SigmaA, RelationArray DeltaArrayA, Function TauA,
Function SigmaB, RelationArray DeltaArrayB, Function TauB, TextBox tb)
        {
            Relation X11 = new Relation(SigmaA.Rightarrow(SigmaB));
            Relation X1 = new Relation(X11.infimum(TauA.Leftarrow(TauB)));
            Boolean startcond1 = TauA.Less(TauB.compositionf(X1));
            Boolean startcond2 = SigmaB.Less(SigmaA.fcomposition(X1));
            Boolean startcond = startcond1 && startcond2;
            Boolean forward = true;
            int numberOfIterations = 0;
            while (startcond && forward && numberOfIterations <= NUMBER_OF_ITERATIONS)
            {
                numberOfIterations++;
                tb.Text += '\n';
                tb.Text += "========== Iteration: " + numberOfIterations + " ==========";
                tb.Text += '\n';
                tb.Text += "========== X " + numberOfIterations + " ==========";
                X1.writeToTextBox(tb);
                Relation X2 = new Relation(X1.bfb(DeltaArrayA, DeltaArrayB));
                startcond1 = TauA.Less(TauB.compositionf(X2));
                startcond2 = SigmaB.Less(SigmaA.fcomposition(X2));
                startcond = startcond1 && startcond2;
                if (X2 == X1)
                    forward = false;
```

MainWindows.cs

```
                X1 = new Relation(X2);
            }
            if (startcond == false)
                tb.Text += "========== No backward-forward bisimulation between A and B
==========";
        }

        private void button5_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }

       private void button6_Click(object sender, RoutedEventArgs e)
        {
            SaveFileDialog saveDialog = new SaveFileDialog();
            saveDialog.DefaultExt = ".txt";
            saveDialog.AddExtension = true;
            saveDialog.FileName = "Result";
            saveDialog.OverwritePrompt = true;
            saveDialog.Title = "Save Result";
            saveDialog.ValidateNames = true;
            if (saveDialog.ShowDialog().Value)
            {
                // create a writer and open the file
                StreamWriter tw = new StreamWriter(saveDialog.FileName);
                // write a line of text to the file
                tw.WriteLine(tbResult.Text);
                // close the stream
                tw.Close();
            }
        }

        private void button7_Click(object sender, RoutedEventArgs e)
        {
            tbResult.Text = "";
        }

        private void button8_Click(object sender, RoutedEventArgs e)
        {

        }

        private void radioButton4_Checked(object sender, RoutedEventArgs e)
        {

        }

        private void tbInitial_TextChanged(object sender, TextChangedEventArgs e)
        {

        }

        private void tbTransitionA_TextChanged(object sender, TextChangedEventArgs e)
        {
```

Relation.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Controls;

namespace releqgraph
{
    class Relation
    {
        private int nRows;
        private int mCols;

        private double[,] mat;

        public Relation()
        {
        }

        public Relation(int n, int m)
        {
            nRows = n;
            mCols = m;
            mat = new double[n, m];
        }

        public Relation(int n)
        {
            nRows = n;
            mCols = n;
            mat = new double[n, n];
        }
        //Pravi kopiju matrice
        public Relation(Relation C)
        {
            nRows = C.getNRows();
            mCols = C.getMCols();
            mat = new double[nRows, mCols];
            for (int i = 0; i < nRows; i++)
                for (int j = 0; j < mCols; j++)
                    mat[i, j] = C.getElem(i, j);
        }
        //Vraca koliko ima redova
        public int getNRows()
        {
            return nRows;
        }
        //Postavlja broj redova na n
        public void setNRows(int n)
        {
            nRows = n;
        }
        //Vraca koliko ima kolona
        public int getMCols()
        {
```

Relation.cs

```
 return mCols;
}
//Postavlja broj kolona na m
public void setMCols(int m)
{
    mCols = m;
}
// vraca odgovarajuci element matrice
public double getElem(int i, int j)
{
    return this.mat[i, j];
}
// postavlja vrednost odgovarajucem elementu matrice
public void setElem(int i, int j, double elem)
{
    this.mat[i, j] = elem;
}
// postavlja sve elemente matrice na 1
public void allOnes(int n)
{
    nRows = n;
    mCols = n;
    mat = new double[n, n];
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            mat[i, j] = 1;
}
// ucitava string koji smo uneli i pravi matricu
public void loadFromString(string s)
{
    string[] lines = Regex.Split(s, "\n");
    List<string> linesList = new List<string>(lines);
    //remove empty lines
    List<string> itemsToRemove = new List<string>();
    foreach (string l in linesList)
    {
        l.Replace("\r", "");
        l.Replace("\n", "");
        l.Trim();
        if (l.Equals(string.Empty) || l.Equals("\r") || l.Equals("\n"))
            itemsToRemove.Add(l);
    }
    foreach (string l in itemsToRemove)
        linesList.Remove(l);
    //check if list contains more then 1 line
    if (linesList.Count() <= 1)
        throw new Exception("Unable to parse matrix!");
    //in the first line we have matrix dimensions
    string[] dims = Regex.Split(linesList[0], " ");
    this.nRows = Convert.ToInt32(dims[0]);
    this.mCols = Convert.ToInt32(dims.Length > 1 ? dims[1] : dims[0]);

    if (linesList.Count() != nRows + 1)
        throw new Exception("Wrong number of rows specified!");
```

Relation.cs

```
 //create relation matrix
            mat = new double[nRows, mCols];

            for (int i = 1; i < linesList.Count(); i++)
            {
                string[] values = Regex.Split(linesList[i], " ");
                if (values.Length != mCols)
                    throw new Exception("Wrong number of colums specified in the row: " +
i);
                for (int j = 0; j < values.Length; j++)
                {
                    mat[i - 1, j] = Convert.ToDouble(values[j]);
                }
            }
        }
        // upis u tekst boks
        public void writeToTextBox(TextBox tb)
        {
            tb.Text += '\n';
            tb.Text += nRows + " " + mCols + '\n';
            for (int i = 0; i < nRows; i++)
            {
                for (int j = 0; j < mCols; j++)
                {
                    tb.Text += mat[i, j];
                    if (j == mCols - 1)
                        tb.Text += '\n';
                    else
                        tb.Text += ' ';
                }
            }
        }
        public String toString()
        {
            String stringRepresentation = "";
            stringRepresentation += this.nRows + " " + this.mCols + "\n";
            for (int i = 0; i < this.nRows; i++)
            {
                for (int j = 0; j < this.mCols; j++)
                {
                    stringRepresentation += this.mat[i, j];
                    if (j < this.mCols - 1)
                        stringRepresentation += " ";
                }
                stringRepresentation += "\n";
            }
            return stringRepresentation;
        }
        public Relation rightResidual1(Relation B)
        {
            if (this.nRows != B.getNRows() || this.mCols != B.getMCols())
                throw new Exception("Incompatible relations for right residual!");
            Relation result = new Relation(this.mCols);
            for (int i = 0; i < result.getNRows(); i++)
            {
```

Relation.cs

```
                for (int j = 0; j < result.getMCols(); j++)
                {
                    if (i == j)
                        result.setElem(i, j, 1);
                    else
                    {
                        int el = 1;
                        for (int k = 0; k < this.nRows && el == 1; k++)
                            if (this.mat[k, i] > B.getElem(k, j))
                                el = 0;
                        result.setElem(i, j, el);
                    }
                }
            }
            return result;
        }
        // levi rezidual tj. A/B
        public Relation leftResidual1(Relation B)
        {
            if (this.nRows != B.getNRows() || this.mCols != B.getMCols())
                throw new Exception("Incompatible relations for left residual!");
            Relation result = new Relation(this.mCols);
            for (int i = 0; i < result.getNRows(); i++)
            {
                for (int j = 0; j < result.getMCols(); j++)
                {
                    if (i == j)
                        result.setElem(i, j, 1);
                    else
                    {
                        int el = 1;
                        for (int k = 0; k < this.mCols && el == 1; k++)
                            if (this.mat[j, k] > B.getElem(i, k))
                                el = 0;
                        result.setElem(i, j, el);
                    }
                }
            }
            return result;
        }
        //A^{-1}
        public Relation inverse()
        {
            Relation result = new Relation(this.mCols, this.nRows);
            for (int i = 0; i < result.getNRows(); i++)
                for (int j = 0; j < result.getMCols(); j++)
                    result.setElem(i, j, this.getElem(j, i));
            return result;
        }
        // infimum dve relacije
        public Relation infimum1(Relation B)
        {
            if (this.nRows != B.getNRows() || this.mCols != B.getMCols())
                throw new Exception("Incompatible relations for infimum");
            Relation result = new Relation(this.nRows, this.mCols);
```

Relation.cs

```
       for (int i = 0; i < this.nRows; i++)
                for (int j = 0; j < this.mCols; j++)
                    result.setElem(i, j, this.mat[i, j] * B.getElem(i, j));
            return result;
        }



        public static Boolean operator ==(Relation A, Relation B)
        {
            if (A.getNRows() != B.getNRows() || A.getMCols() != B.getMCols())
                return false;
            Boolean eq = true;
            for (int i = 0; i < A.getNRows() && eq; i++)
                for (int j = 0; j < A.getMCols() && eq; j++)
                    if (A.getElem(i, j) != B.getElem(i, j))
                        eq = false;
            return eq;
        }

        public static Boolean operator !=(Relation A, Relation B)
        {
            if (A.getNRows() != B.getNRows() || A.getMCols() != B.getMCols())
                return true;
            for (int i = 0; i < A.getNRows(); i++)
                for (int j = 0; j < A.getMCols(); j++)
                    if (A.getElem(i, j) != B.getElem(i, j))
                        return true;
            return false;
        }

        public override bool Equals(Object B)
        {
            if (!(B is Relation))
                return false;
            if (this.nRows != ((Relation)B).getNRows() || this.mCols !=
((Relation)B).getMCols())
                return false;
            for (int i = 0; i < this.nRows; i++)
                for (int j = 0; j < this.mCols; j++)
                    if (this.mat[i, j] != ((Relation)B).getElem(i, j))
                        return false;
            return true;
        }

        public override int GetHashCode()
        {
            return base.GetHashCode();
        }

        private int and(int a, int b)
        {
            return a * b;
        }
```

Relation.cs

```
        private int or(int a, int b)
        {
            if (a + b == 2)
                return 1;
            else
                return a + b;
        }


        // moj A\B
        public Relation rightResidual(Relation B)
        {
            if (this.nRows != B.getNRows())
                throw new Exception("Incompatible relations for right residual!");
            Relation result = new Relation(this.mCols, B.mCols);
            Function pom = new Function(this.nRows);
            for (int i = 0; i < result.getNRows(); i++)
            {
                for (int j = 0; j < result.getMCols(); j++)
                {
                    for (int k = 0; k < pom.getNElems(); k++)
                        if (this.mat[k, i] <= B.getElem(k, j) )
                            pom.setFelem(k, 1);
                        else
                            pom.setFelem(k, (B.getElem(k, j)/this.mat[k, i]));
                    result.setElem(i, j, pom.getMin());
                }
            }
            return result;
        }
        // moj A/B
        public Relation leftResidual(Relation B)
        {
            if (this.mCols!= B.getMCols())
                throw new Exception("Incompatible relations for left residual!");
            Relation result = new Relation(this.nRows, B.nRows);
            Function pom = new Function(this.mCols);
            for (int i = 0; i < result.getNRows(); i++)
            {
                for (int j = 0; j < result.getMCols(); j++)
                {
                    for (int k = 0; k < pom.getNElems() ; k++)
                        if (B.getElem(j, k) <= this.mat[i, k])
                            pom.setFelem(k, 1);
                        else
                            pom.setFelem(k, (this.mat[i, k] / B.getElem(j, k)));
                    result.setElem(i, j, pom.getMin());
                }
            }
            return result;
        }
        public Relation composition(Relation B)
        {
            if (this.mCols != B.getNRows())
                throw new Exception("Incompatible relations for composition");
```

Relation.cs

```
            Relation result = new Relation(this.nRows, B.getMCols());
            Function pom = new Function(this.mCols);
            for (int i = 0; i < this.nRows; i++)
            {
                for (int j = 0; j < B.getMCols(); j++)
                {
                    for (int k = 0; k < pom.getNElems(); k++)
                        pom.setFelem( k, (this.mat[i, k] * B.getElem(k, j)));
                    result.setElem(i, j, pom.getMax());
                }
            }
            return result;
        }

        // infimum dve relacije
        public Relation infimum(Relation B)
        {
            if (this.nRows != B.getNRows() || this.mCols != B.getMCols())
                throw new Exception("Incompatible relations for infimum");
            Relation result = new Relation(this.nRows, this.mCols);
            for (int i = 0; i < this.nRows; i++)
                for (int j = 0; j < this.mCols; j++)
                {
                    if (this.mat[i, j] < B.getElem(i, j))
                        result.setElem(i, j, this.mat[i, j]);
                    else
                        result.setElem(i, j, B.getElem(i, j));
                }
            return result;
        }


        public Relation bs(RelationArray DeltaArrayA, RelationArray DeltaArrayB)
        {
            Relation previous = new Relation(this);
            for (int i = 0; i < DeltaArrayA.getNumberOfRelations(); i++)
            {
                Relation compRel = new
Relation(this.composition(DeltaArrayB.getRelation(i)));
                Relation rightResidual = new
Relation(DeltaArrayA.getRelation(i).rightResidual(compRel));
                Relation next = new Relation(previous.infimum(rightResidual));
                previous = next;
            }
            return previous;
        }
        public Relation fs(RelationArray DeltaArrayA, RelationArray DeltaArrayB)
        {
            Relation previous = new Relation(this);
            Relation invRel = new Relation(this.inverse());
            for (int i = 0; i < DeltaArrayA.getNumberOfRelations(); i++)
            {
                Relation compRel = new
Relation(DeltaArrayB.getRelation(i).composition(invRel));
```

```
                Relation leftResid = new
Relation(compRel.leftResidual(DeltaArrayA.getRelation(i)));
                Relation invRel1 = new Relation(leftResid.inverse());
                Relation next = new Relation(previous.infimum(invRel1));
                previous = next;
            }
                return previous;
        }
        public Relation fb(RelationArray DeltaArrayA, RelationArray DeltaArrayB)
        {
            Relation previous = new Relation(this);
            Relation invRel = new Relation(this.inverse());
            for (int i = 0; i < DeltaArrayA.getNumberOfRelations(); i++)
            {
                Relation compRel1 = new
Relation(DeltaArrayB.getRelation(i).composition(invRel));
                Relation compRel2 = new
Relation(DeltaArrayA.getRelation(i).composition(this));
                Relation leftResid1 = new
Relation(compRel1.leftResidual(DeltaArrayA.getRelation(i)));
                Relation leftResid2 = new
Relation(compRel2.leftResidual(DeltaArrayB.getRelation(i)));
                Relation invRel1 = new Relation(leftResid1.inverse());
                Relation pomInf = new Relation(invRel1.infimum(leftResid2));
                Relation next = new Relation(previous.infimum(pomInf));
                previous = next;
            }
            return previous;
        }
        public Relation bb(RelationArray DeltaArrayA, RelationArray DeltaArrayB)
        {
            Relation previous = new Relation(this);
            Relation invRel = new Relation(this.inverse());
            for (int i = 0; i < DeltaArrayA.getNumberOfRelations(); i++)
            {
                Relation compRel1 = new
Relation(this.composition(DeltaArrayB.getRelation(i)));
                Relation compRel2 = new
Relation(invRel.composition(DeltaArrayA.getRelation(i)));
                Relation rightResidual1 = new
Relation(DeltaArrayA.getRelation(i).rightResidual(compRel1));
                Relation rightResidual2 = new
Relation(DeltaArrayB.getRelation(i).rightResidual(compRel2));
                Relation invRel2 = new Relation(rightResidual2.inverse());
                Relation pomInf = new Relation(rightResidual1.infimum(invRel2));
                Relation next = new Relation(previous.infimum(pomInf));
                previous = next;
            }
            return previous;
        }

        public Relation fbb(RelationArray DeltaArrayA, RelationArray DeltaArrayB)
        {
            Relation previous = new Relation(this);
            Relation invRel = new Relation(this.inverse());
            for (int i = 0; i < DeltaArrayA.getNumberOfRelations(); i++)
            {
```

Function.cs

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Linq;
using System.Text;
using System.Windows.Controls;

namespace releqgraph
{
    class Function
    {
        private int nElems;
        private double[] kol;

        public Function()
        {
        }

        public Function(int n)
        {
            nElems = n;
            kol = new double[n];
        }
        //kopira fju
         public Function(Function C)
        {
            nElems = C.getNElems();
            kol = new double[nElems];
            for (int i = 0; i < nElems; i++)
                    kol[i] = C.getFelem(i);
        }
        //Vraca koliko ima elemenata
        public int getNElems()
        {
            return nElems;
        }
        //Postavlja broj elemenata na n
        public void setElems(int n)
        {
            nElems = n;
        }
        // vraca odgovarajuci element niza
        public double getFelem(int i)
        {
            return this.kol[i];
        }

        // vraca min
        public double getMin()
        {
            double pom = this.getFelem(0);
            for (int i = 1; i < this.getNElems(); i++)
                if (this.getFelem(i) < pom)
                    pom = this.getFelem(i);
            return pom;
        }
```

Function.cs

```
 // vraca max
        public double getMax()
        {
            double pom = this.getFelem(0);
            for (int i = 1; i < this.getNElems(); i++)
                if (this.getFelem(i) > pom)
                    pom = this.getFelem(i);
            return pom;
        }

        // postavlja vrednost odgovarajucem elementu matrice
        public void setFelem(int i, double elem)
        {
            this.kol[i] = elem;
        }


        // postavlja sve elemente niza na 1
        public void allOneArray(int n)
        {
            nElems = n;

            kol = new double[n];
            for (int i = 0; i < n; i++)
                    kol[i] = 1;
        }

        // ucitava string koji smo uneli i pravi niz
        public void loadFromStringArray (string s)
        {
            string[] lines = Regex.Split(s, "\n");
            List<string> linesList = new List<string>(lines);
            linesList.ForEach(delegate(string l)
            {
                l.Trim();
            });
            //remove empty lines
            List<string> itemsToRemove = new List<string>();
            foreach (string l in linesList)
            {
                if (l.Equals(string.Empty))
                    itemsToRemove.Add(l);
            }
            foreach (string l in itemsToRemove)
                linesList.Remove(l);
            //check if list contains more then 1 line
            if (linesList.Count() <= 1)
                throw new Exception("Unable to parse array!" +
linesList.Count().ToString());
            //in the first line we have array dimension
            string[] dims = Regex.Split(linesList[0], " ");
            this.nElems = Convert.ToInt32(dims[0]);

            if (linesList.Count() != 2)
                throw new Exception("Wrong number of rows specified!");
```

Function.cs

```
            //create relation array
            kol = new double[nElems];


                string[] values = Regex.Split(linesList[1], " ");
                if (values.Length != nElems)
                    throw new Exception("Wrong number of elements specified in the row.
");
                for (int j = 0; j < values.Length; j++)
                {
                   kol[j] = Convert.ToDouble(values[j]);
                }

        }
        // upis u tekst boks
        public void writeToTextBox(TextBox tb)
        {
            tb.Text += '\n';
            tb.Text += nElems +  '\n';
            for (int i = 0; i < nElems; i++)
            {

                    tb.Text += kol[i];
                    tb.Text += ' ';
            }
        }

        // right arrow
        public Relation Rightarrow(Function B)
        {
            Relation result = new Relation(this.nElems, B.getNElems());
            for (int i = 0; i < result.getNRows(); i++)
            {
                for (int j = 0; j < result.getMCols(); j++)
                {
                    if (this.kol[i] <= B.getFelem(j))
                        result.setElem(i, j, 1);
                    else
                        result.setElem(i, j, (B.getFelem(j)/this.kol[i]) );
                }
            }
            return result;
        }
        // left arrow
        public Relation Leftarrow(Function B)
        {
            Relation result = new Relation(this.getNElems(),B.getNElems());
            for (int i = 0; i < result.getNRows(); i++)
            {
                for (int j = 0; j < result.getMCols(); j++)
                {
                    if (B.getFelem(j) <= this.getFelem(i))
                        result.setElem(i, j, 1);
                    else
                        result.setElem(i, j,(this.getFelem(i)/B.getFelem(j)));
```

Function.cs

```
                }
            }
            return result;
        }
        public Boolean Less(Function B)
        {
            Boolean result = true;
            int i = 0;
            if ( this.nElems != B.getNElems())
                result = false;
            while (result && i < this.nElems)
            {
                if (this.kol[i]>B.getFelem(i))
                    result =false;
                i++;
            }
            return result;
        }
        public Function fcomposition(Relation B)
        {
            if (this.nElems != B.getNRows())
                throw new Exception("Incompatible relations for composition");
            Function result = new Function(B.getMCols());
            Function pom = new Function(this.nElems);
            for (int i = 0; i < result.nElems; i++)
            {
                    for (int k = 0; k < pom.getNElems(); k++)
                        pom.setFelem(k, (this.kol[k] * B.getElem(k, i)));
                    result.setFelem(i, pom.getMax());

            }
            return result;
        }
        public Function compositionf(Relation B)
        {
            if (this.nElems != B.getMCols())
                throw new Exception("Incompatible relations for composition");
            Function result = new Function(B.getNRows());
            Function pom = new Function(this.nElems);
            for (int i = 0; i < result.nElems; i++)
            {
                for (int k = 0; k < pom.getNElems(); k++)
                    pom.setFelem(k,(B.getElem(i,k)* this.kol[k]));
                result.setFelem(i, pom.getMax());

            }
            return result;
        }
    }

}
```

RelationArray.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Controls;

namespace releqgraph
{
    class RelationArray
    {

        List<Relation> relArray = new List<Relation>();

        public RelationArray()
        {
        }

        public void addNewRelation(Relation r)
        {
            relArray.Add(r);
        }

        public int getNumberOfRelations()
        {
            return this.relArray.Count();
        }
        public Relation getRelation(int i)
        {
            return this.relArray[i];
        }
        public void loadFromStringMatrix(string s)
        {
            string[] lines = Regex.Split(s, "\n");
            List<string> linesList = new List<string>(lines);
            linesList.ForEach(delegate(string l)
                            {
                                l.Replace("\r", "");
                                l.Replace("\n", "");
                                l.Trim();
                            });
            //remove empty lines
            List<string> itemsToRemove = new List<string>();
            foreach (string l in linesList)
            {
                if (l.Equals(string.Empty) || l.Equals("\r") || l.Equals("\n"))
                    itemsToRemove.Add(l);
            }
            foreach (string l in itemsToRemove)
                linesList.Remove(l);
            //check if list contains more then 1 line
            if (linesList.Count() <= 1)
                throw new Exception("Unable to parse matrix!");
            //in the first line we have matrix dimensions
            string[] dims = Regex.Split(linesList[0], " ");
            int numberOfRelations = Convert.ToInt32(dims[0]);
```

RelationArray.cs

```
 int dimension = Convert.ToInt32(dims[1]);

            for (int i = 0; i < numberOfRelations; i++)
            {
                String relStr = dims[1] + " " + dims[1] + "\n";
                for (int j = i * dimension + 1; j < (i + 1) * dimension + 1; j++)
                    relStr += linesList[j] + "\n";
                Relation r = new Relation();
                r.loadFromString(relStr);
                relArray.Add(r);
            }

        }

        public String toString()
        {
            String strRepresentation = "";
            for (int i = 0; i < relArray.Count(); i++)
            {
                strRepresentation += relArray[i].toString();
                strRepresentation += "\n";
            }
            return strRepresentation;
        }
    }
}
```

Appendix B
Bisimulacije za fazi automate

Teorija fazi skupova nastaje iz potrebe da se odredjeni koncepti ljudskog znanja predstave pomoću matematičkih modela. Pojam fazi skupa uveo je L.A.Zadeh 1965. godine. Zadeh je fazi skupove koristio kao sredstvo za predstaljanje neprecizno definisanih pojmova, pri čemu ova nepreciznost proističe iz činjenice da za posmatrane pojmove ne postoje strogo odredjenje granice pripadanja nekoj klasi. Kako se prilikom rada sa klasama objekata u realnom, fizičkom svetu, često srećemo sa ovakvim pojmovima, fazi skupovi imaju široku primenu. U istom radu u kome je uveo pojam fazi skupa, Zadeh je uveo i pojam fazi relacije, a nakon toga i pojam fazi ekvivalencije i fazi kvazi-uređenja. Kako fazi relacije pružaju veliku slobodu izražavanja veza između elemenata, koje se ne mogu lako uočiti, često su korišćene kao modeli različitih pojmova koji se javljaju u takozvanim "mekim" naukama kao što su psihologija, lingvistika i u mnogim drugim naučnim oblastima.

Sa pojavom prvih mašinskih jezika, javlja se potreba da se prevaziđe razlika između preciznosti programskog jezika i nepreciznosti prirodnog jezika. U tu svrhu, u teoriji fazi skupova uveden je pojam fazi automata. Tokom posladnjih decenija fazi automati i jezici su korišćeni u raznim oblastima, uključujući leksičku analizu, kontrolne sisteme, klinički monitoring, opis prirodnih i mašinskih jezika, baze podataka, diskretne sisteme događaja, prepoznavanje uzoraka, kao i mnoge druge oblasti.

Jedan od glavnih problema teorije automata jeste da odredi da li su dva automata ekvivalentna, što obično znači da odredi da li se dva automata ponašaju na identičan način. U kontekstu determinističkih, nedeterminističkih i fazi automata ponašanje automata predstavlja jezik ili fazi jezik koji je raspoznat automatom i prema tome, dva automata su ekvivalentna, ili preciznije, jezički ekvivalentna ako raspoznaju isti jezik. U slučaju konačnih determinističkih automata problem ekvivalencije je rešiv u polinomijalnom vremenu, ali u slučaju nedeterminističkih i fazi automata ovaj problem je NP-kompletan. Sledeći važan zadatak teorija automata jeste da predstavi jezičku ekvivalenciju dva automata kao relaciju između stanja ovih automata, ukoliko takva relacija postoji, odnosno da nađe relaciju koja bi aproksimirala jezičku ekvivalenciju. Jezička ekvivalencija dva deterministička automata može biti predstavljena u terminima relacije između njihovih stanja, dok je u slučaju nedeteminističkih i fazi automata ovaj problem dosta kompleksniji i mozemo razviti samo aproksimacije jezičke ekvivalencije.

Najčešće proučavan koncept za modeliranje "ekvivalencije" između stanja automata je bisimulacija. Bisimulacije su teoriju računarstva uvedene od strane Milnera[81] i Parka[89], a u istom periodu bisimulacije su otkrivene i u nekim oblastima matematike kao što su modalna logika i teorija skupova. Danas su bisimulacije rasprostranjene u različitim oblastima računarstva, kao što su funkcionalni jezici, objektno-orjentisani jezici, tipovi podataka, optimizacija kompajlera, baze podataka, programska analiza, verifikacioni alati itd.

Glavni zadatak ove doktorske disertacije je proučavanje bisimulacija za fazi automate, sa specijalnim osvrtom na nalaženje najvećih bisimulacija na odgovarajućim sistemima kao i uopštenje pojma bisimulacije da bi se dobila relacija koja bolje aproksimira jezičku ekvivalenciju ili daje bolje rezultate pri redukciji fazi automata.

U drugom odeljku disertacije prikazani su fundamentalni pojmovi kao i poznati rezultati teorije fazi automata. Osim toga, razmatrani su krisp-deterministički fazi automati. Kao najvažniji predstavnik klase krisp-determinističkih automata, predstavili smo Nerodov automat. Ovaj automat se dobija iz datog fazi automata metodom koji predstavlja uopštenje klasične podskup konstrukcije na fazi slučaj [55, 56]. Zatim smo razmatrali faktor fazi automate u odnosu na fazi ekvivalencije na skupu stanja tih automata, kao i faktor fazi automate u odnosu na fazi kvazi-uređenja, zvane afterset i foreset automati. Konačno, predstavljena su dva tipa simulacija između fazi automata, forward i backward simulacije i uzimajući u obzir četiri slučaja kada je fazi relacija i njen inverz forward i backward simulacija dobili smo četiri tipa bisimulacija: forward, backward, forward-backward i backward-forward bisimulacije. Ove koncepte uveli su Ćirić i saradnici u [22], gde su predstavljeni značajni rezultati u proučavanju bisimulacija fazi automata nad kompletnim reziduiranim mrežama.

U trećoj glavi je za svaki od napred navedenih tipova simulacija/bisimulacija predstavljen efikasan algoritam za određivanje da li postoji simulacija/bisimulacija tog tipa između datih fazi automata, kao i za računanje najvećih, ukoliko postoje. Algoritam se bazira na metodi razvijenoj u [55], koja računa najveću fiksnu tačku izotone funkcije u mreži fazi relacija, koja je sadržana u datoj fazi relaciji. Na kraju odeljka, rad algoritma je ilustrovan sa nekoliko primera. Ovi primeri takođe pokazuju da nijedan tip bisimulacije nema značajnu prednost u odnosu na ostale. Drugim rečima, ovi primeri pokazuju da može da postoji bisimulacija jednog tipa između fazi automata, a da ne postoji bisimulacija ni jednog od tri ostala tipa.

U odeljku četiri uvodimo dve nove bisimulacije za fazi auotmate, slabu forward i slabu backward bisimulaciju. Slabe forward i backward bisimulacije daju bolju aproksimaciju jezičke ekvivalencije nego forward i backward bisimulacije i kada se koriste u redukciji stanja fazi automata daju bolju redukciju. Procedura za određivanje da li postoji slaba forward i backward bisimulacija kao i procedura za računanje najveće kad god takva bisimulacija postoji, predstavljene su u ovoj glavi. Zatim smo razmatrali slabe bisimulacije u kontekstu unfromnih fazi relacija, takozvane slabe uniformne bisimulacije. Data je, takođe, karakterizacija slabih uniformnih backward i forward bisimulacija između fazi automata u terminima izomorfizama između Nerodovih i reverznih Nerodovih automata.

Sekcija pet posvećena je problemu redukcije stanja fazi automata. Desno i levo invarijantna fazi kvazi-uređenja i fazi ekvivalencije, uvedene u [116], pokazale su se kao veom dobro sredstvo za redukciju broja stanja fazi automata. U [116], je, takođe, pokazano da veća desno (levo) invarijantna fazi kvazi-uređenja (ekvivalencije) daju bolju redukciju. Stoga je ovde razvijen novi algoritam za računanje najvećeg desno (levo) invarijantnog fazi kvazi-uređenja (ekvivalencije), zasnovan na čuvenom Paige-Tarjanovom problemu [87]. Kompleksnost ovog algoritma je $O(n^5m)$, gde je n broj stanja automata $\mathscr{A}$, a m veličina alfabeta. Osim toga data je i modifikovana verzija ovog algoritma koja računa najveću desno-invarijantnu ekvivalenciju na datom nedeterminističkom automatu. Modifikovana verzija algrotima radi u vremenu $O(n^3m)$.

Daćemo u kratkim crtama pregled ključnih rezultata predstavljenih u urađenoj doktorskoj disertaciji.

Na kraju bih želela da se zahvalim svom mentoru Profesorki Jeleni Ignjatović, za srdačnu pomoć i prijateljske savete tokom izrade doktorske disertacije. Pored toga bih želela da zahvalim Profesoru Miroslavu Ćiriću, za stalnu inspiraciju i motivaciju za bavljenje naučnim radom, kao i Ivanu Stankoviću za velikodušnu pomoć u izradi programa za navedene algoritme. Takođe, se zahvaljujem i svojoj porodici na nesebičnoj podršci i razumevanju koje mi je pružala tokom izrade doktorske teze.

1. Osnovni pojmovi

Reziduirana mreža je algebra $\mathscr{L} = (L, \wedge, \vee, \otimes, \to, 0, 1)$ takva da

(L1) $(L, \wedge, \vee, 0, 1)$ je mreža sa najmanjim elementom 0 i najvećim 1,

(L2) $(L, \otimes, 1)$ je komutativni monoid sa jedinicom 1,

(L3) $\otimes$ i $\to$ formiraju *adjungovani par* , zadovoljavaju *svojstvo adjungcije*: za sve $x, y, z \in L$,

$$x \otimes y \leqslant z \iff x \leqslant y \to z. \tag{B.1}$$

Ako je osim toga $(L, \wedge, \vee, 0, 1)$ kompletna mreža, tada $\mathscr{L}$ nazivamo *kompletna reziduirana mreža* .

Operacija $\otimes$ (zvana *multiplikacija*) i $\to$ (zvana *reziduum*) namenjene su za modeliranje konjunkcije i implikacije odgovarajućih logičih računa, dok su supremum ($\bigvee$) i infimum ($\bigwedge$) namenjeni za modeliranje ekstenzionalnog i univerzalnog kvantifikatora, respektivno. Operacija $\leftrightarrow$ definisana sa

$$x \leftrightarrow y = (x \to y) \wedge (y \to x), \tag{B.2}$$

naziva se *bireziduum* (ili *biimplikacija*), i koristi se za modeliranje ekvivalencije istinitosnih vrednosti.

Osnovna svojstva kompletnih reziduranih mreža mogu se naći u [3, 10].

Najčešće korišćene strukture istinitosnih vrednosti, koje su najviše proučavane i definisane na realnom jediničnom intervalu $[0,1]$ sa $x \wedge y = \min(x,y)$ i $x \vee y = \max(x,y)$, su *Łukasiewiczeva struktura* ($x \otimes y = \max(x+y-1,0)$, $x \to y = \min(1-x+y,1)$), *Goguenova* (*proizvod*) *struktura* ($x \otimes y = x \cdot y$, $x \to y = 1$ if $x \leqslant y$ i $= y/x$ inače) i *Gödelova struktura* ($x \otimes y = \min(x,y)$, $x \to y = 1$ ako $x \leqslant y$ i $= y$ inače). Sledeći važan skup istinitosnih vrednosti je skup $\{a_0, a_1, \dots, a_n\}$, $0 = a_0 < \dots < a_n = 1$, sa $a_k \otimes a_l = a_{\max(k+l-n,0)}$ i $a_k \to a_l = a_{\min(n-k+l,n)}$. Specijalan slučaj ove algebre je dvoelementna Bulova algebra klasične logike, čiji je nosač skup $\{0,1\}$. Jedini adjungovani par operacija u dvoelementnoj Bulovoj algebri su klasična konjukcija i implikacija. Ova struktura istinitosnih vrednosti se naziva *Bulova struktra*.

Parcijalno uređen skup P zadovoljava *uslov opadajućih lanaca* (kraće *UOL*) ako se svaki opadajući niz elemenata u P završava nakon konačno mnogo koraka, tj. ako za svaki opadajući niz $\{a_k\}_{k\in\mathbb{N}}$ elemenata u P postoji $k \in \mathbb{N}$ tako da $a_k = a_{k+l}$, za sve $l \in \mathbb{N}$. Drugim rečima, P zadovoljava UOL ako ne postoji beskonačan opadajući lanac u P.

U nastavku će $\mathscr{L}$ biti kompletna reziduirana mreža. *Fazi podskup* skupa *A nad* $\mathscr{L}$, ili samo *fazi podskup* od A, je svako preslikavanje iz A u L. Pod običnim podskupom skupa A podrazumevamo fazi poskup od A koji uzima vrednosti u skupu $\{0,1\} \subseteq L$. Neka su f i g dva fazi podskupa od A. *Jednakost* fazi poskupova f i g definiše se obično kao jednakost preslikavanja, tj. $f = g$ ako i samo ako $f(x) = g(x)$, za svaki $x \in A$. *Inkluzija* $f \leqslant g$ takođe se definiše kao jednakost preslikavanja: $f \leqslant g$ ako i samo ako $f(x) \leqslant g(x)$, za svaki $x \in A$. Zajedno sa ovim parcijalnim uređenjem skup L^A svih fazi podskupova od A formira kompletnu reziduiranu mrežu, u kojoj su presek $\bigwedge_{i\in I} f_i$ i unija $\bigvee_{i\in I} f_i$ proizvoljne familije $\{f_i\}_{i\in I}$ fazi podskupova od A preslikavanja iz A u L definisana sa

$$\left(\bigwedge_{i\in I} f_i\right)(x) = \bigwedge_{i\in I} f_i(x), \qquad \left(\bigvee_{i\in I} f_i\right)(x) = \bigvee_{i\in I} f_i(x),$$

i *proizvod* $f \otimes g$ je fazi poskup definisan sa: $f \otimes g(x) = f(x) \otimes g(x)$, za svaki $x \in A$.

Fazi relacija između skupova A i B (ovim redom) je svako preslikavanje iz $A \times B$ u L, tj. svaki fazi podskup od $A \times B$, i jednakost, inkluzija (uređenje), unija i presek fazi relacija definisani su kao u slučaju fazi skupova. Skup svih fazi relacija između A i B označićemo sa $L^{A\times B}$. Specijalno, fazi relacija na skupu A je svaka funkcija iz $A \times A$ u L, tj. fazi poskup od $A \times A$. Skup svih fazi relacija na A biće označen sa $L^{A\times A}$. *Reverz* ili *inverz* fazi relacije $\alpha \in L^{A\times B}$ je fazi relacija $\alpha^{-1} \in L^{B\times A}$ definisana sa $\alpha^{-1}(b,a) = \alpha(a,b)$, za svaki $a \in A$ i $b \in B$. Krisp relacija je fazi relacije koja uzima vrednosti samo u skupu $\{0,1\}$, i ako je α krisp relacija iz A u B, tada izrazi "$\alpha(a,b) = 1$" i "$(a,b) \in \alpha$" imaju isto značenje.

Za neprazne skupove A, B i C, i fazi relaciju $\alpha \in L^{A\times B}$ i $\beta \in L^{B\times C}$, njihova *kompozicija* $\alpha \circ \beta \in L^{A\times C}$ je fazi relacija definisana sa

$$(\alpha \circ \beta)(a,c) = \bigvee_{b\in B} \alpha(a,b) \otimes \beta(b,c), \tag{B.3}$$

za sve $a \in A$ i $c \in C$. Za $f \in L^A$, $\alpha \in L^{A\times B}$ i $g \in L^B$, kompozicije $f \circ \alpha \in L^B$ i $\alpha \circ g \in L^A$ su fazi skupovi definisani

$$(f \circ \alpha)(b) = \bigvee_{a\in A} f(a) \otimes \alpha(a,b), \qquad (\alpha \circ g)(a) = \bigvee_{b\in B} \alpha(a,b) \otimes g(b), \tag{B.4}$$

za svaki $a \in A$ i $b \in B$. Konačno, kompozicija fazi skupova $f, g \in L^A$ je skalar $f \circ g \in L$ definisan sa

$$f \circ g = \bigvee_{a\in A} f(a) \otimes g(a). \tag{B.5}$$

Lako je pokazati da je kompozicija fazi relacija asocijativna, tj.

$$(\alpha \circ \beta) \circ \gamma = \alpha \circ (\beta \circ \gamma), \tag{B.6}$$

za sve $\alpha \in L^{A\times B}$, $\beta \in L^{B\times C}$ i $\gamma \in L^{C\times D}$, i

$$(f \circ \alpha) \circ \beta = f \circ (\alpha \circ \beta), \quad (f \circ \alpha) \circ g = f \circ (\alpha \circ g), \quad (\alpha \circ \beta) \circ h = \alpha \circ (\beta \circ h) \tag{B.7}$$

za sve $\alpha \in L^{A\times B}$, $\beta \in L^{B\times C}$, $f \in L^A$, $g \in L^B$ i $h \in L^C$. Stoga, sve zagrade u (B.6) i (B.7) mogu biti izostavljene.

Fazi relacija φ na skupu A naziva se *refleksivna*, ako $\varphi(a,a) = 1$, *simetrična*, ako $\varphi(a,b) = \varphi(b,a)$, i *transitivna*, ako $\varphi(a,b) \otimes \varphi(b,c) \leqslant \varphi(a,c)$, za sve $a,b,c \in A$. Refleksivna i tranzitivna fazi relacija naziva se *fazi kvazi-uređenje*. Simetrično fazi kvazi uređenje je *fazi ekivalencija*. Za kvazi-uređenje φ na A i element $a \in A$, *φ-afterset* od a je fazi skup $a\varphi \in L^A$ definisan sa $a\varphi(b) = \varphi(a,b)$, i *$\varphi$-foreset* od a je fazi skup $\varphi a \in L^A$ definisan sa $\varphi a(b) = \varphi(b,a)$, za sve $b \in A$. Ako je φ fazi ekvivalencija, tada se *φ-afterset* od a poklapa sa *φ-foresetom* od a, i naziva se *fazi klasa ekvivalencije* od a.

2. Fazi automati

U daljem tekstu, $\mathscr{L}$ će biti kompletna reziduirana mreža i X će biti (konačan) alfabet.
Fazi automat nad $\mathscr{L}$ i X (*fazi automat*), je uređena četvorka $\mathscr{A} = (A, \delta, \sigma, \tau)$, gde:

- A je neprazan skup, *skup stanja*;
- $\delta : A \times X \times A \to L$ je fazi poskup od $A \times X \times A$, zvan *fazi tranziciona funkcija*;
- $\sigma : A \to L$ je fazi podskup od A, zvan *fazi skup inicijalnih stanja*;
- $\tau : A \to L$ je fazi podskup od A, zvan *fazi skup završnih stanja*.

Simbol $\delta(a,x,b)$ možemo interpretirati kao stepen prelaza iz stanja $a \in A$ u stanje $b \in A$ pod uticajem ulaznog slova $x \in X$, dok oznake $\sigma(a)$ i $\tau(a)$ možemo interpretirati kao stepen pripadnosti stanja a skupu inicijalnih odnosno završnih stanja. Iz metodoloških razloga ćemo dozvoliti da skup stanja A bude beskonačan. Fazi automat čiji je skup stanja konačan naziva se *konačan fazi automat*.

Reverzni fazi automat automata $\mathscr{A} = (A,\delta,\sigma,\tau)$ je definisan kao fazi automat $\bar{\mathscr{A}} = (A,\bar{\delta},\bar{\sigma},\bar{\tau})$ čija je fazi funkcija prelaza data sa:

$$\bar{\delta}(a_1,x,a_2) = \delta(a_2,x,a_1) \quad \text{za sve } a_1,a_2 \in A, \quad x \in X,$$

dok su fazi inicijalna i završna stanja zamenila svoja mesta $\bar{\sigma} = \tau$ i $\bar{\tau} = \sigma$.

Fazi automat $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ i $\mathscr{A}' = (A',\delta^{A'},\sigma^{A'},\tau^{A'})$ su *izomorfni* ako postoji bijekcija $\phi : A \to A'$ takva da $\delta_x^A(a,b) = \delta_x^{A'}(\phi(a),\phi(b))$, za sve $a,b \in A$ i $x \in X$, i takođe, $\sigma^A(a) = \sigma^{A'}(\phi(a))$ i $\tau^A(a) = \tau^{A'}(\phi(a))$, za svaki $a \in A$.

Fazi jezik u X^* nad $\mathscr{L}$, ili samo *fazi jezik*, je svaki podskup skupa X^*, tj. svaka funkcija iz X^* u L. A *fazi jezik raspoznat fazi automatom* $\mathscr{A} = (A,\delta,\sigma,\tau)$, u oznaci $[\![\mathscr{A}]\!]$, je fazi jezik $\mathscr{F}(X^*)$ definisan sa

$$\begin{aligned} [\![\mathscr{A}]\!](e) &= \sigma^A \circ \tau^A, \\ [\![\mathscr{A}]\!](u) &= \sigma^A \circ \delta_{x_1}^A \circ \delta_{x_2}^A \circ \cdots \circ \delta_{x_n}^A \circ \tau^A, \end{aligned} \tag{B.8}$$

za sve $u = x_1x_2\ldots x_n \in X^+$, gde je $x_1,x_2,\ldots,x_n \in X$.

Fazi automati $\mathscr{A}$ i $\mathscr{B}$ su *jezički ekvivalentni*, ili samo *ekvivalentni*, ako $[\![\mathscr{A}]\!] = [\![\mathscr{B}]\!]$.

Za dati fazi jezik $\varphi : X^* \to L$ i $v \in X^*$, definišemo $v^{-1}\varphi : X^* \to L$ i $\varphi v^{-1} : X^* \to L$ sa $v^{-1}\varphi(u) = \varphi(vu)$ i $\varphi v^{-1}(u) = \varphi(uv)$ za $u \in X^*$. Fazi jezik $v^{-1}\varphi$ naziva se *levi izvod*, a fazi jezik φv^{-1} *desni izvod* od φ u odnosu na v.

Kardinalnost fazi automata $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$, u oznaci $|\mathscr{A}|$, se definiše kao kardinalnost skupa stanja automata $\mathscr{A}$. Fazi automat $\mathscr{A}$ je *minimalni fazi automat* jezika $f \in \mathscr{F}(X^*)$ ako raspoznaje jezik f i $|\mathscr{A}| < |\mathscr{A}'|$, za svaki automat $\mathscr{A}'$ koji raspoznaje f. Minimalni fazi automat koji raspoznaje dati fazi jezik f ne mora obavezno biti jedinstven do na izomorfizam. Ovo takođe važi i u slučaju nedeterminističkih automata.

Neka je $\mathscr{A} = (A,\delta,\sigma,\tau)$ fazi automat nad X i $\mathscr{L}$. Fazi tranziciona funkcija δ je *krisp-deterministička* ako za svaki $x \in X$ i svaki $a \in A$ postoje $a' \in A$ takvi da $\delta_x(a,a') = 1$, i $\delta_x(a,b) = 0$, za svaki $b \in A \setminus \{a'\}$. Fazi skup inicijalnih stanja σ zove se *krisp deterministički* ako postoji $a_0 \in A$ takvo da $\sigma(a_0) = 1$ i $\sigma(a) = 0$ za svaki $a \in A \setminus \{a_0\}$. Ako su i inicijalno stanje σ i δ krisp-deterministički, tada se $\mathscr{A}$ naziva *konačan krisp-deterministički automat* (kraće: *kkda*).

Nerodov automat fazi automata $\mathscr{A} = (A,\delta,\sigma,\tau)$ je krisp-determinističi fazi automat $\mathscr{A}_N = (A_N,\delta_N,\sigma_e^A,\tau_N)$, takav da $A_N = \{\sigma_u^A | u \in X^*\}$ gde je $\sigma_u^A = \sigma^A \circ \delta_u^A$,za svaki $u \in X^*$ i $\delta_N : A_N \times X \longrightarrow A_N$ i $\tau_N \in \mathscr{F}(A_N)$ su definisani sa

$$\delta_N(\sigma_u^A,x) = \sigma_{ux}^A, \qquad \tau_N(\sigma_u^A) = \sigma_u^A \circ \tau^A,$$

za svaki $u \in X^*$ i $x \in X$. Automat $\mathscr{A}_N$ je jezički ekvivalentan automatu $\mathscr{A}$.

Neka je $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ fazi automat i neka je E fazi ekvivalencija na A. *Faktor fazi automat* od $\mathscr{A}$ u odnosu na E je fazi automat $\mathscr{A}/E = (A/E, \delta^{A/E}, \sigma^{A/E}, \tau^{A/E})$, gde je: skup stanja $A/E = \{E_a | \, a \in A\}$, fazi tranziciona relacija $\delta^{A/E} : A/E \times X \times A/E \to L$ definisana sa:

$$\delta^{A/E}(E_a, x, E_b) = \bigvee_{a',b' \in A} E(a,a') \otimes \delta^A(a',x,b') \otimes E(b',b) = E_a \circ \delta^A_x \circ E_b,$$

za svaki $E_a, E_b \in A/E$, i $x \in X$, fazi skup $\sigma^{A/E} \in \mathscr{F}(A)$ inicijalnih stanja se definiše sa:

$$\sigma^{A/E}(E_a) = \sigma^A \circ E_a, \text{ za svaki } E_a \in A/E,$$

i fazi skup $\tau^{A/E} \in \mathscr{F}(A)$ finalnih stanja sa:

$$\tau^{A/E}(E_a) = E_a \circ \tau^A, \text{ za svaki } E_a \in A/E.$$

Ukoliko u gornjoj definciji umseto ekvivalencije E posmatramo kvazi-uređenje dobijamo definiciju afterset automata.

Fazi jezik $[\![\mathscr{A}/\mathscr{E}[\![$ raspoznat faktor fazi automatom $\mathscr{A}/\mathscr{E}$ je dat sa

$$[\![\mathscr{A}/E]\!](e) = \sigma^A \circ E \circ \tau^A, \tag{B.9}$$

$$[\![\mathscr{A}/E]\!](u) = \sigma \circ E \circ \delta_{x_1} \circ E \circ \delta_{x_2} \circ E \cdots \circ E \circ \delta_{x_n} \circ E \circ \tau, \tag{B.10}$$

za $u = x_1 x_2 \dots x_n \in X^+$, gde je $x_1, x_2, \dots, x_n \in X$.

Neka su $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ i $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ fazi automati, i neka je $\gamma \in \mathscr{R}(A,B)$ neprazna fazi relacija. Relacija γ zove se *forward simulacija* ako zadovoljava

$$\sigma^A \leqslant \sigma^B \circ \gamma^{-1}, \tag{fs-1}$$

$$\gamma^{-1} \circ \delta^A_x \leqslant \delta^B_x \circ \gamma^{-1}, \quad \text{za svaki } x \in X, \tag{fs-2}$$

$$\gamma^{-1} \circ \tau^A \leqslant \tau^B, \tag{fs-3}$$

i *backward simulacija* ako

$$\tau^A \leqslant \gamma \circ \tau^B, \tag{bs-1}$$

$$\delta^A_x \circ \gamma \leqslant \gamma \circ \delta^B_x, \quad \text{za svaki } x \in X, \tag{bs-2}$$

$$\sigma^A \circ \gamma \leqslant \sigma^B. \tag{bs-3}$$

Dalje, relaciju γ nazivamo *forward bisimulacija* ako su obe relacije γ i γ^{-1} forward simulacije, tj. ako γ zadovoljava

$$\sigma^A \leqslant \sigma^B \circ \gamma^{-1}, \qquad \sigma^B \leqslant \sigma^A \circ \gamma, \tag{fb-1}$$

$$\gamma^{-1} \circ \delta_x^A \leqslant \delta_x^B \circ \gamma^{-1}, \quad \gamma \circ \delta_x^B \leqslant \delta_x^A \circ \gamma, \quad \text{za svaki } x \in X, \tag{fb-2}$$

$$\gamma^{-1} \circ \tau^A \leqslant \tau^B, \qquad \gamma \circ \tau^B \leqslant \tau^A, \tag{fb-3}$$

i *backward bisimulacija*, ako su obe relacije γ i γ^{-1} backward simulacije, tj. ako γ zadovoljava

$$\tau^A \leqslant \gamma \circ \tau^B, \qquad \tau^B \leqslant \gamma^{-1} \circ \tau^A, \tag{bb-1}$$

$$\delta_x^A \circ \gamma \leqslant \gamma \circ \delta_x^B, \quad \delta_x^B \circ \gamma^{-1} \leqslant \gamma^{-1} \circ \delta_x^A, \quad \text{za svaki } x \in X, \tag{bb-2}$$

$$\sigma^A \circ \gamma \leqslant \sigma^B, \qquad \sigma^B \circ \gamma^{-1} \leqslant \sigma^A. \tag{bb-3}$$

Takođe, ako je γ forward simulacija i γ^{-1} je backward simulacija, tj. ako γ zadovoljava

$$\sigma^A \leqslant \sigma^B \circ \gamma^{-1}, \qquad \tau^B \leqslant \gamma^{-1} \circ \tau^A, \tag{fbb-1}$$

$$\gamma^{-1} \circ \delta_x^A = \delta_x^B \circ \gamma^{-1}, \qquad \text{za svaki } x \in X, \tag{fbb-2}$$

$$\sigma^B \circ \gamma^{-1} \leqslant \sigma^A, \qquad \gamma^{-1} \circ \tau^A \leqslant \tau^B, \tag{fbb-3}$$

tada se γ naziva *forward-backward bisimulacija*, i ako je γ backward simulacija i γ^{-1} forward simulacija, tj. ako

$$\sigma^B \leqslant \sigma^A \circ \gamma, \qquad \tau^A \leqslant \gamma \circ \tau^B, \tag{bfb-1}$$

$$\delta_x^A \circ \gamma = \gamma \circ \delta_x^B, \qquad \text{za svaki } x \in X, \tag{bfb-2}$$

$$\sigma^A \circ \gamma \leqslant \sigma^B \qquad \gamma \circ \tau^B \leqslant \tau^A. \tag{bfb-3}$$

tada se γ naziva *backward-forward bisimulacija*.

3. Računanje najvećih simulacija i bisimulacija između fazi automata

Kao što smo naveli u uvodu, problem odlučivanja da li postoji simulacija odnosno bisimulacija određenog tipa između datih fazi automata, kao i problem izračunavanja najveće simulacije odnosno bisimulacije tog tipa, svodi se na problem računanja najveće post-fiksne tačke, sadržane u datoj fazi relaciji odgovarajuće izotone funkcije na mreži fazi relacija. U tu svrhu uvodimo sledeće oznake kao i fazi relacije i funkcije na mreži fazi relacija.

Za neprazne skupove A i B i fazi podskupove $\eta \in \mathscr{F}(A)$ i $\xi \in \mathscr{F}(B)$, fazi relacije $\eta \backslash \xi \in \mathscr{R}(A,B)$ i $\eta / \xi \in \mathscr{R}(A,B)$ su definisane na sledeći način:

$$(\eta\backslash\xi)(a,b) = (\,\eta(a) \to \xi(b)\,), \tag{B.11}$$

$$(\eta/\xi)(a,b) = (\,\xi(b) \to \eta(a)\,), \tag{B.12}$$

za proizvoljne $a \in A$ i $b \in B$. Uvedimo oznaku $\eta/\xi = (\xi\backslash\eta)^{-1}$.

Lemma B.1. *Neka su A i B neprazni skupovi i neka su $\eta \in \mathscr{F}(A)$ i $\xi \in \mathscr{F}(B)$.*

(a)*Skup svih rešenja nejednačine $\eta \circ \chi \leqslant \xi$, gde je χ nepoznata fazi relacija između A i B, je glavni ideal mreže $\mathscr{R}(A,B)$ generisan fazi relacijom $\eta\backslash\xi$.*
(b)*Skup svih rešenja nejednačine $\chi \circ \xi \leqslant \eta$, gde je χ nepoznata fazi relacija između A i B, je glavni ideal mreže $\mathscr{R}(A,B)$ generisan fazi relacijom η/ξ.*

Primetimo da je $(\eta\backslash\xi) \wedge (\eta/\xi) = \eta|\xi$, gde je $\eta|\xi$ fazi relacija između A i B definisana sa

$$(\eta|\xi)(a,b) = (\,\eta(a) \leftrightarrow \xi(b)\,), \tag{B.13}$$

za proizvoljne $a \in A$ i $b \in B$.

Dalje, neka su A i B neprazni skupovi i neka $\alpha \in \mathscr{R}(A)$, $\beta \in \mathscr{R}(B)$ i $\gamma \in \mathscr{R}(A,B)$. *Desni rezidual* od γ sa α je fazi relacija $\alpha\backslash\gamma \in \mathscr{R}(A,B)$ definisana sa

$$(\alpha\backslash\gamma)(a,b) = \bigwedge_{a'\in A} (\,\alpha(a',a) \to \gamma(a',b)\,), \tag{B.14}$$

za sve $a \in A$ i $b \in B$, i *levi rezidual* od γ sa β je fazi relacija $\gamma/\beta \in \mathscr{R}(A,B)$ definisana sa

$$(\gamma/\beta)(a,b) = \bigwedge_{b'\in B} (\,\beta(b,b') \to \gamma(a,b')\,), \tag{B.15}$$

za sve $a \in A$ i $b \in B$.

Lema 2.1. *Neka su A i B neprazni skupovi i neka $\alpha \in \mathscr{R}(A)$, $\beta \in \mathscr{R}(B)$ i $\gamma \in \mathscr{R}(A,B)$.*

(a)*Skup svih rešenja nejednačine $\alpha \circ \chi \leqslant \gamma$, gde je χ nepoznata fazi relacija između A i B, je glavni ideal mreže $\mathscr{R}(A,B)$ generisan desnim rezidualom $\alpha\backslash\gamma$ od γ sa α.*
(b)*Skup svih rešenja nejednačine $\chi \circ \beta \leqslant \gamma$, gde je χ nepoznata fazi relacija između A i B, je glavni ideal mreže $\mathscr{R}(A,B)$ generisan levim rezidualom γ/β od γ sa β.*

Neka su $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ i $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ fazi automati. Definišemo fazi relaciju $\pi^w \in \mathscr{R}(A,B)$, za $w \in \{fs,bs,fb,bb,fbb,bfb\}$, na sledeći način:

$$\pi^{fs} = \tau^A\backslash\tau^B, \tag{B.16}$$

$$\pi^{bs} = \sigma^A\backslash\sigma^B, \tag{B.17}$$

$$\pi^{fb} = (\tau^A\backslash\tau^B) \wedge (\tau^A/\tau^B) = \tau^A|\tau^B, \tag{B.18}$$

$$\pi^{bb} = (\sigma^A\backslash\sigma^B) \wedge (\sigma^A/\sigma^B) = \sigma^A|\sigma^B, \tag{B.19}$$

$$\pi^{fbb} = (\tau^A\backslash\tau^B) \wedge (\sigma^A/\sigma^B), \tag{B.20}$$

$$\pi^{bfb} = (\sigma^A\backslash\sigma^B) \wedge (\tau^A/\tau^B). \tag{B.21}$$

Takođe, definišemo funkcije $\phi^w : \mathscr{R}(A,B) \to \mathscr{R}(A,B)$, za $w \in \{fs, bs, fb, bb, fbb, bfb\}$, na sledeći način:

$$\phi^{fs}(\gamma) = \bigwedge_{x\in X} [(\delta_x^B \circ \gamma^{-1})/\delta_x^A]^{-1}, \tag{B.22}$$

$$\phi^{bs}(\gamma) = \bigwedge_{x\in X} \delta_x^A \backslash (\gamma \circ \delta_x^B), \tag{B.23}$$

$$\phi^{fb}(\gamma) = \bigwedge_{x\in X} [(\delta_x^B \circ \gamma^{-1})/\delta_x^A]^{-1} \wedge [(\delta_x^A \circ \gamma)/\delta_x^B] = \phi^{fs}(\gamma) \wedge [\phi^{fs}(\gamma^{-1})]^{-1}, \tag{B.24}$$

$$\phi^{bb}(\gamma) = \bigwedge_{x\in X} [\delta_x^A \backslash (\gamma \circ \delta_x^B)] \wedge [\delta_x^B \backslash (\gamma^{-1} \circ \delta_x^A)]^{-1} = \phi^{bs}(\gamma) \wedge [\phi^{bs}(\gamma)]^{-1}, \tag{B.25}$$

$$\phi^{fbb}(\gamma) = \bigwedge_{x\in X} [(\delta_x^B \circ \gamma^{-1})/\delta_x^A]^{-1} \wedge [\delta_x^B \backslash (\gamma^{-1} \circ \delta_x^A)]^{-1} = \phi^{fs}(\gamma) \wedge [\phi^{bs}(\gamma^{-1})]^{-1}, \tag{B.26}$$

$$\phi^{bfb}(\gamma) = \bigwedge_{x\in X} [\delta_x^A \backslash (\gamma \circ \delta_x^B)] \wedge [(\delta_x^A \circ \gamma)/\delta_x^B] = \phi^{bs}(\gamma) \wedge [\phi^{fs}(\gamma^{-1})]^{-1}, \tag{B.27}$$

za svaki $\gamma \in \mathscr{R}(A,B)$.

Naredna teorema daje ekvivalentne forme za drugi i treći uslov u definiciji simulacija i bisimulacija.

Teorema 2.1. *Neka su $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ i $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ fazi automati i neka je $w \in \{fs, bs, fb, bb, fbb, bfb\}$. Fazi relacija $\gamma \in \mathscr{R}(A,B)$ zadovoljava uslove* (w-2) *i* (w-3) *ako i samo ako važi:*

$$\gamma \leqslant \phi^w(\gamma), \qquad \gamma \leqslant \pi^w. \tag{B.28}$$

U nastavku predstavljamo metod za računanje najvećih simulacija i bisimulacija između fazi automata.

Neka su A i B neprazni skupovi i neka je $\phi : \mathscr{R}(A,B) \to \mathscr{R}(A,B)$ izotona funkcija, tj. neka $\alpha \leqslant \beta$ povlači $\phi(\alpha) \leqslant \phi(\beta)$, za sve $\alpha, \beta \in \mathscr{R}(A,B)$. Fazi relacija $\alpha \in \mathscr{R}(A,B)$ zove se *post-fiksna tačka* od ϕ ako $\alpha \leqslant \phi(\alpha)$. Dobro poznata Knaster-Tarski theorema o fiksnoj tački (data i dokazana u opštijem obliku, za kompletne mreže) tvrdi da skup svih post-fiksnih tačaka od ϕ formira kompletnu mrežu (cf. [104]). Pored toga, za svaku fazi relaciju $\pi \in \mathscr{R}(A,B)$ imamo da je skup svi post-fiksnih tačaka od ϕ koje su sadržane u π takođe kompletna mreža. Prema Theoremi B.28, naš glavni zadatak je da nađemo efektivnu proceduru za računanje najveće post-fiksne tačke funkcije ϕ^w koja je sadržana u fazi relaciji π^w, za svaki $w \in \{fs, bs, fb, bb, fbf, bfb\}$.

Treba naglasiti da je skup svih post-fiksnih tačaka izotone funkcije na kompletnoj mreži uvek neprazan, jer sadrži barem najmanji element kompletne mreže. Međutim, ovaj skup može sadržati samo jedan element. U našem slučaju, kada se radi sa mrežom fazi relacija, prazna relacija može biti jedina post-fiksna tačka, dok sa druge strane simulacija i bisimulacija po definiciji moraju biti neprazne relacije. Ovaj zahtev je neophodan, jer prazna relacija ne može da zadovolji uslov (w-1), osim ako fazi skup inicijanih stanja ili fazi skup završnih stanja nije prazan. Stoga je naš zadatak zapravo da nađemo efektivnu proceduru za određivanje da li postoji neprazna post-fiksna tačka od ϕ^w sadržana u π^w, i ukoliko postoji, da se nađe najveća.

Neka je $\phi : \mathscr{R}(A,B) \to \mathscr{R}(A,B)$ izotona funkcija $\pi \in \mathscr{R}(A,B)$. Definišemo niz $\{\gamma_k\}_{k\in\mathbb{N}}$ fazi relacija na $\mathscr{R}(A,B)$ sa

$$\gamma_1 = \pi, \qquad \gamma_{k+1} = \gamma_k \wedge \phi(\gamma_k), \text{ for each } k \in \mathbb{N}. \tag{B.29}$$

Sekvenca $\{\gamma_k\}_{k\in\mathbb{N}}$ je očigledno opadajuća. Ako mi označimo sa $\hat{\gamma}$ najveću post-fiksnu tačku ϕ sadržanu u π, lako možemo pokazati da važi:

$$\hat{\gamma} \leqslant \bigwedge_{k\in\mathbb{N}} \gamma_k. \tag{B.30}$$

Sada dolazimo do dva veoma važna pitanja. Prvo, pod kojim uslovima važi jednakost u (B.30)? Drugo, pod kojim uslovima je niz $\{\gamma_k\}_{k\in\mathbb{N}}$ konačan? Ako je ovaj niz konačan, nije teško pokazati da postoji $k \in \mathbb{N}$ takvo da $\gamma_k = \gamma_m$, za svaki $m \geqslant k$, tj. postoji $k \in \mathbb{N}$ takvo da se niz stabilizuje u γ_k. Niz će se stabilizovati (zaustaviti) kada nađemo najmanji $k \in \mathbb{N}$ takav da $\gamma_k = \gamma_{k+1}$. U tom slučaju $\hat{\gamma} = \gamma_k$, i imamo algoritam koji računa $\hat{\gamma}$ u konačnom broju koraka.

Zapazimo da je niz $\{\gamma_k\}_{k\in\mathbb{N}}$ fazi relacija iz $\mathscr{R}(A,B)$ konačan ako i samo ako je *image-konačan*, što zapravo znači da je skup $\bigcup_{k\in\mathbb{N}} \mathrm{Im}(\gamma_k)$. Dalje, funkcija $\phi : \mathscr{R}(A,B) \to \mathscr{R}(A,B)$ se zove *image-lokalizovana* ako postoji konačan $K \subseteq L$ takav da za svaku fazi relaciju $\gamma \in \mathscr{R}(A,B)$ imamo

$$\mathrm{Im}(\phi(\gamma)) \subseteq \langle K \cup \mathrm{Im}(\gamma)\rangle, \tag{B.31}$$

gde $\langle K \cup \mathrm{Im}(\gamma)\rangle$ označava podalgebru od $\mathscr{L}$ generisanu sa $K \cup \mathrm{Im}(\gamma)$. Takvo K će se zvati *lokalizovan skup* funkcije ϕ.

Teorema 2.2. *Neka je ϕ image-lokalizovana funkcija, skup K lokalizovan skup, $\pi \in \mathscr{R}(A,B)$, i neka je $\{\gamma_k\}_{k\in\mathbb{N}}$ niz relacija u $\mathscr{R}(A,B)$ definisan sa (B.29). Tada*

$$\bigcup_{k\in\mathbb{N}} \mathrm{Im}(\gamma_k) \subseteq \langle K \cup \mathrm{Im}(\pi)\rangle. \tag{B.32}$$

Ako je pored toga $\langle K \cup \mathrm{Im}(\pi)\rangle$ konačna podalgebra od $\mathscr{L}$, tada je niz $\{\gamma_k\}_{k\in\mathbb{N}}$ konačan.

Vratimo se sada na ϕ^w, za $w \in \{fs, bs, fb, bb, fbb, bfb\}$, i predtavimo sledeći rezultat.

Teorema 2.3. *Neka su $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ i $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ proizvoljni fazi automati.*

Za proizvoljno $w \in \{fs, bs, fb, bb, fbb, bfb\}$ funkcija ϕ^w je izotona i slika-konačna.

U nastavku sledi glavni rezultat ovog odeljka, koji obezbeđuje algoritam za određivanje da li postoji simulacija ili bisimulacija određenog tipa između fazi automata i za računanje najveće simulacije i bisimulacije kada takva postoji.

Teorema 2.4. *Neka su $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ i $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ fazi automati, neka je $w \in \{fs, bs, fb, bb, fbb, bfb\}$, i neka je sekvenca $\{\gamma_k\}_{k\in\mathbb{N}}$ fazi relacija iz $\mathscr{R}(A,B)$ definisana sa*

$$\gamma_1 = \pi^w, \qquad \gamma_{k+1} = \gamma_k \wedge \phi^w(\gamma_k), \ \textit{za svaki}\ k \in \mathbb{N}. \tag{B.33}$$

Ako $\langle \mathrm{Im}(\pi^w) \cup \bigcup_{x\in X}(\mathrm{Im}(\delta_x^A) \cup \mathrm{Im}(\delta_x^B))\rangle$ je konačna podalgebra od $\mathscr{L}$, tada važi sledeće:

(a) *niz $\{\gamma_k\}_{k\in\mathbb{N}}$ je konačan i opadajući, i postoji bar jedan prirodan broj k takav da $\gamma_k = \gamma_{k+1}$;*
(b) *γ_k je najveća fazi relacija $\mathscr{R}(A,B)$ koja zadovoljava (w-2) i (w-3);*
(c) *ako γ_k zadovoljava (w-1), tada je γ_k najveća fazi relacija $\mathscr{R}(A,B)$ koja zadovoljava (w-1), (w-2) i (w-3);*
(d) *ako γ_k ne zadovoljava (w-1), tada ne postoji ni jedna fazi relacija u $\mathscr{R}(A,B)$ koja zadovoljava (w-1), (w-2) i (w-3).*

4. Slabe bisimulacije na fazi automatima

Neka je $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ fazi automat. Za svaki $u \in X^*$, definišemo fazi skupove $\sigma_u^A, \tau_u^A \in \mathscr{F}(A)$ na sledeći način:

$$\sigma_u^A = \sigma^A \circ \delta_u^A, \qquad \tau_u^A = \delta_u^A \circ \tau^A.$$

Treba naglasiti da σ_u^A ($u \in X^*$) igra važnu ulogu u determinizaciji fazi automata $\mathscr{A}$ (cf. [53, 56]), i samim tim, fazi skupovi τ_u^A ($u \in X^*$) koriste se u determinizaciji reverznih fazi automata $\mathscr{A}$.

Osim toga, za svaki $a \in A$ *levi fazi jezik stanja a* i *desni fazi jezik stanja a* su fazi jezici $\sigma_a, \tau_a : X^* \to L$ definisani sa:

$$\sigma_a^A(u) = \bigvee_{b\in A} \sigma^A(b) \circ \delta_u^A(b,a), \qquad \tau_a(u) = \bigvee_{b\in A} \delta_u^A(a,b) \otimes \tau(b) \qquad u \in X^*.$$

Lako je pokazati da, za svaki $u \in X^*$ i $a \in A$ važi:

$$\sigma_a(u) = \sigma_u(a), \qquad \tau_a(u) = \tau_u(a). \tag{B.34}$$

Dalje, neka su $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ i $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ fazi automati. Fazi relacija $\varphi \in \mathscr{R}(A,B)$ koja je rešenje sistema fazi relacijskih jednačina:

$$\sigma^A \leqslant \sigma^B \circ \varphi^{-1} \tag{B.35}$$

$$\varphi^{-1} \circ \tau_u^A \leqslant \tau_u^B \qquad u \in X^* \tag{B.36}$$

se naziva *slaba forward simulacija*, i ako je φ rešenje sistema:

$$\tau^A \leqslant \varphi \circ \tau^B \tag{B.37}$$

$$\sigma_u^A \circ \varphi \leqslant \sigma_u^B \qquad u \in X^* \tag{B.38}$$

naziva se *slaba backward simulacija*.
Ako su obe relacije φ i φ^{-1} slabe forward simulacije, tj. ako φ zadovoljava (B.35), (B.36) i

$$\sigma^B \leqslant \sigma^A \circ \varphi \tag{B.39}$$

$$\varphi \circ \tau_u^B \leqslant \tau_u^A \qquad u \in X^* \tag{B.40}$$

tada je φ *slaba forward bisimulacija*, i ako φ zadovoljava (B.38), (B.37) i

$$\tau^B \leqslant \varphi^{-1} \circ \tau^A \tag{B.41}$$

$$\sigma_u^B \circ \varphi^{-1} \leqslant \sigma_u^A \qquad u \in X^* \tag{B.42}$$

tj. ako su obe relacije φ i φ^{-1} slabe backward simulacije, tada se φ naziva *slaba backward bisimulacija*.

Ovi koncepti uopštavaju pojam simulacija i bisimulacija za fazi automate.
Jednostavnosti radi, relaciju φ zvaćemo samo *slaba simulacija* ako je φ slaba forward ili slaba backward simulacija, i samo *slaba bisimulacija* ako je φ slaba forward ili slaba backward bisimulacija.

Lema 2.2. *Neka su $\mathscr{A} = (A, \sigma^A, \delta^A, \tau^A)$ i $\mathscr{B} = (B, \sigma^B, \delta^B, \tau^B)$ fazi automati i neka je $\varphi \in \mathscr{R}(A, B)$ fazi relacija, tada:*

(a) *Ako je φ forward (resp. backward) simulacija, tada je φ slaba forward (resp. backward) simulacija;*
(b) *Ako je φ forward (resp. backward) bisimulacija, tada je φ slaba forward (resp. backward) bisimulacija.*

Lema 2.3. *Neka su $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ i $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ fazi automati i neka je $\varphi \in \mathscr{R}(A, B)$ fazi relacija, tada:*

(a) *Ako je φ slaba simulacija, tada je $L(\mathscr{A}) \leqslant L(\mathscr{B})$;*
(b) *Ako je φ slaba bisimulacija, tada je $L(\mathscr{A}) = L(\mathscr{B})$.*

Sledeća teorema može se lako dokazati korišćenjem definicija slabe forward i backward simulacije i reverznog fazi automata.

Lema 2.4. *Neka su $\mathscr{A} = (A, \delta^A, \sigma^A, \tau^A)$ i $\mathscr{B} = (B, \delta^B, \sigma^B, \tau^B)$ fazi automati. Fazi relacija $\varphi \in \mathscr{R}(A, B)$ je slaba backward simulacija između automata $\mathscr{A}$ i $\mathscr{B}$ ako i samo ako je φ slaba forward simulacija između reverznih fazi automata $\bar{\mathscr{A}}$ i $\bar{\mathscr{B}}$.*

Iz ove leme, zaključujemo da za svako tvrđenje koje važi za slabe forward simulacije (resp. bisimulacije) za sve automate, postoji dogovarajuće tvrđenje za slabe backward simulacije (resp. bisimulacije). Stoga ćemo obratiti pažnju samo na slabe forward simulacije (resp. bisimulacije).

Prva teorema daje metod za određivanje da li postoji slaba forward simulacija između dva automata i predstavlja metod za konstrukciju najveće, kad god postoji, dok druga teorema daje istu proceduru za slabe forward bisimulacije.

Teorema 2.5. *Neka su $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ i $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ fazi automati i neka je $\varphi \in \mathscr{R}(A,B)$ fazi relacija definisana sa:*

$$\varphi(a,b) = \bigwedge_{u\in X^*} \tau_u^A(a) \to \tau_u^B(b)$$

za svaki $a \in A, b \in B$. Ako φ zadovoljava (B.35), tada je φ najveća slaba forward simulacija iz $\mathscr{A}$ u $\mathscr{B}$. Ako φ ne zadovoljava (B.35), tada ne postoji nijedna slaba forward simulacija iz $\mathscr{A}$ u $\mathscr{B}$.

Teorema 2.6. *Neka su $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ i $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ fazi automati i neka je $\varphi \in \mathscr{R}(A,B)$ fazi relacija definisana sa:*

$$\varphi(a,b) = \bigwedge_{u\in X^*} \tau_u^A(a) \leftrightarrow \tau_u^B(b), \tag{B.43}$$

za sve $a \in A, b \in B$. Ako φ zadovoljava (B.35) i (B.39), tada je φ najveća slaba forward simulacija iz $\mathscr{A}$ u $\mathscr{B}$, i φ je parcijalna fazi funkcija, inače, ne postoji nijedna slaba forward bisimulacija iz $\mathscr{A}$ u $\mathscr{B}$.

Slabe bisimulacije automata $\mathscr{A}$ u sam taj automat, koje su pored toga i relacije ekvivalencije, zvaćemo slabe bisimulacione ekvivalencije na $\mathscr{A}$. Primetimo, na osnovu predhodne teoreme, da je najveća slaba bisimulacija na datom automatu zapravo najveća slaba bisimulaciona ekvivalencija na tom automatu.

U nastavku ćemo razmatrati slabe simulacije i bisimulacije koje su uniformne fazi relacije.

Neka su $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ i $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ fazi automati. Bijekcija ϕ iz A u B se naziva *slabi forward izomorfizam* fazi automata $\mathscr{A}$ i $\mathscr{B}$ ako

$$\sigma^A(a) = \sigma^B(\phi(a)), \qquad a \in A, \tag{B.44}$$

$$\tau_u^A(a) = \tau_u^B(\phi(a)), \qquad a \in A, u \in X^*, \tag{B.45}$$

i *slabi backward izomorfizam* fazi automata $\mathscr{A}$ i $\mathscr{B}$ ako

$$\sigma_u^A(a) = \sigma_u^B(\phi(a)), \qquad a \in A, u \in X^*, \tag{B.46}$$

$$\tau^A(a) = \tau^B(\phi(a)), \qquad a \in A. \tag{B.47}$$

Teorema 2.7. *Neka su $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ i $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ fazi automati, i $\varphi \in \mathscr{R}(A,B)$ uniformna fazi relacija. Tada je φ slaba forward bisimulacija ako i samo ako važi sledeće:*

(1) *E_A^φ je slaba forward bisimulaciona equivalencija na fazi automatu $\mathscr{A}$;*
(2) *E_B^φ je slaba forward bisimulaciona equivalencija na fazi automatu $\mathscr{B}$;*
(3) *$\tilde{\varphi}$ je slabi forward izomorfizam faktor fazi automata $\mathscr{A}/E_A^\varphi$ i $\mathscr{B}/E_B^\varphi$.*

Teorema 2.8. *Neka su $\mathscr{A} = (A,\sigma^A,X,\delta^A,\tau^A)$ i $\mathscr{B} = (B,\sigma^B,X,\delta^B,\tau^B)$ fazi automati, i neka je E slaba forward bisimulaciona ekvivalencija na $\mathscr{A}$ i F slaba forward bisimulaciona ekvivalencija na $\mathscr{B}$.*
Tada postoji uniformna slaba forward bisimulacija $\varphi \in \mathscr{R}(A,B)$ takva da

$$E_A^\varphi = E \qquad i \qquad E_B^\varphi = F, \tag{B.48}$$

ako i samo ako postoji slabi forward izomorfizam $\phi : \mathscr{A}/E \to \mathscr{B}/F$ takav da za svaki $a_1,a_2 \in A$ važi

$$\tilde{E}(E_{a_1},E_{a_2}) = \tilde{F}(\phi(E_{a_1}),\phi(E_{a_1})). \tag{B.49}$$

Naredna teorema daje karakterizaciju uniformnih slabih forward bisimulacija u terminima reverznih Nerodovih automata. Na sličan način možemo dati karakterizaciju uniformnih slabih backward bisimulacija u terminima Nerodovih automata.

Teorema 2.9. *Neka su $\mathscr{A} = (A,\delta^A,\sigma^A,\tau^A)$ i $\mathscr{B} = (B,\delta^B,\sigma^B,\tau^B)$ fazi automati, i neka je $\varphi \in \mathscr{R}(A,B)$ uniformna fazi relacija. Tada je φ slaba forward bisimulacija iz $\mathscr{A}$ u $\mathscr{B}$ ako i samo ako zadovoljava (B.35) i (B.39), i funkcije*

$$\tau_u^A \mapsto \varphi^{-1} \circ \tau_u^A, \quad \tau_u^B \mapsto \varphi \circ \tau_u^B, \quad u \in X^*,$$

su međusobno inverzni izomorfizmi između reverznih Nerodovih automata $\bar{\mathscr{A}}_N$ i $\bar{\mathscr{B}}_N$.

5. Algoritmi Paige-Tarjanovog tipa

Neka je $\mathscr{A} = (A,\sigma,\delta,\tau)$ fazi automat nad X i $\mathscr{L}$. Fazi kvazi-uređenje $R \in \mathscr{R}(A)$ naziva se *desno stabilno* ili samo *d-stabilno* ako zadovoljava sledeći sistem fazi relacijskih nejednačina:

$$R \circ \delta_x \circ R \leqslant \delta_x \circ R, \quad x \in X. \tag{B.50}$$

Slično, fazi kvazi-uređenje $R \in \mathscr{R}(A)$ zove se *levo stabilno*(kraće *l-stabilno*) ako zadovoljava sledeći sistem fazi relacijskih nejednačina:

$$R \circ \delta_x \circ R \leqslant R \circ \delta_x, \quad x \in X. \tag{B.51}$$

Dalje, fazi kvazi-uređenje $R \in \mathscr{R}(A)$ naziva se *desno-invarijantno* na $\mathscr{A}$ ako je d-stabilno i zadovoljava sledeću nejednakost:

$$R \circ \tau \leqslant \tau. \tag{B.52}$$

Analogno, R je *levo-invarijantno* na $\mathscr{A}$ ako je l-stabilno i zadovoljava sledeću nejednakost:

$$\sigma \circ R \leqslant \sigma. \tag{B.53}$$

Na osnovu činjenice da je fazi kvazi-uređenje R rešenje nejednačine (B.52) ako i samo ako:

$$R \leqslant \tau \backslash \tau = \pi^{fs},$$

zaključujemo da je d-stabilno fazi kvazi-uređenje na datom fazi automatu $\mathscr{A}$ desno-invarijantno kvazi-uređenje na $\mathscr{A}$ ako i samo ako je sadržano u π^{fs}. Slično, kako je fazi kvazi-uređenje R rešenje nejednačine (B.53) ako i samo ako:

$$R \leqslant \sigma \backslash \sigma = \pi^{bs},$$

dobijamo da je R l-stabilno fazi kvazi-uređenje na datom fazi automatu $\mathscr{A}$ levo-invarijantno kvazi-uređenje na $\mathscr{A}$ ako i samo ako je sadržano u π^{bs}.

Važno je pomenuti da se pojam d-stabilnosti i l-stabilnosti kvazi-uređenja na fazi automatu $\mathscr{A} = (A, X, \delta, \sigma, \tau)$, koja je ovde uvedena, poklapa sa pojmom desno-inverijantnosti odnosno levo-invarijantnosti kvazi-uređenja na datom fazi tranzicionom sistemu $\mathscr{A} = (A, X, \delta)$. Stoga se problem nalaženja najvećeg desno-invarijantnog fazi kvazi-uređenja na odgovarajućem fazi automatu svodi na problem nalaženja najvećeg desno-invarijantnog fazi kvazi-uređenja na odgovarajućem tranzicionom sistemu sadržanom u datom fazi kvazi-uređenju.

Neka je $\mathscr{A} = (A, X, \delta^A)$ fazi tranzicioni sistem, i neka je R fazi kvazi-uređenje na $\mathscr{A}$.

Sa $\mathscr{L}(\mathscr{A}, R)$ označićemo podalgebru od $\mathscr{L}$ generisanu sa $\delta^A(A \times X \times A) \cup R(A \times A)$.

Sledeća teorema daje metod za računanje najvećeg desno-invarijantnog kvazi-uređenja na datom fazi automatu:

Teorema 2.10. *Neka je $\mathscr{A} = (A, \sigma, \delta, \tau)$ fazi automat nad X i $U \in \mathscr{R}(A)$ univerzalna relacija na A.*

Definišemo nizove $\{Q_k\}_{k\in N}$ i $\{R_k\}_{k\in N}$ fazi kvazi-uređenja na A na sledeći način: Inicijalno za $k = 1$

$$\begin{aligned} R_1 &= U, \\ Q_1 &= \pi^{fs} \wedge \Big(\bigwedge_{x\in X} (\delta_x \circ R_1^a)/(\delta_x \circ R_1^a) \Big), \end{aligned} \tag{B.54}$$

gde je $a \in A$ proizvoljni element.

Dalje, za svako $k \geqslant 2$ ponavljamo sledeći postupak: Nalazimo $a \in A$, tako da je $R_k^a \neq Q_k^a$ i stavljamo

$$R_{k+1} = R_k \wedge (Q_k^a / Q_k^a), \tag{B.55}$$

$$Q_{k+1} = Q_k \wedge \Big(\bigwedge_{x\in X} \Big(\bigwedge_{b\in S_k^a} (\delta_x \circ R_{k+1}^b)/(\delta_x \circ R_{k+1}^b) \Big)\Big), \tag{B.56}$$

($S_k^a = \{b \in A \mid Q_k^a(b) \neq 0\}$), dok ne postane $R_k = Q_k$. Tada:

(a) Nizovi $\{Q_k\}_{k\in N}$ i $\{R_k\}_{k\in N}$ su opadajući;
(b) $Q_k \leqslant R_k$ za svaki $k \in N$;
(c) Za svaki $k \in N$, i sve $c \in A$ važi :

$$Q_k \leqslant \bigwedge_{x\in X} \delta_x \circ R_k^c / \delta_x \circ R_k^c; \tag{B.57}$$

(d) Q_k i R_k su fazi kvazi-uređenja za svaki $k \in N$;
(e) Ako $R_k = Q_k$, za neko $k \in N$, tada je Q_k najveće desno-inarijantno fazi kvazi-uređenje na $\mathscr{A}$;
(f) Ako je $\mathscr{A}$ konačan i $\mathscr{L}(\mathscr{A}, R)$ zadovoljava UOL, tada su nizovi $\{Q_k\}_{k\in N}$ i $\{R_k\}_{k\in N}$ konačni, tj. postoji $k \in N$ tako da je $R_k = Q_k$.

Kompleksnost algoritma baziranog na predhodnoj teoremi je $O(n^5 m)$, gde je n broj stanja automata $\mathscr{A}$ a m veličina alfabeta.

Naredna teorema daje postupak za računanje najveće desno-invarijantne fazi eqvivalencije na datom fazi automatu.

Teorema 2.11. *Neka je $\mathscr{A} = (A, \sigma, \delta, \tau)$ fazi automat nad X, i $U \in \mathscr{R}(A)$ univerzalna relacija na A.*

Definišemo niz $\{E_k\}_{k\in N}$ i $\{X_k\}_{k\in N}$ ekvivalencija na A na sledeći način: Inicijalno za $k = 1$

$$X^1 = U,$$
$$E^1 = \pi^{fb} \wedge \Big((\delta \circ X_1^a) | (\delta \circ X_1^a) \Big), \quad \textit{for some } a \in A. \tag{B.58}$$

U nastavku, za svaki $k \in N$ ponavljamo sledeći korak: Nalazimo $a \in A$, tako da je $X_k^a \neq E_k^a$ i stavljamo

$$X_{k+1} = X_k \wedge (E_k^a | E_k^a),$$
$$E_{k+1} = E_k \wedge \Big((\delta \circ E_k^a) | (\delta \circ E_k^a) \Big) \wedge \Big(\bigwedge_{b\in T_k^a} (\delta \circ X_{k+1}^b) | (\delta \circ X_{k+1}^b) \Big), \tag{B.59}$$

($T_k^a = \{b \in A \mid E_k^a(b) \neq 1\}$), dok ne postane $X_k = E_k$. Tada važi:

(a) Nizovi $\{E_k\}_{k\in N}$ i $\{X_k\}_{k\in N}$ su opadajući;
(b) Za svaki $k \in N$, $E_k \leqslant X_k$;
(c) Za svaki $k \in N$, i sve $c \in A$ važi sledeće:

$$E_k \leqslant \bigwedge_{x\in X} (\delta_x \circ X_k^c) | (\delta_x \circ X_k^c); \tag{B.60}$$

(d) Za svaki $k \in N$, X_k i E_k su fazi ekvivalencije;
(e) Ako je $X_k = E_k$, za neko $k \in N$, tada je E_k najveća desno invarijantna fazi ekvivalencija na $\mathscr{A}$;
(f) Ako je $\mathscr{A}$ konačan i $\mathscr{L}(\mathscr{A},R)$ zadovoljava UOL, tada su nizovi $\{E_k\}_{k\in N}$ i $\{X_k\}_{k\in N}$ konačni, tj. postoji $k \in N$ tako da $X_k = E_k$.

Sledeća teorema, predstavlja specijalan slučaj predhodne teoreme, naime daje postupak za računanje najveće desno-invarijantne ekvivalencije na nedeterminističkom automatu.

Teorema 2.12. *Neka je $\mathscr{A} = (A,\sigma,\delta,\tau)$ nedeterministički automata i $U = A \times A$ univerzalna relacija na A.*

Definišimo nizove $\{E_k\}_{k\in N}$ i $\{R_k\}_{k\in N}$ ekvivalencija na A na sledeći način: Inicijalno za $k = 1$

$$R_1 = U, \qquad E_1 = \pi^{fb} \cap \bigcap_{x\in X}\Big((\delta_x \circ U^a)|(\delta_x \circ U^a)\Big),$$

gde je a proizvoljni element iz A.

Dalje, za svaki $k \in N$ ponavljamo sledeći korak: Nalazimo $a \in A$, tako da je $R_k^a \neq E_k^a$ i stavljamo

$$R_{k+1} = R_k \cap (E_k^a|E_k^a), \tag{B.61}$$

$$E_{k+1} = E_k \cap \Big(\bigcap_{x\in X}\Big((\delta_x \circ (R_k^a - E_k^a))|(\delta_x \circ (R_k^a - E_k^a))\Big) \cap ((\delta_x \circ E_k^a)|(\delta_x \circ E_k^a))\Big), \tag{B.62}$$

dok ne bude $R_k = E_k$. Tada važi:

(a) Nizovi $\{E_k\}_{k\in N}$ i $\{R_k\}_{k\in N}$ su opadajući;
(b) Za svaki $k \in N$, $E_k \subseteq R_k$;
(c) Za svaki $k \in N$, i sve $c \in A$ važi:

$$E_k \subseteq \bigcap_{x\in X}(\delta_x \circ R_k^c)|(\delta_x \circ R_k^c); \tag{B.63}$$

(d) Za svaki $k \in N$, R_k i E_k su relacije ekvivalencije;
(e) Procedura je konačna i završava se nakon $|A| - 1$ koraka i ukoliko se završi nakon n koraka tada je E_n najveća desno invarijantna ekvivalencija na $\mathscr{A}$.

Primetimo da ova modifikacija ima znatno manju kompleksnost, odnosno kompleksnost ovog algoritma iznosi $O(n^3m)$.

Appendix C
Biography of Author

BIOGRAPHICAL INFORMATION

Ivana Micić was born in July 13th,1984. in Niš, where she finished the Primary School "Branko Miljković" with excellent marks. She got the diploma "Vuk Karadzić" and she was declared as the best pupil in the generation. She finished high school "Bora Stanković" in Niš with all excellent marks. She started her studies on the department of mathematics and informatics, on the Faculty of Sciences and Mathematics, University of Niš, in the school year 2003/2004. and graduated in 2008., with the average grade 9,50. She defended her graduate thesis "Fuzzy equivalence relations and their application " and got excellent mark 10.

During the period from 2006. to 2008. she was receiving the city scholarship for talented students and in July, 2007. she won a scholarship "Travel the Europe" for best students of finial year of studies in the Republic of Serbia. She started her PhD studies in 2008. on the Faculty of Sciences and Mathematics, Department of Computer Science. From that period she started working as the researcher assistant on the same faculty.

From the February 2009. she is a junior researcher on project "Algebraic structures and methods for information processing" (144011). From 2011. she is working as a member of the project "Development of methods for calculation and information processing: theory and application" (174013). In the September 2010. she participated in the project "Natural language processing and automata" on the Technical University Dresden, Germany.

In the January, 2011. she was involved in the teaching process and now she is preforming exercises for the following exams: Basis of informatics, Methodology of e-learning and Theory of automata, algorithms and languages.

LIST OF SCIENTIFIC PAPERS

Papers published in international journals on the SCI list

1. M. Ćirić, J.Ignjatović, M. Bašić, I.Jančić, Nondeterministic automata: equivalence, bisimulations, and uniform relations, Information Sciences 261 (2014) 185-218
2. I.Jančić, Weak bisimulation for fuzzy automata, Fuzzy Sets and Systems (2013) http://dx.doi.org/10.1016/j.fss.2013.10.006.
3. M. Ćirić, J.Ignjatović, I.Jančić, N. Damljanović, Computation of the greatest simulations and bisimulations between fuzzy automata, Fuzzy Sets and Systems 208 (2012) 22-42
4. J. Ignjatović, M. Ćirić, N. Damljanović, I. Jančić, Weakly linear systems of fuzzy relation inequalities: The heterogeneous case, Fuzzy Sets and Systems 199 (2012) 64-91.

Participation in international conferences with published list of abstracts

5. I.Jančić, Z.Jančić, J. Ignjatović, M. Ćirić, Fuzzy automata: Determinization using simulations International Workshop on Weighted Automata: Theory and Applications, WATA 2012, Dresden, Germany, May 29-June 2, 2012.
6. I.Jančić, J. Ignjatović, M. Ćirić, Fuzzy automata: weak bisimulations, International Workshop on Weighted Automata: Theory and Applications, WATA 2010, Leipzig, Germany, May 3-7, 2010.
7. I.Jančić, J. Ignjatović, M. Ćirić, Fuzzy network analysis:Regular equivalences and bisimulation The 3rd Novi Sad Algebraic Conference, Novi Sad, Serbia, August 17-21, 2009.

References

1. L. Aceto, A. Ingolfsdottir, K. G. Larsen, J. Srba, Reactive Systems: Modelling, Specification and Verification, Cambridge University Press, Cambridge, 2007.
2. N. C. Basak, A. Gupta, On quotient machines of a fuzzy automaton and the minimal machine, Fuzzy Sets and Systems 125 (2002) 223–229.
3. R. Bělohlávek, Determinism and fuzzy automata, Information Sciences 143 (2002) 205–209.
4. R. Bělohlávek, Fuzzy Relational Systems: Foundations and Principles, Kluwer, New York, 2002.
5. T.S.Blyth, Lattices and Ordered Algebraic Structures, Springer, 2006.
6. G.Birkhoff, Lattice theory, American mathematical Society,1964.
7. M. P. Béal, S. Lombardy, J. Sakarovitch, On the equivalence of $\mathbb{Z}$-automata, In: L. Caires et al. (eds.), ICALP 2005, Springer, Heidelberg, Lecture Notes in Computer Science 3580 (2005) 397–409.
8. M. P. Béal, S. Lombardy, J. Sakarovitch, Conjugacy and equivalence of weighted automata and functional transducers. In: D. Grigoriev, J. Harrison, and E. A. Hirsch (eds.), CSR 2006, Springer, Heidelberg, Lecture Notes in Computer Science 3967 (2006) 58–69.
9. M. P. Béal, D. Perrin, On the generating sequences of regular languages on k symbols, Journal of the ACM 50 (2003) 955–980.
10. R. Bělohlávek, V. Vychodil, Fuzzy Equational Logic, Springer, Berlin/Heidelberg, 2005.
11. S. L. Bloom, Z. Ésik, Iteration Theories: The Equational Logic of Iterative Processes, EATCS Monographs on Theoretical Computer Science, Springer, Berlin-Heilderberg, 1993.
12. T. Brihaye, Words and bisimulations of dynamical systems, Discrete Mathematics and Theoretical Computer Science 9 (2) (2007) 11–32.
13. P. Buchholz, Bisimulation relations for weighted automata, Theoretical Computer Science 393 (2008) 109–123.
14. J.-M. Champarnaud, F. Coulon, Theoretical study and implementation of the canonical automaton, Technical Report AIA 2003.03, LIFAR, Université de Rouen, 2003.
15. J.-M. Champarnaud, F. Coulon, NFA reduction algorithms by means of regular inequalities, in: Z. Ésik, and Z. Fülöp (eds.), DLT 2003, Lecture Notes in Computer Science 2710 (2003) 194–205.
16. J.-M. Champarnaud, F. Coulon, NFA reduction algorithms by means of regular inequalities, Theoretical Computer Science 327 (2004) 241–253 (erratum: Theoretical Computer Science 347 (2005) 437–40).

17. C. S. Calude, E. Calude, B. Khoussainov, Finite nondeterministic automata: Simulation and minimality, Theoretical Computer Science 242 (2000) 219–235.
18. Y. Cao, G. Chen, E. Kerre, Bisimulations for fuzzy transition systems, IEEE Transactions on Fuzzy Systems 19 (2011) 540-552.
19. M. Ćirić, M. Droste, J. Ignjatović, H. Vogler, Determinization of weighted finite automata over strong bimonoids, Information Sciences 180 (2010) 3497–3520.
20. M. Ćirić, J. Ignjatović, S. Bogdanović, Uniform fuzzy relations and fuzzy functions, Fuzzy Sets and Systems 160 (2009) 1054–1081.
21. M. Ćirić, J. Ignjatovic, M. Bašić, I. Jančić, Nondeterministic automata: Simulation, bisimulation and structural equivalence, Information Science 261 (2014) 185-218
22. M. Ćirić, J. Ignjatović, N. Damljanović, M. Bašić, Bisimulations for fuzzy automata, Fuzzy Sets and Systems 186(2012) 100-139
23. M. Ćirić, J. Ignjatović, I. Jančić, N. Damljanović, Computation of the greatest simulations and bisimulations between fuzzy automata, Fuzzy Sets and Systems 208 (2012) 22-42.
24. C. G. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, Springer, 2008.
25. W. Cheng, Z. Mo, Minimization algorithm of fuzzy finite automata, Fuzzy Sets and Systems 141 (2004) 439-448.
26. M. Ćirić, A. Stamenković, J. Ignjatović, T. Petković, Factorization of fuzzy automata, In: Csuhaj-Varju, E., Ésik, Z. (eds.), FCT 2007, Springer, Heidelberg, Lecture Notes in Computer Science 4639 (2007) 213-225.
27. M. Ćirić, A. Stamenković, J. Ignjatović, T. Petković, Fuzzy relation equations and reduction of fuzzy automata, Journal of Computer and System Sciences 76 (2010) 609-633.
28. C. Câmpeanu, N. Sântean, S. Yu, Mergible states in large NFA, Theoretical Computer Science 330 (2005) 23-34.
29. Y. Z. Cao, M. S. Ying, Supervisory control of fuzzy discrete event systems, IEEE Transactions on Systems, Man, and Cybernetics - Part B 35 (2005) 366-371.
30. Y. Z. Cao, M. S. Ying, Observability and decentralized control of fuzzy discrete-event systems, IEEE Transactions Fuzzy Systems 14 (2006) 202-216.
31. Y. Z. Cao, M. S. Ying, G. Q. Chen, State-based control of fuzzy discrete-event systems, IEEE Transactions on Systems, Man, and Cybernetics - Part B 37(2007) 410-424.
32. J.-M. Champarnaud, D. Ziadi, New finite automaton constructions based on canonical derivatives, in: S. Yu, A. Paun (eds.), CIAA 2000, Springer, Berlin, Lecture Notes in Computer Science 2088 (2001) 94-104.
33. J.-M. Champarnaud, D. Ziadi, Computing the equation automaton of a regular expression in $\mathscr{O}(s^2)$ space and time, in: A. Amir, G. Landau (eds.), CPM 2001, Springer, Berlin, Lecture Notes in Computer Science 2089 (2001) 157-168.
34. M. Demirci, Fuzzy functions and their applications, Journal of Mathematical Analysis and Applications 252 (2000) 495-517.
35. M. Demirci, Foundations of fuzzy functions and vague algebra based on many-valued equivalence relations, Part I: Fuzzy functions and their applications, International Journal of general Systems 32 (2) (2003) 123-155.
36. M. Demirci, A theory of vague lattices based on many-valued equivalence relations - I: general representation results, Fuzzy Sets and Systems 151 (2005) 437-472.
37. M. Droste, T. Stüber, H. Vogler, Weighted finite automata over strong bimonoids, Information Sciences 180 (2010) 156-166.
38. D. Dubois, H. Prade, Fuzzy Sets and Systems: Theory and Applications, Academic Press, New York, 1980.
39. D. Dubois, H. Prade (eds.), Fundamentals of Fuzzy Sets, The Handbooks of Fuzzy Sets Series, Vol. 1, Kluwer Academic Publishers, 2000.
40. A. Dovier, C. Piazza, A. Policriti, An efficient algorithm for computing bisimulation equivalence, Theoretical Computer Science 311 (2004) 221-256.

41. Z. Ésik, W. Kuich, A generalization of Kozen's axiomatization of the equational theory of the regular sets, in: Words, semigroups, and transductions, World Scientific, River Edge, NJ, 2001, pp. 99-114.
42. Z. Ésik, A. Maletti, Simulation vs. Equivalence, CoRR abs/1004.2426 (2010).
43. M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco,1979.
44. R. Gentilini, C. Piazza, A. Policriti, From bisimulation to simulation: coarsest partition problems, Journal of Automated Reasoning 31 (2003) 73-103.
45. M. M. Gupta, G. N. Saridis, B. R. Gaines, Fuzzy Automata and Decision Processes, North-Holland, New York, 1977.
46. P. Hájek, Mathematics of fuzzy logic, Kluwer, Dordrecht, 1998.
47. T. A. Henzinger, P. W. Kopke, A. Puri, P. Varaiya, What's decidable about hybrid automata? Journal of Computer and System Sciences 57 (1998) 94-124.
48. J. Högberg, A. Maletti, J. May, Backward and forward bisimulation minimisation of tree automata, in: J. Holub, J. Ždárek (eds.), IAA07, Springer, Heidelberg, Lecture Notes in Computer Science 4783 (2007) 109-121.
49. J. Högberg, A. Maletti, J. May, Backward and forward bisimulation minimisation of tree automata, Theoretical Computer Science 410 (2009) 3539-3552.
50. U. Höhle, Commutative, residuated ℓ-monoids, in: U. Höhle and E. P. Klement (Eds.), Non-Classical Logics and Their Applications to Fuzzy Subsets, Kluwer Academic Publishers, Boston, Dordrecht, 1995, pp. 53-106.
51. B. Hrúz, M. C. Zhou, Modeling and control of discrete-event dynamical systems: with Petri nets and other tools, Springer, 2007.
52. J. Ignjatović, M. Ćirić, Formal power series and regular operations on fuzzy languages, Information Sciences 180 (2010) 1104-1120.
53. J. Ignjatović, M. Ćirić, S. Bogdanović, Determinization of fuzzy automata with membership values in complete residuated lattices, Information Sciences 178 (2008) 164-180.
54. J. Ignjatović, M. Ćirić, S. Bogdanović, Fuzzy homomorphisms of algebras, Fuzzy Sets and Systems 160 (2009), 2345-2365.
55. J. Ignjatović, M. Ćirić, S. Bogdanović, On the greatest solutions to weakly linear systems of fuzzy relation inequalities and equations, Fuzzy Sets and Systems 161 (2010) 3081-3113.
56. J. Ignjatović, M. Ćirić, S. Bogdanović, T. Petković, Myhill-Nerode type theory for fuzzy languages and automata, Fuzzy Sets and Systems 161 (2010) 1288-1324.
57. J. Ignjatović, M. Ćirić, N. Damljanović, I. Jančić, Weak linear system of fuzzy relation inequalities: The heterogenous case, Fuzzy Sets and Systems 199 (2012) 64-91.
58. L. Ilie, S. Yu, Constructing NFAs by optimal use of positions in regular expressions, in: A. Apostolico, M. Takeda (eds.), CPM 2002, Springer, Berlin, Lecture Notes in Computer Science 2373 (2002) 279-288.
59. L. Ilie, S. Yu, Algorithms for computing small NFAs, in: K. Diks et al. (eds): MFCS 2002, Lecture Notes in Computer Science 2420 (2002) 328-340.
60. L. Ilie, S. Yu, Reducing NFAs by invariant equivalences, Theoretical Computer Science 306 (2003) 373-390.
61. L. Ilie, S. Yu, Follow automata, Information and Computation 186 (2003) 140-Ű162
62. L. Ilie, G. Navarro, S. Yu, On NFA reductions, in: J. Karhumäki et al. (eds): Theory is Forever, Lecture Notes in Computer Science 3113 (2004) 112-124.
63. L. Ilie, R. Solis-Oba, S. Yu, Reducing the size of NFAs by using equivalences and preorders, in: A. Apostolico, M. Crochemore, and K. Park (eds): CPM 2005, Lecture Notes in Computer Science 3537 (2005) 310-321.
64. I. Jančić, Weak bisimulation for fuzzy automata, Fuzzy Sets and Systems (2013) http://dx.doi.org/10.1016/j.fss.2013.10.006
65. Z. Jančić, J. Ignjatović, M. Ćirić, An improved algorithm for determinization of weighted and fuzzy automata, Information Sciences 181 (2011) 1358-1368.

66. T. Jiang, B. Ravikumar, Minimal NFA problems are hard, SIAM J. Comput. 22 (6) (1993) 1117-1141.
67. P. C. Kannellakis, S. A. Smolka, CCS expressions, finite state processes, and three problems of equivalence, Information and Computation 86 (1990) 43-68.
68. E. Kilic, Diagnosability of fuzzy discrete event systems, Information Sciences 178 (2008) 858-870.
69. G. J. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic, Theory and Application, Prentice-Hall, Englevood Cliffs, NJ, 1995.
70. D. C. Kozen, Automata and Computability, Springer, 1997.
71. T. Kameda, P. Weiner, On the state minimization of nondeterministic finite automata, IEEE Trans. Computers C-19(7) (1970) 617-627.
72. F. Lin, H. Ying, R. D. MacArthur, J. A. Cohn, D. Barth-Jones, L. R. Crane, Decision making in fuzzy discrete event systems, Information Sciences 177 (2007) 3749-3763.
73. E. T. Lee, L. A. Zadeh, Note on fuzzy languages, Information Sciences 1 (1969) 421-434.
74. H. Lei, Y. M. Li, Minimization of states in automata theory based on finite lattice-ordered monoids, Information Sciences 177 (2007) 1413-1421.
75. J. P. Liu, Y. M. Li, The relationship of controllability between classical and fuzzy discrete-event systems, Information Sciences 178 (2008) 4142-4151.
76. Y. M. Li, W. Pedrycz, Fuzzy finite automata and fuzzy regular expressions with membership values in lattice ordered monoids, Fuzzy Sets and Systems 156 (2005) 68-92.
77. N. Lynch, F. Vaandrager, Forward and backward simulations: Part I. Untimed systems, Information and Computation 121 (1995), 214-233.
78. F. Lin, H. Ying, Fuzzy discrete event systems and their observability, in: Proceedings of the 2001 IFSA/NAFIP Conference, 2001, pp. 1271-1276.
79. F. Lin, H. Ying, Modeling and control of fuzzy discrete event systems, IEEE Transactions on Man, Systems and Cybernetics - Part B 32 (2002) 408-415.
80. E. T. Lee, L. A. Zadeh, Note on fuzzy languages, Information Sciences 1 (1969) 421-434.
81. R. Milner, A calculus of communicating systems, in G. Goos and J. Hartmanis (eds.), Lecture Notes in Computer Science, vol. 92, Springer, 1980.
82. R. Milner, Communication and Concurrency, Prentice-Hall International, 1989.
83. R. Milner, Communicating and Mobile Systems: the π-Calculus, Cambridge University Press, Cambridge, 1999.
84. J. N. Mordeson, D. S. Malik, Fuzzy Automata and Languages: Theory and Applications, Chapman & Hall/CRC, Boca Raton, London, 2002.
85. D. S. Malik, J. N. Mordeson, M. K. Sen, Minimization of fuzzy finite automata, Information Sciences 113 (1999) 323-330.
86. B. F. Melnikov, A new algorithm of the state-minimization for the nondeterministic finite automata, Korean J. Comput. Appl. Math. 6(2) (1999) 277-290.
87. B. F. Melnikov, Once more about the state-minimization of the nondeterministic finite automata, Korean J. Comput. Appl. Math. 7(3) (2000) 655-662.
88. R. Paige, R. E. Tarjan, Three partition refinement algorithms, SIAM Journal of Computing 16 (1987) 973-989.
89. D. Park, Concurrency and automata on infinite sequences, in: P. Deussen (ed.), Proc. 5th GI Conf., Karlsruhe, Germany, Lecture Notes in Computer Science 104 (1981), Springer-Verlag, pp. 167-183.
90. K. Peeva, Finite L-fuzzy acceptors, regular L-fuzzy grammars and syntactic pattern recognition, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 12 (2004) 89-104.
91. K. Peeva, Finite L-fuzzy machines, Fuzzy Sets and Systems 141 (2004) 415-437.
92. K. Peeva, Y. Kyosev, Fuzzy Relational Calculus: Theory, Applications, and Software (with CD-ROM), in Series "Advances in Fuzzy Systems - Applications and Theory", Vol 22, World Scientific, 2004.
93. K. Peeva, Z. Zahariev, Computing behavior of finite fuzzy machines - Algorithm and its application to reduction and minimization, Information Sciences 178 (2008) 4152-4165.

94. T. Petkovic, Congruences and homomorphisms of fuzzy automata, Fuzzy Sets and Systems 157 (2006) 444-458.
95. Ketty Peeva, Finite L-fuzzy machines, Fuzzy Sets and Systems 141 (2004) 415-437
96. D. W. Qiu, Automata theory based on completed residuated lattice-valued logic (I), Science in China, Ser. F, 44 (6) (2001) 419-429.
97. D. W. Qiu, Automata theory based on completed residuated lattice-valued logic (II), Science in China, Ser. F, 45 (6) (2002) 442-452.
98. D. W. Qiu, Characterizations of fuzzy finite automata, Fuzzy Sets and Systems 141 (2004) 391-414.
99. D. Qiu, Supervisory control of fuzzy discrete event systems: a formal approach, IEEE Transactions on Systems, Man and Cybernetics - Part B 35 (2005) 72-88.
100. D. W. Qiu, Pumping lemma in automata theory based on complete residuated lattice-valued logic: A note, Fuzzy Sets and Systems 157 (2006) 2128-2138.
101. D. W. Qiu, F. C. Liu, Fuzzy discrete-event systems under fuzzy observability and a test algorithm, IEEE Transactions on Fuzzy Systems 17 (3) (2009), 578-589.
102. F. Ranzato, F. Tapparo, Generalizing the Paige-Tarjan algorithm by abstract interpretation, Information and Computation 206 (2008) 620-651.
103. M. Roggenbach, M. Majster-Cederbaum, Towards a unified view of bisimulation: a comparative study, Theoretical Computer Science 238 (2000) 81-130.
104. S. Roman, Lattices and Ordered Sets, Springer, New York, 2008.
105. D. Saha, An incremental bisimulation algorithm, In: V. Arvind, S. Prasad (eds.), FSTTCS 2007, Springer, Heidelberg, Lecture Notes in Computer Science 4855 (2007), 204-215.
106. E. Sanchez, Resolution of composite fuzzy relation equations, Information and Control 30 (1976) 38-48.
107. E. Sanchez, Solutions in composite fuzzy relation equations: application to medical diagnosis in Brouwerian logic, in: M. M. Gupta, G. N. Saridis, B. R. Gaines (Eds.), Fuzzy Automata and Decision Processes, North-Holland, Amsterdam, 1977, pp. 221-234.
108. E. Sanchez, Resolution of eigen fuzzy sets equations, Fuzzy Sets and Systems 1 (1978) 69-74.
109. D. Sangiorgi, On the origins of bisimulation and coinduction, ACM Transactions on Programming Languages and Systems 31 (4) (2009) 111-151.
110. E. S. Santos, Maximin automata, Information and Control 12 (1968) 367-377.
111. E. S. Santos, On reduction of maxŰmin machines, Journal of Mathematical Analysis and Applications 37 (1972) 677-686.
112. E. Santos, Max-product machines, J. Math. Anal. Appl. 37 (1972) 677-686.
113. E. S. Santos, Fuzzy automata and languages, Information Sciences 10 (1976) 193-197.
114. H. Sengoku, Minimization of nondeterministic finite automata, Master thesis, Kyoto University, 1992.
115. S. S. Skiena, The Algorithm Design Manual, Springer, London, 2008.
116. A. Stamenković, M. Ćirić, Construction of fuzzy automata from fuzzy regular expressions, Fuzzy Sets and Systems 199 (2012) 1-27
117. A. Stamenković, M. Ćirić, J. Ignjatović, Reduction of fuzzy automata by means of fuzzy quasi-orders, Information Sciences, http://dx.doi.org/10.1016/j.ins.2014.02.028
118. D. D. Sun, Y. M. Li, W. W. Yang, Bisimulation relations for fuzzy finite automata, Fuzzy Systems and Mathematics 23 (2009) 92-100 (in Chinese).
119. S. S. Skiena, The Algorithm Design Manual, Springer, London, 2008.
120. S. Yu, Regular languages, in: Handbook of Formal Languages (G. Rozenberg, A. Salomaa, eds.), Vol 1, Springer-Verlag, Berlin - Heidelberg, 1997, pp. 41-110.
121. W. Wechler, The Concept of Fuzziness in Automata and Language Theory, Akademie-Verlag, Berlin, 1978.
122. W. G. Wee, On generalizations of adaptive algorithm and application of the fuzzy sets concept to pattern classification, Ph.D. Thesis, Purdue University, June 1967.

123. W. G. Wee, K. S. Fu, A formulation of fuzzy automata and its application as a model of learning systems, IEEE Transactions on Systems, Man and Cybernetics 5 (1969) 215-223.
124. L. H. Wu, D. W. Qiu, Automata theory based on complete residuated lattice-valued logic: Reduction and minimization, Fuzzy Sets and Systems 161 (2010) 1635-1656.
125. H. Xing, D. W. Qiu, Pumping lemma in context-free grammar theory based on complete residuated lattice-valued logic, Fuzzy Sets and Systems 160 (2009) 1141-1151.
126. H. Xing, D. W. Qiu, Automata theory based on complete residuated lattice-valued logic: A categorical approach, Fuzzy Sets and Systems 160 (2009) 2416-2428.
127. H. Xing, D. W. Qiu, F. C. Liu, Automata theory based on complete residuated lattice-valued logic: Pushdown automata, Fuzzy Sets and Systems 160 (2009) 1125-1140.
128. H. Xing, D. W. Qiu, F. C. Liu, Z. J. Fan, Equivalence in automata theory based on complete residuated lattice-valued logic, Fuzzy Sets and Systems 158 (2007) 1407-1422.

Прилог 1.

ИЗЈАВА О АУТОРСТВУ

Изјављујем да је докторска дисертација, под насловом

БИСИМУЛАЦИЈЕ ЗА ФАЗИ АУТОМАТЕ

- резултат сопственог истраживачког рада,
- да предложена дисертација, ни у целини, ни у деловима, није била предложена за добијање било које дипломе, према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица.

У Нишу, ___________________

Аутор дисертације: Ивана Мицић

Потпис докторанда:

____

Прилог 2.

ИЗЈАВА О ИСТОВЕТНОСТИ ШТАМПАНЕ И ЕЛЕКТРОНСКЕ ВЕРЗИЈЕ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Име и презиме аутора:	Ивана Мицић
Студијски програм:	Информатика
Наслов рада:	Бисимулације за фази аутомате
Ментор:	Јелена Игњатовић

Изјављујем да је штампана верзија моје докторске дисертације истоветна електронској верзији, коју сам предао/ла за уношење у Дигитални репозиторијум Универзитета у Нишу.

Дозвољавам да се објаве моји лични подаци, који су у вези са добијањем академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада, и то у каталогу Библиотеке, Дигиталном репозиторијуму Универзитета у Нишу, као и у публикацијама Универзитета у Нишу.

У Нишу, ___________________

Аутор дисертације:

____

Потпис докторанда:

____

Прилог 3.

ИЗЈАВА О КОРИШЋЕЊУ

Овлашћујем Универзитетску библиотеку „Никола Тесла“ да, у Дигитални репозиторијум Универзитета у Нишу, унесе моју докторску дисертацију, под насловом:

БИСИМУЛАЦИЈЕ ЗА ФАЗИ АУТОМАТЕ

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату, погодном за трајно архивирање.

Моју докторску дисертацију, унету у Дигитални репозиторијум Универзитета у Нишу, могу користити сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons), за коју сам се одлучио/ла.

1. Ауторство
2. Ауторство – некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – делити под истим условима
5. Ауторство – без прераде
6. Ауторство – делити под истим условима

(Молимо да подвучете само једну од шест понуђених лиценци; кратак опис лиценци је у наставку текста).

У Нишу, ________________

Аутор дисертације: Ивана Мицић

____

Потпис докторанда

____

Прилог 4/1

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ НИШ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:	
Идентификациони број, **ИБР**:	
Тип документације, **ТД**:	монографска
Тип записа, **ТЗ**:	текстуални / графички
Врста рада, **ВР**:	докторска дисертација
Аутор, **АУ**:	Ивана З. Мицић
Ментор, **МН**:	Јелена Игњатовић
Наслов рада, **НР**:	Бисимулације за фази аутомате
Језик публикације, **ЈП**:	енглески
Језик извода, **ЈИ**:	српски
Земља публиковања, **ЗП**:	Србија
Уже географско подручје, **УГП**:	Србија
Година, **ГО**:	2014.
Издавач, **ИЗ**:	ауторски репринт
Место и адреса, **МА**:	Ниш, Вишеградска 33.
Физички опис рада, **ФО**: (поглавља/страна/цитата/табела/слика/графика/прилога)	154 стр., граф. прикази
Научна област, **НО**:	рачунарске науке
Научна дисциплина, **НД**:	теорија израчунавања
Предметна одредница / Кључне речи, **ПО**:	фази аутомати и језици, фази бисимулације
УДК	519.71, 519.713, 519.76
Чува се, **ЧУ**:	библиотека
Важна напомена, **ВН**:	
Извод, **ИЗ**:	У докторској дисертацији представљене су бисимулације за фази аутомате. Новине које су предложене јесу испитивање егзистенције и представљање ефективних потупака за рачунање највећих бисимулација одговарајућих система, као и уопштење појма бисимулације са циљем да се добију релације које боље апроксимирају језичку еквиваленцију или дају бољу редукцију фази аутомата, такозване слабе форвард и беквард бисимулације. Поред тога разматрана је редукција фази помоћу десно и лево инваријантних еквиваленција и предложени су алгоритми за рачунање највећих еквиваленција овог типа.
Датум прихватања теме, **ДП**:	14.11.2012.
Датум одбране, **ДО**:	}
Чланови комисије, **КО**: Председник:	
Члан:	
Члан, ментор:	

Образац Q4.09.13 - Издање 1

Прилог 4/2

ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ
НИШ

KEY WORDS DOCUMENTATION

Accession number, **ANO**:		
Identification number, **INO**:		
Document type, **DT**:		**monograph**
Type of record, **TR**:		**textual / graphic**
Contents code, **CC**:		**doctoral dissertation**
Author, **AU**:		**Ivana Z. Micić**
Mentor, **MN**:		**Jelena Ignjatović**
Title, **TI**:		BISIMULATIONS FOR FUZZY AUTOMATA
Language of text, **LT**:		**English**
Language of abstract, **LA**:		**Serbian**
Country of publication, **CP**:		**Serbia**
Locality of publication, **LP**:		**Serbia**
Publication year, **PY**:		**2014**
Publisher, **PB**:		**author's reprint**
Publication place, **PP**:		**Niš, Višegradska 33.**
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/		**154 p. ; graphic representations**
Scientific field, **SF**:		**Computer Science**
Scientific discipline, **SD**:		**Theory of computing**
Subject/Key words, **S/KW**:		**fuzzy automata and languages, fuzzy bisimulations**
UC		519.71, 519.713, 519.76
Holding data, **HD**:		**library**
Note, **N**:		
Abstract, **AB**:		In the doctoral dissertation bisimulations for fuzzy automata are presented. New results that have been proposed here are algorithms for chacking weather there exists simulation/bisimulation between the given fuzzy automata and compute the greatest one, whenever it exists. Also, more general types of bisimulations, which provide better approximations of the language-equivalence and better results in the state reduction, called weak forward and backward bisimulations, are introduced. Moreover, right and left invariant equivalences are discussed.
Accepted by the Scientific Board on, **ASB**:		14.11.2012.
Defended on, **DE**:		
Defended Board, **DB**:	President:	
	Member:	
	Member, Mentor:	