Characterization and computation of approximate bisimulations for fuzzy automata*

Ivana Micić[a], Linh Anh Nguyen[b,c], Stefan Stanimirović[a,*]

[a] *University of Niš, Faculty of Sciences and Mathematics, Višegradska 33, 18000 Niš, Serbia*
[b] *Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland*
[c] *Faculty of Information Technology, Nguyen Tat Thanh University, Ho Chi Minh City, Vietnam*

Abstract

Approximate bisimulations for fuzzy automata have recently drawn attention of researches, since they allow to correlate different fuzzy automata which behave equivalently only to the chosen degree. This is of a particular interest in the state reduction of fuzzy automata, since it allows us to find a fuzzy automaton with a much smaller number of states and a slightly different behaviour from the original fuzzy automaton. The contribution of this paper is twofold. First, we study special types of approximate bisimulations which allow us to correlate all elements from both sets of states of two fuzzy automata. We show their properties, and particularly that they induce the so-called approximate isomorphism between the corresponding factor fuzzy automata. We pay a special attention to similarities and differences between homotypic and heterotypic approximate bisimulations. Second, we provide an efficient algorithm with the complexity $O((m+n)n)$ for computing the greatest λ-approximate bisimulation between two finite fuzzy automata over the Gödel structure (if it exists), where λ is the threshold for approximation, n is the number of states and m is the number of non-zero transitions of the automata. This algorithm gives a great reduction of complexity when compared to the previously developed one, which has the complexity $O(mn^5)$. We also design an algorithm with the complexity $O((m+n)n)$ for computing the greatest $\lambda \in [0,1]$ such that there exists a λ-approximate bisimulation between two given finite fuzzy automata over the Gödel structure.

Keywords: Fuzzy automata, Fuzzy bisimulation, Approximate bisimulation, Uniform fuzzy relation

1. Introduction

Real-life systems have been modelled via various mathematical notions, such as weighted graphs, weighted automata, labelled transition systems, weighted networks, weighted Petri nets, discrete event systems, etc., depending on the fields of application. For a chosen mathematical notion, it happens frequently that multiple models exist for the observed system. Thus, it comes not as a surprise that numerous techniques have been developed to determine the equivalence of these models.

When real-life systems are modelled by fuzzy automata, there are many ways to check the equivalence of two systems. One way is to determine whether there exists an isomorphism (in a mathematical sense) between the states of two fuzzy automata, which can be too rigorous and computationally hard. The other way can be to determine whether two fuzzy automata behave in the same way, i.e., if they accept the same input words with the identical membership value. This also can be computationally hard, and even

*Ivana Micić and Stefan Stanimirović acknowledge the support by Science Fund of the Republic of Serbia, Grant No. 7750185, Quantitative Automata Models: Fundamental Problems and Applications - QUAM, and Ministry of Education, Science and Technological Development, Republic of Serbia under Grant 174013 and Contract 451-03-68/2022-14/200124

*Corresponding author.

Email addresses: ivana.micic@pmf.edu.rs (Ivana Micić), nguyen@mimuw.edu.pl (Linh Anh Nguyen), stefan.stanimirovic@pmf.edu.rs (Stefan Stanimirović)

impractical in some systems with concurrent behaviour (cf. [6, 34, 35, 39, 44]). The most common way is to use the established notion of bisimulation, which intuitively allows us to group fuzzy automata by the way they match each other's moves. In other words, the notion of bisimulation allows us to model the indiscernibility of states of fuzzy automata. The usefulness of bisimulations lies in the fact that they are easier to establish and compute, while their existence implies that the observed fuzzy automata will have the same behaviour (or even for some special types of bisimulations, there exists an isomorphism between fuzzy automata).

In fact, the notion of bisimilarity can be traced back to the works of Milner [42] and Park [53], where it has been used in concurrency theory to test the behavioural equivalence between processes. Since then, it has experienced an expansion in many other fields, such as modal logic, functional and object-oriented programming languages, compiler optimizations, program analysis, verification tools, etc. (for more details we refer the reader to [18, 24, 25, 38, 43, 58, 65]). Bisimulations have been extensively studied in the literature of fuzzy sets, including fuzzy automata [5, 12, 13, 32, 41, 52, 63, 68, 72], fuzzy discrete event systems [17, 72], fuzzy labelled transition systems [6, 7, 19, 46, 66, 67, 69], fuzzy modal or description logics [23, 47–49]. In the fuzzy automata theory, bisimulations have been applied in the state reduction and determinization of fuzzy automata (see [9, 11, 33, 40, 59, 60]). Bisimulations between fuzzy systems have been studied in the literature using two different settings, either as crisp relations or as fuzzy relations between the sets of states of the considered systems.

As stated before, bisimulations allow us to consider different fuzzy automata as equivalent. In a recent series of papers, some authors considered the so-called approximate bisimulations for fuzzy automata. Such bisimulations are defined either as crisp relations [74–76] or as fuzzy relations [61] between the sets of states of fuzzy automata, with the property that, if there exists an approximate bisimulation between two fuzzy automata, then they are equivalent to some degree (i.e., each input word is recognized by the fuzzy automata in degrees that differ up to the chosen limit). Although approximate bisimulations have been studied in various settings, including discrete event systems, (labelled) transition and hybrid systems (see [26–28, 50, 51, 54, 77–79]), only recently they have been studied in the context of fuzzy automata.

Particularly important types of (approximate) bisimulations are those which are surjective and functional, because they are able to connect all states from both fuzzy automata. Moreover, they allow us to construct the aggregated (or factor) fuzzy automaton, which is a fuzzy automaton equivalent to the starting one (or differs in behaviour from the starting one up to the chosen degree in the case of approximate bisimulations), but usually with a smaller number of states. Since Yang and Li [74–76] studied approximate bisimulations for fuzzy automata as crisp relations, they defined surjective and functional approximate bisimulations in the classical sense. This paper, however, deals with approximate bisimulations for fuzzy automata which are defined as fuzzy relations in the way as in [61]. And in the case of fuzzy relations, there is no unique way to define surjective and functional fuzzy relations. One very conventional way is to employ uniform fuzzy relations introduced in [10]. Uniform fuzzy relations were originally intended to serve as fuzzy functions which provide a correspondence between fuzzy functions and fuzzy equivalence relations, analogous to the correspondence between crisp functions and crisp equivalence relations (cf. [10]), but they have been applied in many different situations, to systems of fuzzy relation equations, fuzzy homomorphisms and fuzzy relational morphisms of algebras, bisimulations for fuzzy automata, fuzzy social network analysis, etc. (cf. [10, 12, 29–31]; see also [14, 15]).

The aim of this paper is twofold. First, we give characterizations and properties of uniform approximate bisimulations for fuzzy automata. The importance of using uniform approximate bisimulations lies in the fact that, contrary to classical approximate bisimulations which can only help to determine the degree of equivalence between given automata, uniform approximate bisimulations may be used for factorization of the large fuzzy automata. In other words, for any two very large fuzzy automata, which are inconspicuous for analysis, if we can represent the connection between them by means of uniform approximate bisimulations, we can factorize both automata into smaller automata which preserve previous structure and have special type of homomorphism between them. In that way, we obtain structures more convenient for further examination. Precisely, we show that a (uniform) approximate forward bisimulation between fuzzy automata induces forward bisimulations on both fuzzy automata (Theorem 3.4). We also give an additional characterization of uniform approximate forward bisimulations for fuzzy automata (Theorem 3.5). In addition, we prove

that a uniform approximate bisimulation between fuzzy automata induces what we call here an approximate isomorphism between the corresponding factor fuzzy automata, and conversely, this kind of mapping, when satisfying some additional properties, induces a uniform approximate bisimulation (Theorem 3.7). We also prove the equivalence of the existence of a uniform forward bisimulation between fuzzy automata with a given kernel and co-kernel and the existence of a certain approximate isomorphism between the corresponding factor fuzzy automata (Theorem 3.8). Analogous results are provided for backward-forward bisimulations. Second, we provide an efficient algorithm with the complexity $O((m+n)n)$ for computing the greatest λ-approximate forward bisimulation between fuzzy automata (if it exists), where λ is the threshold for approximation, n is the number of states and m is the number of non-zero transitions of the automata. This algorithm gives a great reduction of complexity when compared to the one given in [61], which has the complexity $O(mn^5)$ (with the same meanings of m and n). Our algorithm is obtained by adapting the algorithm given by Nguyen and Tran in [48] to deal with λ-approximation. Their algorithm computes the greatest fuzzy bisimulation between two finite fuzzy interpretations in the fuzzy description logic $f\mathcal{ALC}$ under the Gödel semantics. As a further contribution, we also design an algorithm with the complexity $O((m+n)n)$ for computing the greatest $\lambda \in [0,1]$ such that there exists a λ-approximate forward bisimulation between two given finite fuzzy automata over the Gödel structure.

It is worth noting that fuzzy automata have a long history of applications in various fields, including control engineering, decision-making, robot control, clinical monitoring, pattern analysis, image processing, and artificial intelligence (see, e.g., [1, 4, 56, 62, 64, 71]). In all such applications where we use a fuzzy automaton as a model of an observed system, it is very convenient to use a simpler fuzzy automaton, even at a cost that the new fuzzy automaton is not strictly language equivalent to the original one, but has slightly different behavior. One of the most celebrated applications of fuzzy automata includes the field of fuzzy discrete event systems [8, 17, 22, 73]. Precisely, in telecommunications, manufacturing and clinical monitoring, it is hard to describe the behavior of a dynamical system by differential equations due to the asynchronous events that occur in the system. It is common to use a fuzzy automaton to model a fuzzy discrete event system because this model overcomes the mentioned issue and captures the uncertainty that exists when describing such systems. Such fuzzy discrete event systems have been succesfully used in fault diagnosis. Precisely, there we have a situation where an output of the monitored system is transformed into the so-called sequence of "templates", and this sequence is further used as an input of a fuzzy automaton with transitions corresponding to the system's templates. Then the system operates or diagnoses a failure depending on whether the fuzzy automaton accepts the input sequence in a chosen degree. Rigatos [57] has provided two application examples in fault diagnosis in the case of a DC-motor, as well as in ECG analysis for automated clinical monitoring. In addition, fuzzy automata have successfully been used in decision-making systems for target control that first preprocess the signal and then perform a two-level decision on the target to achieve optimal control [70]. It turns out that not only such decision-making systems are faster when compared to other ones, but their correct control rate is higher (see [70] for more details). Also, many problems that arise from the field of fuzzy model checking have successfully been modeled by fuzzy automata (see, e.g., [36, 37]). In all described situations, it is desirable to substitute a fuzzy automaton with another one that behaves slightly different from the original one but is more convenient to use. More precisely, we can use a fuzzy automaton that is approximately bisimilar to the original one in such situations.

The structure of the paper is as follows. In Section 2, basic notions and notations concerning Heyting algebras, fuzzy sets, fuzzy relations, uniform fuzzy relations, fuzzy automata and approximate bisimulations are given, which are needed for the rest of the paper. Section 3 is devoted to the study of homotypic approximate bisimulations, while Section 4 is devoted to the study of heterotypic approximate bisimulations. In Section 4, we also present the differences and similarities between homotypic and heterotypic approximate bisimulations. In Section 5, we specify our algorithms of computing the greatest λ-approximate forward bisimulation between two finite fuzzy automata defined over the Gödel structure, if it exists, and the greatest $\lambda \in [0,1]$ which guarantees that existence. Their details together with proofs are presented in the Supplemental Material.

2. Preliminaries

2.1. Heyting algebras

In this paper we use Heyting algebras as structures of membership values. First, recall that a *lattice* is a tuple $\langle L, \wedge, \vee \rangle$, where the binary operations $\wedge$ (meet) and $\vee$ (join) on L satisfy the laws of commutativity, associativity and absorption. Then, a *Heyting algebra* is a tuple $\mathcal{H} = \langle H, \wedge, \vee, \to, 0, 1 \rangle$ such that $\langle H, \wedge, \vee, 0, 1 \rangle$ is a bounded distributive lattice with 0 being the least element and 1 being the greatest element and, for every $x, y, z \in H$, the following *residuation property* is satisfied:

$$x \wedge y \leqslant z \text{ iff } x \leqslant y \to z. \tag{1}$$

A Heyting algebra $\mathcal{H} = \langle H, \wedge, \vee, \to, 0, 1 \rangle$ is complete if the lattice $\langle H, \wedge, \vee \rangle$ is complete. The operation $\to$ is called *implication*. If $\mathcal{H}$ is complete, then $y \to z$ is equal to $\sup\{x \in H \mid x \wedge y \leqslant z\}$, for every $y, z \in H$. It is also known as a *relative pseudo-complement* of y in z (see [21]). Alternatively, Heyting algebras can be defined as idempotent integral commutative residuated lattices (cf. [55]).

While $\wedge$ and $\vee$ serve to model the universal and existential quantifiers, respectively, $\wedge$ and $\to$ serve to model the conjunction and implication of the corresponding logical calculus, respectively. The *biimplication* operation defined by

$$x \leftrightarrow y = (x \to y) \wedge (y \to x),$$

is used to model the equivalence of truth values. For every $x_1, x_2, y_1, y_2 \in H$, we have

$$x_1 \leqslant x_2 \text{ and } y_1 \leqslant y_2 \quad \text{implies} \quad x_1 \wedge y_1 \leqslant x_2 \wedge y_2,$$

and when $\mathcal{H}$ is a complete Heyting algebra, we have

$$x \wedge \left(\bigvee_{i \in I} y_i \right) = \bigvee_{i \in I} (x \wedge y_i), \quad \text{for every } x \in H \text{ and } \{y_i\}_{i \in I} \subseteq H. \tag{2}$$

A prominent example of a Heyting algebra is the so-called *Gödel algebra*, with the support set $[0, 1]$, where $x \wedge y = \min(x, y)$, $x \vee y = \max(x, y)$, $x \to y = 1$ if $x \leqslant y$, and $x \to y = y$ otherwise. It can be easily verified that $x \leftrightarrow y = 1$ for $x = y$ and $x \leftrightarrow y = \min(x, y)$ otherwise. Another important example is the *Boolean algebra*, which is defined as a Heyting algebra in which the relative pseudo-complement $x \to y$ is defined by $(x \to 0) \vee y$, for every $x, y \in H$.

2.2. Fuzzy sets and fuzzy relations

Throughout the paper, $\mathcal{H}$ denotes a complete Heyting algebra $\langle H, \wedge, \vee, \to, 0, 1 \rangle$, while A denotes a non-empty set. A *fuzzy set over A and $\mathcal{H}$*, or simply just a *fuzzy set*, is any mapping from A to H. A fuzzy set taking membership values in the set $\{0, 1\} \subseteq H$ is considered as an ordinary (crisp) subset of A. The *crisp part* of a fuzzy set $\alpha : A \to H$ is a crisp subset of A, defined as $\widehat{\alpha} = \{a \in A \mid \alpha(a) = 1\}$. The collection of all fuzzy sets over A and $\mathcal{H}$ is denoted by H^A. Similarly, any mapping from $A \times B$ to H is called a *fuzzy relation between A and B over $\mathcal{H}$*, or simply just a *fuzzy relation*, and the collection of all of such fuzzy relations is denoted by $H^{A \times B}$. A fuzzy relation between a set A and itself, i.e., a mapping from $A \times A$ to H, is called a *fuzzy relation on A*. The *identity relation* J on a set A is defined as $J(a, a') = 0$, for every $a, a' \in A$ such that $a \neq a'$, and $J(a, a) = 1$ for every $a \in A$. The *inverse fuzzy relation* $\varphi^{-1} \in H^{B \times A}$ of a fuzzy relation $\varphi \in H^{A \times B}$ is defined by $\varphi^{-1}(b, a) = \varphi(a, b)$, for $a \in A$ and $b \in B$.

The inclusion and the equality of two fuzzy sets are defined point-wise, while the meet and the join of a family $\{\alpha_i\}_{i \in I} \subseteq H^A$ of fuzzy sets are given for every $a \in A$ by:

$$\left(\bigwedge_{i \in I} \alpha_i \right)(a) = \bigwedge_{i \in I} \alpha_i(a), \quad \left(\bigvee_{i \in I} \alpha_i \right)(a) = \bigvee_{i \in I} \alpha_i(a).$$

In addition, for every two fuzzy sets $\alpha_1, \alpha_2 \in H^A$, we define their *graded inclusion* $S(\alpha_1, \alpha_2) \in H$ and *graded equality* $E(\alpha_1, \alpha_2) \in H$ by the following expressions:

$$S(\alpha_1, \alpha_2) = \bigwedge_{a \in A} \alpha_1(a) \to \alpha_2(a), \tag{3}$$

$$E(\alpha_1, \alpha_2) = \bigwedge_{a \in A} \alpha_1(a) \leftrightarrow \alpha_2(a). \tag{4}$$

Intuitively, $S(\alpha_1, \alpha_2)$ is the degree in which α_1 is included in α_2, while $E(\alpha_1, \alpha_2)$ is the degree in which α_1 is equivalent to α_2 (cf. [2, 3]). Note that fuzzy relations are actually fuzzy sets over $A \times B$ and $\mathcal{H}$, so all of the above operations defined on fuzzy sets can be easily extended to fuzzy relations.

The *product* $x \wedge \alpha \in H^A$ of an element $x \in H$ and a fuzzy set $\alpha \in H^A$ is defined by:

$$(x \wedge \alpha)(a) = x \wedge \alpha(a), \quad \text{for every } a \in A. \tag{5}$$

Again, (5) can be naturally extended when $\alpha \in H^{A \times B}$. Moreover, for non-empty sets A, B and C, and fuzzy relations $\varphi \in H^{A \times B}$ and $\psi \in H^{B \times C}$, the *product* $\varphi \circ \psi \in H^{A \times C}$ is defined as:

$$(\varphi \circ \psi)(a, c) = \bigvee_{b \in B} \varphi(a, b) \wedge \psi(b, c), \quad \text{for every } a \in A \text{ and } c \in C. \tag{6}$$

Similarly, when $\varphi \in H^{A \times B}$, $\alpha, \alpha_1, \alpha_2 \in H^A$ and $\beta \in H^B$, we define the *products* $\alpha \circ \varphi \in H^B$, $\varphi \circ \beta \in H^A$ and $\alpha_1 \circ \alpha_2 \in H$ as follows:

$$(\alpha \circ \varphi)(b) = \bigvee_{a \in A} \alpha(a) \wedge \varphi(a, b), \quad \text{for every } b \in B,$$

$$(\varphi \circ \beta)(a) = \bigvee_{b \in B} \varphi(a, b) \wedge \beta(b), \quad \text{for every } a \in A,$$

$$\alpha_1 \circ \alpha_2 = \bigvee_{a \in A} \alpha_1(a) \wedge \alpha_2(a).$$

When $\xi \in H^{B \times C}$ and $x \in H$, φ is an element either from $H^{A \times B}$ or H^B, and ψ is an element either from $H^{C \times D}$ or H^C, we have:

$$x \wedge (\varphi \circ \xi) = (x \wedge \varphi) \circ \xi, \tag{7}$$

$$\varphi \circ (\xi \circ \psi) = (\varphi \circ \xi) \circ \psi. \tag{8}$$

Therefore, we can omit the parentheses in (7) and (8). In addition, when $\xi_1, \xi_2 \in H^{B \times C}$, $\varphi \in H^{A \times B}$ and $\psi \in H^{C \times D}$, we have:

$$\xi_1 \leqslant \xi_2 \quad \text{implies} \quad \xi_1^{-1} \leqslant \xi_2^{-1}, \quad \varphi \circ \xi_1 \leqslant \varphi \circ \xi_2 \quad \text{and} \quad \xi_1 \circ \psi \leqslant \xi_2 \circ \psi.$$

Furthermore, when $\varphi, \varphi_i \in H^{A \times B}$ $(i \in I)$ and $\psi, \psi_i \in H^{B \times C}$ $(i \in I)$, we have:

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

$$\varphi \circ \left(\bigvee_{i \in I} \psi_i \right) = \bigvee_{i \in I} (\varphi \circ \psi_i), \quad \left(\bigvee_{i \in I} \varphi_i \right) \circ \psi = \bigvee_{i \in I} (\varphi_i \circ \psi),$$

$$\left(\bigvee_{i \in I} \varphi_i \right)^{-1} = \bigvee_{i \in I} \varphi_i^{-1}.$$

A fuzzy relation φ on A is called a *fuzzy quasi-order* if it satisfies $J \leqslant \varphi$ (reflexivity) and $\varphi \circ \varphi \leqslant \varphi$ (transitivity). Moreover, if φ also satisfies the condition $\varphi^{-1} = \varphi$ (symmetry), then φ is called a *fuzzy*

equivalence. A fuzzy equivalence φ on A for which we have that $\varphi(a_1, a_2) = 1$ implies $a_1 = a_2$, for every $a_1, a_2 \in A$, is called a *fuzzy equality on* A.

For a fuzzy relation $\varphi \in H^{A\times B}$, we define the φ-afterset of a (denoted by $a\varphi \in H^B$), and the φ-foreset of b (denoted by $\varphi b \in H^A$) as:

$$(a\varphi)(b) = \varphi(a, b) \quad \text{and} \quad (\varphi b)(a) = \varphi(a, b), \quad \text{for every } a \in A \text{ and } b \in B.$$

When φ is a fuzzy equivalence on A, then $a\varphi = \varphi a$, for every $a \in A$, and this fuzzy set is called an *equivalence class* of φ with respect to a, and simply denoted by φ_a. The set $A/\varphi = \{\varphi_a \mid a \in A\}$ is called the *factor set of* A *with respect to* φ. For a fuzzy equivalence $\varphi \in H^{A\times A}$ on A, we define the fuzzy relation $\widetilde{\varphi} \in H^{A/\varphi\times A/\varphi}$ on the factor set A/φ by

$$\widetilde{\varphi}(\varphi_{a_1}, \varphi_{a_2}) = \varphi(a_1, a_2),$$

for every $a_1, a_2 \in A$. It can easily be verified that, for every fuzzy equivalence φ on A, $\widetilde{\varphi}$ is a well-defined fuzzy equality on A/φ.

2.3. Uniform fuzzy relations

For a given fuzzy relation $\varphi \in H^{A\times B}$, the *kernel of* φ is the fuzzy relation $\ker\varphi \in H^{A\times A}$ defined by:

$$(\ker\varphi)(a, a') = E(a\varphi, a'\varphi), \quad \text{for every } a, a' \in A,$$

while the *co-kernel of* φ is the fuzzy relation $\operatorname{coker}\varphi \in H^{B\times B}$ defined by:

$$(\operatorname{coker}\varphi)(b, b') = E(\varphi b, \varphi b'), \quad \text{for every } b, b' \in B.$$

It can easily be verified that $\ker\varphi$ is a fuzzy equivalence on A, and $\operatorname{coker}\varphi$ is a fuzzy equivalence on B.

A fuzzy relation $\varphi \in H^{A\times B}$ is said to be a $\mathcal{H}$-*function* if, for every $a \in A$, there exists $b \in B$ such that $\varphi(a, b) = 1$. It can be verified that a fuzzy relation φ is a $\mathcal{H}$-function if and only if there exists a function $f : A \to B$ such that $\varphi(a, f(a)) = 1$, for every $a \in A$ (cf. [16, 20]). Any such function f is called a *crisp description* of φ. The set of all such functions is denoted by $CR(\varphi)$. Moreover, φ is called a *surjective fuzzy relation* if, for every $b \in B$, there exists $a \in A$ such that $\varphi(a, b) = 1$. Also, φ is called a *partial fuzzy function* if $\varphi \circ \varphi^{-1} \circ \varphi \leqslant \varphi$. A fuzzy relation which is both partial fuzzy function and surjective $\mathcal{H}$-function is called a *uniform fuzzy relation*.

Let $\varphi \in H^{B\times B}$ be a fuzzy equivalence. Then, a function $f : A \to B$ is called a φ-*surjective* if, for every $b \in B$, there exists $a \in A$ such that $\varphi(f(a), b) = 1$.

In the sequel, we just recall the main features of uniform fuzzy relations which are used through the rest of the paper (cf. [10]).

Theorem 2.1. *For a fuzzy relation $\varphi \in H^{A\times B}$, the following conditions are equivalent:*

(a) φ *is a uniform fuzzy relation,*
(b) φ^{-1} *is a uniform fuzzy relation,*
(c) φ *is a surjective* $\mathcal{H}$*-function and*

$$\varphi \circ \varphi^{-1} \circ \varphi = \varphi,$$

(d) φ *is a surjective* $\mathcal{H}$*-function and*

$$\ker\varphi = \varphi \circ \varphi^{-1},$$

(e) φ *is a surjective* $\mathcal{H}$*-function and*

$$\operatorname{coker}\varphi = \varphi^{-1} \circ \varphi,$$

(f) *φ is a $\mathcal{H}$-function, and for every $f \in CR(\varphi)$, $a \in A$ and $b \in B$, we have that f is (*$\operatorname{coker}\varphi$*)-surjective and*

$$\varphi(a,b) = (\operatorname{coker}\varphi)(f(a),b),$$

(g) *φ is a $\mathcal{H}$-function, and for every $f \in CR(\varphi)$ and $a, a' \in A$, we have that f is (*$\operatorname{coker}\varphi$*)-surjective and*

$$\varphi(a, f(a')) = (\ker\varphi)(a,a').$$

The following three lemmas are used through the rest of the work (for the proof see [10, 12]).

Lemma 2.2. *Let $\varphi \in H^{A\times B}$ be a uniform fuzzy relation, $\theta = \ker\varphi$ and $\vartheta = \operatorname{coker}\varphi$. Define a mapping $f_\varphi : A/\theta \to B/\vartheta$ by:*

$$f_\varphi(\theta_a) = \vartheta_{f(a)}, \tag{9}$$

for any $a \in A$ and $f \in CR(\varphi)$. Then, f_φ is a well-defined bijective mapping from A/θ to B/ϑ, and the following holds:

$$f_\varphi^{-1} = f_{\varphi^{-1}}.$$

Lemma 2.3. *If $\varphi \in H^{A\times B}$ and $\psi \in H^{B\times C}$ are uniform fuzzy relations such that $\operatorname{coker}\varphi \leqslant \ker\psi$, then $\varphi\circ\psi \in H^{A\times C}$ is also a uniform fuzzy relation.*

Lemma 2.4. *Let $\theta \in H^{A\times A}$ be a fuzzy equivalence on A, and $\vartheta \in H^{B\times B}$ be a fuzzy equivalence on B. Then the following conditions are equivalent:*

(a) *there exists a uniform fuzzy relation $\varphi \in H^{A\times B}$ such that:*

$$\ker\varphi = \theta \quad \textit{and} \quad \operatorname{coker}\varphi = \vartheta;$$

(b) *there exists a ϑ-surjective function $f : A \to B$ such that:*

$$\theta(a_1,a_2) = \vartheta(f(a_1),f(a_2)), \quad \textit{for every } a_1,a_2 \in A;$$

(c) *there exists a bijective function $h : A/\theta \to B/\vartheta$ such that:*

$$\widetilde{\theta}(\theta_{a_1},\theta_{a_2}) = \widetilde{\vartheta}(h(\theta_{a_1}),h(\theta_{a_2})), \quad \textit{for every } a_1,a_2 \in A.$$

2.4. Fuzzy automata

In the rest of the work, we consider fuzzy automata over a complete Heyting algebra, which were initially studied in [61]. We consider fuzzy automata over finite alphabets and with a finite state space.

Let X and X^* denote a finite alphabet and the free monoid over X, respectively. A *fuzzy automaton over $\mathcal{H}$ and X*, or a *fuzzy automaton*, is a quadruple $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A\rangle$, where:

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

A *fuzzy language* over $\mathcal{H}$ and X is any mapping from X^* to H. The *fuzzy language recognized by a fuzzy automaton* $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A \rangle$ is the fuzzy language $L(\mathcal{A}) : X^* \to H$ defined in the following way:

$$L(\mathcal{A})(e) = \sigma^A \circ \tau^A,$$
$$L(\mathcal{A})(u) = \sigma^A \circ \delta^A_{x_1} \circ \delta^A_{x_2} \circ \ldots \circ \delta^A_{x_n} \circ \tau^A, \quad \text{for every } u = x_1 x_2 \ldots x_n \in X^+.$$

In fact, the value $L(\mathcal{A})(u)$ is the degree in which the word u is accepted by the fuzzy automaton $\mathcal{A}$, for every $u \in X^*$. We say that two fuzzy automata $\mathcal{A}$ and $\mathcal{B}$ are equivalent if they accept all words in the same degree, or in other words, if $L(\mathcal{A}) = L(\mathcal{B})$.

Let $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A \rangle$ be a fuzzy automaton and φ an arbitrary fuzzy equivalence on A. Then, we define the *factor fuzzy automaton of* $\mathcal{A}$ (also called the *quotient fuzzy automaton of* $\mathcal{A}$ in some sources) as $\mathcal{A}/\varphi = \langle A/\varphi, \delta^{A/\varphi}, \sigma^{A/\varphi}, \tau^{A/\varphi} \rangle$, where A/φ is the factor set of A w.r.t. φ and the mappings $\delta^{A/\varphi} : (A/\varphi) \times X \times (A/\varphi) \to H$, $\sigma^{A/\varphi} : A \to H$ and $\tau^{A/\varphi} : A \to H$ are defined as:

$$\delta^{A/\varphi}(\varphi_a, x, \varphi_b) = \varphi_a \circ \delta^A_x \circ \varphi_b,$$
$$\sigma^{A/\varphi}(\varphi_a) = \sigma^A \circ \varphi_a,$$
$$\tau^{A/\varphi}(\varphi_a) = \varphi_a \circ \tau^A,$$

for every $a, b \in A$ and $x \in X$. It can easily be verified that $\delta^{A/\varphi}$, $\sigma^{A/\varphi}$ and $\tau^{A/\varphi}$ are all well-defined mappings, i.e., they do not depend on the choice of the class representatives. For more information on fuzzy automata, we refer the reader to [9, 11–13, 41, 59–61].

2.5. Approximate simulations and bisimulations

Two types of approximate simulations and four types of approximate bisimulations for fuzzy automata were initially studied in [61]. Here, we recall the basic definitions from the mentioned paper.

Let $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A \rangle$ and $\mathcal{A}' = \langle A', \delta^{A'}, \sigma^{A'}, \tau^{A'} \rangle$ be fuzzy automata, $\varphi \in H^{A \times A'}$ a fuzzy relation between A and A', and $\lambda \in H$ an arbitrary element from a complete Heyting algebra $\mathcal{H}$. Then φ is called a *λ-approximate forward simulation between $\mathcal{A}$ and $\mathcal{A}'$* if for all $x \in X$:

$$S(\sigma^A, \sigma^{A'} \circ \varphi^{-1}) \geqslant \lambda,$$
$$S(\varphi^{-1} \circ \delta^A_x, \delta^{A'}_x \circ \varphi^{-1}) \geqslant \lambda,$$
$$S(\varphi^{-1} \circ \tau^A, \tau^{A'}) \geqslant \lambda.$$

Also, φ is called a *λ-approximate backward simulation between $\mathcal{A}$ and $\mathcal{A}'$* if for all $x \in X$:

$$S(\tau^A, \varphi \circ \tau^{A'}) \geqslant \lambda,$$
$$S(\delta^A_x \circ \varphi, \varphi \circ \delta^{A'}_x) \geqslant \lambda,$$
$$S(\sigma^A \circ \varphi, \sigma^{A'}) \geqslant \lambda.$$

We drop the words "between $\mathcal{A}$ and $\mathcal{A}'$" from the definitions of approximate simulations and bisimulations when the considered fuzzy automata are clear from the context. Moreover, a fuzzy relation φ is called:

- a *λ-approximate forward bisimulation* (λ-AFB), if φ is a λ-approximate forward simulation and φ^{-1} is a λ-approximate forward simulation;
- a *λ-approximate forward-backward bisimulation*(λ-AFBB), if φ is a λ-approximate forward simulation and φ^{-1} is a λ-approximate backward simulation;
- a *λ-approximate backward-forward bisimulation*(λ-ABFB), if φ is a λ-approximate backward simulation and φ^{-1} is a λ-approximate forward simulation;

- a λ-*approximate backward bisimulation*(λ-ABB), if φ is a λ-approximate backward simulation and φ^{-1} is a λ-approximate backward simulation.

The first type and the last type of approximate bisimulations are called *homotypic*, while the other two types are called *heterotypic*. It is well-known that, if φ is an approximate simulation of any from the mentioned types between $\mathcal{A}$ and $\mathcal{A}'$, then $S(L(\mathcal{A}), L(\mathcal{A}')) \geqslant \lambda$, and if φ is an approximate bisimulation of any from the mentioned types between $\mathcal{A}$ and $\mathcal{A}'$, then $E(L(\mathcal{A}), L(\mathcal{A}')) \geqslant \lambda$ (see [61]).

From the definitions of S and E, we can easily conclude that φ is a λ-AFB iff the following conditions hold for every $x \in X$:

$$\lambda \wedge \sigma^{A} \leqslant \sigma^{A'} \circ \varphi^{-1}, \tag{10}$$

$$\lambda \wedge \varphi^{-1} \circ \delta_x^{A} \leqslant \delta_x^{A'} \circ \varphi^{-1}, \tag{11}$$

$$\lambda \wedge \varphi^{-1} \circ \tau^{A} \leqslant \tau^{A'}, \tag{12}$$

$$\lambda \wedge \sigma^{A'} \leqslant \sigma^{A} \circ \varphi, \tag{13}$$

$$\lambda \wedge \varphi \circ \delta_x^{A'} \leqslant \delta_x^{A} \circ \varphi, \tag{14}$$

$$\lambda \wedge \varphi \circ \tau^{A'} \leqslant \tau^{A}. \tag{15}$$

Analogous inequalities can be derived for λ-ABBs. In addition, φ is a λ-AFBB iff it satisfies the following equations for every $x \in X$:

$$\lambda \wedge \sigma^{A} = \lambda \wedge \sigma^{A'} \circ \varphi^{-1}, \tag{16}$$

$$\lambda \wedge \varphi^{-1} \circ \delta_x^{A} = \lambda \wedge \delta_x^{A'} \circ \varphi^{-1}, \tag{17}$$

$$\lambda \wedge \varphi^{-1} \circ \tau^{A} = \lambda \wedge \tau^{A'}. \tag{18}$$

Analogous equations can be derived for λ-ABFBs.

3. Homotypic approximate bisimulations

In this section, we focus on λ-AFBs for fuzzy automata and study their properties. Analogous properties can also be formulated and proved for λ-ABBs. We begin with the following result, which is an expansion of the [61, Theorem III.6].

Theorem 3.1. *For fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$ such that there exists at least one λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, there exists the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, and this fuzzy relation is also a partial fuzzy function.*

Proof. As proved in [61], the existence of one λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ implies that the join of all λ-AFBs between $\mathcal{A}$ and $\mathcal{A}'$ is the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. Denote it by φ. Now, according to [61, Lemma III.4], $\varphi \circ \varphi^{-1} \circ \varphi$ is also a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, and since φ is the greatest one, we conclude that $\varphi \circ \varphi^{-1} \circ \varphi \leqslant \varphi$. This means that φ is a partial fuzzy function. □

The previous theorem suggests that λ-AFBs which are partial fuzzy functions are of a particular significance. However, as stated in the previous sections, we aim to study λ-AFBs which are surjective $\mathcal{H}$-functions, since these fuzzy relations connect all elements from the sets of states of both fuzzy automata. Since partial fuzzy functions which are surjective $\mathcal{H}$-functions are exactly uniform fuzzy relations, what we aim to study in the sequel are λ-AFBs which are uniform fuzzy relations.

With the following lemma, we bring a connection between the greatest λ-AFB and the existence of a uniform λ-AFB.

Lemma 3.2. *Let $\mathcal{A}$ and $\mathcal{A}'$ be fuzzy automata such that there exists the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. If such a fuzzy relation is not a uniform fuzzy relation, then there is no uniform λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$.*

Proof. According to the previous theorem, the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ is a partial fuzzy function. Denote it by φ. Assume the opposite that φ is not a uniform fuzzy relation, but there exists a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, denoted by ϕ, which is a uniform fuzzy relation. Then, ϕ is a surjective $\mathcal{H}$-function, which means that, for every $a \in A$, there exists $b \in B$ such that $\phi(a,b) = 1$, and for every $b \in B$, there exists $a \in A$ such that $\phi(a,b) = 1$. But, since $\phi \leqslant \varphi$, it follows that φ is also a surjective $\mathcal{H}$-function, which contradicts the fact that φ is not a uniform fuzzy relation. □

According to the previous lemma, if the greatest λ-AFB is not a uniform fuzzy relation, then there exists no uniform λ-AFB. This means that, when developing an algorithm for computing the greatest uniform λ-AFB between two fuzzy automata, there is no need to impose additional conditions and we can only aim to compute the greatest λ-AFB between two fuzzy automata (as we do in Section 5), and then check if that fuzzy relation is a uniform fuzzy relation. If that is the case, then we choose such fuzzy relation to perform the results from the rest of this section. If the opposite holds, then we know that for chosen fuzzy automata there exists no uniform λ-AFB.

For approximate bisimulations on a single fuzzy automaton, the following holds.

Lemma 3.3. *For every fuzzy automaton $\mathcal{A}$ and $\lambda \in H$, there exists the greatest λ-AFB on $\mathcal{A}$ and it is a uniform fuzzy relation on $\mathcal{A}$.*

Proof. The proof directly follows from the fact that the greatest λ-AFB on $\mathcal{A}$ is a fuzzy equivalence on A (see [61]). □

The following theorem suggests that a (uniform) λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ generates λ-AFBs on both $\mathcal{A}$ and $\mathcal{A}'$.

Theorem 3.4. *For fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$, a fuzzy relation $\varphi \in H^{A\times A'}$ and $\lambda \in H$ we have:*

(a) *if φ is a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, then $\varphi \circ \varphi^{-1}$ is a λ-AFB on $\mathcal{A}$, while $\varphi^{-1} \circ \varphi$ is a λ-AFB on $\mathcal{A}'$;*
(b) *if φ is a uniform λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, then $\varphi \circ \varphi^{-1}$ is a fuzzy equivalence on A, while $\varphi^{-1} \circ \varphi$ is a fuzzy equivalence on A'.*

Proof. (a) Let $\varphi \in H^{A\times A'}$ be a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. Then, the following inequalities hold:

$$\begin{aligned}
&\lambda \wedge \sigma^A = \lambda^2 \wedge \sigma^A \leqslant \lambda \wedge \sigma^{A'} \circ \varphi^{-1} \leqslant \sigma^A \circ (\varphi \circ \varphi^{-1}),\\
&\lambda \wedge (\varphi \circ \varphi^{-1}) \circ \delta_x^A = \lambda^2 \wedge \varphi \circ (\varphi^{-1} \circ \delta_x^A) \leqslant \lambda \wedge (\varphi \circ \delta_x^{A'}) \circ \varphi^{-1} \leqslant \delta_x^A \circ (\varphi \circ \varphi^{-1}),\\
&\lambda \wedge (\varphi \circ \varphi^{-1}) \circ \tau^A = \lambda^2 \wedge \varphi \circ (\varphi^{-1} \circ \tau^A) \leqslant \lambda \wedge (\varphi \circ \tau^{A'}) \leqslant \tau^A.
\end{aligned} \tag{19}$$

This means that (10)–(15) hold on $\mathcal{A}$, from which it follows that $\varphi \circ \varphi^{-1}$ is a λ-AFB on $\mathcal{A}$. The following inequalities can be proved analogously:

$$\begin{aligned}
&\lambda \wedge \sigma^{A'} \leqslant \sigma^{A'} \circ (\varphi^{-1} \circ \varphi),\\
&\lambda \wedge (\varphi^{-1} \circ \varphi) \circ \delta_x^{A'} \leqslant \delta_x^{A'} \circ (\varphi^{-1} \circ \varphi),\\
&\lambda \wedge (\varphi^{-1} \circ \varphi) \circ \tau^{A'} \leqslant \tau^{A'}.
\end{aligned}$$

This means that (10)–(15) hold on $\mathcal{A}'$, from which it follows that $\varphi^{-1} \circ \varphi$ is a λ-AFB on $\mathcal{A}'$.

(b) According to Theorem 2.1, if φ is a uniform fuzzy relation, then $\varphi \circ \varphi^{-1} = \ker \varphi$ and $\varphi^{-1} \circ \varphi = \operatorname{coker} \varphi$. Since $\ker \varphi$ is a fuzzy equivalence on A, and $\operatorname{coker} \varphi$ is a fuzzy equivalence on A', the assertion of the theorem follows from the part (a). □

The following theorem gives a characterization of uniform λ-AFBs. This characterization allows us to define uniform λ-AFBs in an alternative way.

Theorem 3.5. *Let $\mathcal{A}$ and $\mathcal{A}'$ be two fuzzy automata and $\varphi \in H^{A\times A'}$ be a uniform fuzzy relation. Then, φ is a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ if and only if the following conditions hold, for every $x \in X$:*

$$E(\sigma^A \circ \varphi \circ \varphi^{-1}, \sigma^{A'} \circ \varphi^{-1}) \geqslant \lambda \tag{20}$$

$$E(\delta_x^A \circ \varphi \circ \varphi^{-1}, \varphi \circ \delta_x^{A'} \circ \varphi^{-1}) \geqslant \lambda, \tag{21}$$

$$E(\tau^A, \varphi \circ \tau^{A'}) \geqslant \lambda, \tag{22}$$

$$E(\sigma^A \circ \varphi, \sigma^{A'} \circ \varphi^{-1} \circ \varphi) \geqslant \lambda \tag{23}$$

$$E(\varphi^{-1} \circ \delta_x^A \circ \varphi, \delta_x^{A'} \circ \varphi^{-1} \circ \varphi) \geqslant \lambda, \tag{24}$$

$$E(\varphi^{-1} \circ \tau^A, \tau^{A'}) \geqslant \lambda. \tag{25}$$

Proof. Let φ be a uniform λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. Thus, (10)–(15) are valid. We prove the following:
Eqs. (20) and (23): From $\lambda \wedge \sigma^A \leqslant \sigma^{A'} \circ \varphi^{-1}$ it follows that:

$$\lambda \wedge \sigma^A \circ \varphi \circ \varphi^{-1} \leqslant \sigma^{A'} \circ \varphi^{-1} \circ \varphi \circ \varphi^{-1} = \sigma^{A'} \circ \varphi^{-1},$$

while from $\lambda \wedge \sigma^{A'} \leqslant \sigma^A \circ \varphi$ it follows that:

$$\lambda \wedge \sigma^{A'} \circ \varphi^{-1} \leqslant \sigma^A \circ \varphi \circ \varphi^{-1},$$

thus, we have that $\lambda \wedge \sigma^A \circ \varphi \circ \varphi^{-1} = \lambda \wedge \sigma^{A'} \circ \varphi^{-1}$, or equivalently, (20) follows. In addition, we have:

$$\lambda \wedge \sigma^A \circ \varphi = \lambda \wedge \sigma^A \circ \varphi \circ \varphi^{-1} \circ \varphi = \lambda \wedge \sigma^{A'} \circ \varphi^{-1} \circ \varphi,$$

which means that (23) holds.
Eqs. (21) and (24): From $\lambda \wedge \varphi \circ \delta_x^{A'} \leqslant \delta_x^A \circ \varphi$, we conclude:

$$\lambda \wedge \varphi \circ \delta_x^{A'} \circ \varphi^{-1} \leqslant \delta_x^A \circ \varphi \circ \varphi^{-1}.$$

On the other hand, from Theorem 3.4, we conclude that $\varphi \circ \varphi^{-1}$ is reflexive. Thus, from this fact and (19) we have:

$$\lambda \wedge \delta_x^A \circ \varphi \circ \varphi^{-1} \leqslant \lambda \wedge \varphi \circ \varphi^{-1} \circ \delta_x^A \circ \varphi \circ \varphi^{-1} \leqslant \varphi \circ \delta_x^{A'} \circ \varphi^{-1} \circ \varphi \circ \varphi^{-1} = \varphi \circ \delta_x^{A'} \circ \varphi^{-1},$$

from which (21) follows. Analogously, (24) can be proved.
Eqs. (22) and (25): From the reflexivity of $\varphi \circ \varphi^{-1}$ it follows that:

$$\lambda \wedge \tau^A \leqslant \lambda \wedge \varphi \circ \varphi^{-1} \circ \tau^A \leqslant \varphi \circ \tau^{A'},$$

while $\lambda \wedge \tau^{A'} \leqslant \tau^A$ follows from the definition of λ-AFBs. Thus, (22) follows. Analogously, (25) can be proved.

Conversely, let φ be a uniform fuzzy relation which satisfies (20) – (25). Then, from the reflexivity of $\varphi \circ \varphi^{-1}$ we conclude:

$$\lambda \wedge \sigma^A \leqslant \lambda \wedge \sigma^A \circ \varphi \circ \varphi^{-1} \leqslant \sigma^{A'} \circ \varphi^{-1},$$
$$\lambda \wedge \varphi^{-1} \circ \delta_x^A \leqslant \lambda \wedge \varphi^{-1} \circ \delta_x^A \circ \varphi \circ \varphi^{-1} \leqslant \delta_x^{A'} \circ \varphi^{-1} \circ \varphi \circ \varphi^{-1} = \delta_x^{A'} \circ \varphi^{-1},$$

while $\lambda \wedge \varphi^{-1} \circ \tau^A \leqslant \tau^{A'}$ immediately follows from (25). This means that (10)–(12) hold, thus φ is a λ-approximate forward simulation. Analogously, it can be proved that φ^{-1} is also a λ-approximate forward simulation. Therefore, φ is a λ-AFB. □

As stated by Theorem 3.4, if φ is a uniform λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, then it generates a fuzzy equivalence on $\mathcal{A}$ (according to which we can construct the factor fuzzy automaton of $\mathcal{A}$), and it generates a fuzzy equivalence on $\mathcal{A}'$ (according to which we can construct the factor fuzzy automaton of $\mathcal{A}'$). These two factor fuzzy automata can be connected via specific mathematical notion, which we introduce in the sequel. Namely, we call such a notion the λ-*approximate isomorphism*, since when $\lambda = 1$ we obtain the isomorphism between $\mathcal{A}$ and $\mathcal{A}'$.

Definition 3.6. Let $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A \rangle$ and $\mathcal{A}' = \langle A', \delta^{A'}, \sigma^{A'}, \tau^{A'} \rangle$ be fuzzy automata and let $\lambda \in H$. Then, a mapping $f : A \to A'$ is called λ-*approximate isomorphism* between $\mathcal{A}$ and $\mathcal{A}'$ if:

(a) f is a bijective mapping;
(b) for every $a_1, a_2 \in A$ and $x \in X$:

$$(\delta_x^A(a_1, a_2) \leftrightarrow \delta_x^{A'}(f(a_1), f(a_2))) \geqslant \lambda, \tag{26}$$

$$(\sigma^A(a_1) \leftrightarrow \sigma^{A'}(f(a_1))) \geqslant \lambda, \tag{27}$$

$$(\tau^A(a_1) \leftrightarrow \tau^{A'}(f(a_1))) \geqslant \lambda. \tag{28}$$

Theorem 3.7. *Let $\mathcal{A}$ and $\mathcal{A}'$ be fuzzy automata and $\varphi \in H^{A \times A'}$ be a uniform fuzzy relation. Then, φ is a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ if and only if the following conditions are satisfied:*

(a) $\ker \varphi$ *is λ-AFB on $\mathcal{A}$;*
(b) $\operatorname{coker} \varphi$ *is λ-AFB on $\mathcal{A}'$;*
(c) *if f_φ is a mapping defined by (9), then f_φ is a λ-approximate homomorphism between $\mathcal{A}/\ker\varphi$ and $\mathcal{A}'/\operatorname{coker}\varphi$.*

Proof. Denote $\theta = \ker \varphi$ and $\vartheta = \operatorname{coker} \varphi$. Consider the "only if" direction and suppose that φ is a uniform λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. Then, according to Theorems 2.1 and 3.4, the assertions (a) and (b) hold. Let us prove the assertion (c). Indeed, according to Lemma 2.2, the mapping f_φ is a bijective mapping from A/θ to A'/ϑ. Thus, it remains to prove that the following conditions hold for every $a \in A$:

$$(\sigma^{A/\theta}(\theta_a) \leftrightarrow \sigma^{A'/\vartheta}(f_\varphi(\theta_a))) \geqslant \lambda, \tag{29}$$

$$(\tau^{A/\theta}(\theta_a) \leftrightarrow \tau^{A'/\vartheta}(f_\varphi(\theta_a))) \geqslant \lambda, \tag{30}$$

$$(\delta_x^{A/\theta}(\theta_{a_1}, \theta_{a_2}) \leftrightarrow \delta_x^{A'/\vartheta}(f_\varphi(\theta_{a_1}), f_\varphi(\theta_{a_2}))) \geqslant \lambda, \quad \text{for every } x \in X. \tag{31}$$

Indeed, for any $\lambda \in H$ and $a \in A$, from (20) it follows that:

$$\lambda \wedge \sigma^{A/\theta}(\theta_a) = \lambda \wedge (\sigma^A \circ \theta)(a) = \lambda \wedge (\sigma^A \circ \varphi \circ \varphi^{-1})(a) = \lambda \wedge (\sigma^{A'} \circ \varphi^{-1})(a) = \lambda \wedge \bigvee_{a' \in A'} \sigma^{A'}(a') \wedge \varphi(a, a').$$

According to Theorem 2.1(f), this means that, for an arbitrary $f \in CR(\varphi)$, we have that:

$$\begin{aligned}\lambda \wedge \sigma^{A/\theta}(\theta_a) &= \lambda \wedge \bigvee_{a' \in A'} \sigma^{A'}(a') \wedge \vartheta(f(a), a') = \lambda \wedge \bigvee_{a' \in A'} \sigma^{A'}(a') \wedge \vartheta(a', f(a)) = \lambda \wedge (\sigma^{A'} \circ \vartheta)(f(a)) \\ &= \lambda \wedge \sigma^{A'/\vartheta}(\vartheta_{f(a)}) = \lambda \wedge \sigma^{A'/\vartheta}(f_\varphi(\theta_a)).\end{aligned}$$

This means that (29) is valid. The assertion (30) can be proved analogously. In addition, by the reflexivity of θ, we have:

$$\lambda \wedge \delta_x^A \circ \theta \leqslant \lambda \wedge \theta \circ \delta_x^A \circ \theta,$$

while from the fact that φ is a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, which satisfies Theorem 2.1, we have:

$$\begin{aligned}\lambda \wedge \theta \circ \delta_x^A \circ \theta &= \lambda \wedge \varphi \circ \varphi^{-1} \circ \delta_x^A \circ \varphi \circ \varphi^{-1} \leqslant \lambda \wedge \varphi \circ \delta_x^{A'} \circ \varphi^{-1} \circ \varphi \circ \varphi^{-1} = \lambda \wedge \varphi \circ \delta_x^{A'} \circ \varphi^{-1} \\ &\leqslant \lambda \wedge \delta_x^A \circ \varphi \circ \varphi^{-1} = \lambda \wedge \delta_x^A \circ \theta.\end{aligned}$$

Thus, we have proved that $\lambda \wedge \delta_x^A \circ \theta = \lambda \wedge \theta \circ \delta_x^A \circ \theta = \lambda \wedge \varphi \circ \delta_x^{A'} \circ \varphi^{-1}$. From this and Theorem 2.1(f) we obtain that, for every $a_1, a_2 \in A$:

$$\lambda \wedge \delta^{A/\theta}(\theta_{a_1}, x, \theta_{a_2}) = \lambda \wedge (\theta \circ \delta_x^A \circ \theta)(a_1, a_2) = \lambda \wedge (\varphi \circ \delta_x^{A'} \circ \varphi^{-1})(a_1, a_2)$$

$$
\begin{aligned}
&= \lambda \wedge \bigvee_{a_1', a_2' \in A'} \varphi(a_1, a_1') \wedge \delta_x^{A'}(a_1', a_2') \wedge \varphi(a_2, a_2') \\
&= \lambda \wedge \bigvee_{a_1', a_2' \in A'} \vartheta(f(a_1), a_1') \wedge \delta_x^{A'}(a_1', a_2') \wedge \vartheta(f(a_2), a_2') \\
&= \lambda \wedge (\vartheta \circ \delta_x^{A'} \circ \vartheta)(f(a_1), f(a_2)) \\
&= \lambda \wedge \delta^{A'/\vartheta}(\vartheta_{f(a_1)}, x, \vartheta_{f(a_2)}),
\end{aligned}
$$

and we conclude that (31) holds. Thus, f_φ is a λ-approximate homomorphism between $\mathcal{A}/\theta$ and $\mathcal{A}'/\vartheta$.

Conversely, consider the "if" direction and suppose that the conditions (a)–(c) hold. Let $g \in CR(\varphi), h \in CR(\varphi^{-1}), a, a_1, a_2 \in A$, $a', a_1', a_2' \in A'$ and $x \in X$. Then, we have:

$$
\begin{aligned}
\lambda \wedge \sigma^A(a) &\leqslant \lambda \wedge (\sigma^A \circ \theta)(a) = \lambda \wedge \sigma^{A/\theta}(\theta_a) = \lambda \wedge \sigma^{A'/\vartheta}(f_\varphi(\theta_a)) = \lambda \wedge \sigma^{A'/\vartheta}(\vartheta_{g(a)}) \\
&= \lambda \wedge (\sigma^{A'} \circ \vartheta)(g(a)) = \lambda \wedge \bigvee_{a' \in A'} \sigma^{A'}(a') \wedge \vartheta(a', g(a)) = \lambda \wedge \bigvee_{a' \in A'} \sigma^{A'}(a') \wedge \vartheta(g(a), a') \\
&= \lambda \wedge \bigvee_{a' \in A'} \sigma^{A'}(a') \wedge \varphi(a, a') = \lambda \wedge \bigvee_{a' \in A'} \sigma^{A'}(a') \wedge \varphi^{-1}(a', a) = \lambda \wedge (\sigma^{A'} \circ \varphi^{-1})(a) \\
&\leqslant (\sigma^{A'} \circ \varphi^{-1})(a).
\end{aligned}
$$

Similarly, it can be shown that $\lambda \wedge \sigma^B \leqslant \sigma^A \circ \varphi$. Next, we have:

$$
\begin{aligned}
\lambda \wedge (\theta \circ \delta_x^A \circ \theta)(a_1, a_2) &= \lambda \wedge \delta^{A/\theta}(\theta_{a_1}, x, \theta_{a_2}) = \lambda \wedge \delta^{A'/\vartheta}(f_\varphi(\theta_{a_1}), x, f_\varphi(\theta_{a_2})) \\
&= \lambda \wedge \delta^{A'/\vartheta}(\vartheta_{g(a_1)}, x, \vartheta_{g(a_2)}) = \lambda \wedge (\vartheta \circ \delta_x^{A'} \circ \vartheta)(g(a_1), g(a_2)).
\end{aligned}
$$

And analogously, it can be shown that:

$$
\lambda \wedge (\vartheta \circ \delta_x^{A'} \circ \vartheta)(a_1', a_2') = \lambda \wedge (\theta \circ \delta_x^A \circ \theta)(h(a_1), h(a_2)).
$$

From (a) and (b) it follows that:

$$
\begin{aligned}
\lambda \wedge (\varphi^{-1} \circ \delta_x^A)(a', a) &\leqslant \lambda \wedge (\varphi^{-1} \circ \theta \circ \delta_x^A)(a', a) \leqslant \lambda \wedge (\varphi^{-1} \circ \delta_x^A \circ \theta)(a', a) \\
&= \lambda \wedge \bigvee_{a_1 \in A} \varphi^{-1}(a', a_1) \wedge (\delta_x^A \circ \theta)(a_1, a) = \\
&= \lambda \wedge \bigvee_{a_1 \in A} \theta(h(a'), a_1) \wedge (\delta_x^A \circ \theta)(a_1, a) = \lambda \wedge (\theta \circ \delta_x^A \circ \theta)(h(a'), a) \\
&= \lambda \wedge (\vartheta \circ \delta_x^{A'} \circ \vartheta)(g(h(a')), g(a)) = \lambda \wedge \bigvee_{a_1' \in A'} \vartheta(g(h(a')), a_1') \wedge (\delta_x^{A'} \circ \vartheta)(a_1', g(a)) \\
&= \lambda \wedge \bigvee_{a_1' \in A'} \vartheta(a', a_1') \wedge (\delta_x^{A'} \circ \vartheta)(a_1', g(a)) = \lambda \wedge (\vartheta \circ \delta_x^{A'} \circ \vartheta)(a', g(a)) \\
&= \lambda \wedge (\delta_x^{A'} \circ \vartheta)(a', g(a)) = \lambda \wedge \bigvee_{a_2' \in A'} \delta_x^{A'}(a', a_2') \wedge \vartheta(a_2', g(a)) \\
&= \lambda \wedge \bigvee_{a_2' \in A'} \delta_x^{A'}(a', a_2') \wedge \varphi(a, a_2') = \lambda \wedge (\delta_x^{A'} \circ \varphi^{-1})(a', a) \leqslant (\delta_x^{A'} \circ \varphi^{-1})(a', a).
\end{aligned}
$$

Therefore, $\lambda \wedge \varphi^{-1} \circ \delta_x^A \leqslant \delta_x^{A'} \circ \varphi^{-1}$. And analogously, it can be shown that $\lambda \wedge \varphi \circ \delta_x^{A'} \leqslant \delta_x^A \circ \varphi$. We also have:

$$
\lambda \wedge \tau^A(a) \leqslant \lambda \wedge (\theta \circ \tau^A)(a) = \lambda \wedge \tau^{A/\theta}(\theta_a) = \lambda \wedge \tau^{A'/\vartheta}(f_\varphi(\theta_a)) = \lambda \wedge \tau^{A'/\vartheta}(\vartheta_{g(a)})
$$

$$= \lambda \wedge \bigvee_{a' \in A'} \vartheta(g(a), a') \wedge \tau^{A'}(a') = \lambda \wedge \bigvee_{a' \in A'} \varphi(a, a') \wedge \tau^{A'}(a') = \lambda \wedge (\varphi \circ \tau^{A'})(a')$$
$$\leqslant (\varphi \circ \tau^{A'})(a').$$

Analogously, it can be shown that $\lambda \wedge \varphi^{-1} \circ \tau^{A} \leqslant \tau^{A'}$. Therefore, φ is a λ-AFB. □

As the previous theorem suggests, given fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$, uniform λ-AFB allows us to construct appropriate factor fuzzy automata of $\mathcal{A}$ and $\mathcal{A}'$ that are mutually isomorphic to some extent. An equally important property is proven in the following theorem. Namely, if each of fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$ has a λ-AFB on its own set of states, then we can establish a uniform λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ which gives a connection between these three λ-AFBs. This uniform λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ induces an approximate isomorphism between the corresponding factor fuzzy automata, and vice versa.

Theorem 3.8. *Let $\lambda \in H$ and let $\mathcal{A}$ and $\mathcal{A}'$ be fuzzy automata, $\theta \in H^{A \times A}$ a λ-AFB on $\mathcal{A}$, and $\vartheta \in H^{A' \times A'}$ a λ-AFB on $\mathcal{A}'$. Then, the following conditions are equivalent:*

(a) *there exists a uniform λ-AFB $\varphi \in H^{A \times A'}$ between $\mathcal{A}$ and $\mathcal{A}'$ such that*

$$\ker \varphi = \theta \quad \textit{and} \quad \operatorname{coker} \varphi = \vartheta; \tag{32}$$

(b) *there exists a λ-approximate isomorphism $f : A/\theta \to A'/\vartheta$ such that, for every $a_1, a_2 \in A$,*

$$\widetilde{\theta}(\theta_{a_1}, \theta_{a_2}) = \widetilde{\vartheta}(f(\theta_{a_1}), f(\theta_{a_2})). \tag{33}$$

Proof. Assume that (a) holds and choose some $h \in CR(\varphi)$. Then, Lemmas 2.2 and 2.4 yield:

$$\widetilde{\theta}(\theta_{a_1}, \theta_{a_2}) = \theta(a_1, a_2) = \vartheta(h(a_1), h(a_2)) = \widetilde{\vartheta}(\vartheta_{h(a_1)}, \vartheta_{h(a_2)}) = \widetilde{\vartheta}(f_\varphi(\theta_{a_1}), f_\varphi(\theta_{a_2})).$$

From Theorem 3.7, it follows that f_φ is a λ-approximate isomorphism between A/θ and A'/ϑ satisfying (33), thus (b) follows.

Conversely, assume that (b) holds. Now, define functions $f_\theta : A \to A/\theta$ and $f_\vartheta : A'/\vartheta \to A'$ as follows:

$$f_\theta(a) = \theta_a, \quad \text{for every } a \in A;$$
$$f_\vartheta(\vartheta_{a'}) = \widehat{a'}, \quad \text{for every } a' \in A', \text{ where } \widehat{a'} \text{ is a fixed element from } \widehat{\vartheta_{a'}}.$$

Now, define a function $g : A \to A'$ as $g = f_\theta \circ f \circ f_\vartheta$ (recall that f is a λ-approximate isomorphism $f : A/\theta \to A'/\vartheta$ satisfying (33)). Define a fuzzy relation $\varphi \in H^{A \times A'}$ as:

$$\varphi(a, a') = \vartheta(g(a), a'), \quad \text{for every } a \in A \text{ and } a' \in A'.$$

By Lemma 2.4, φ is a uniform fuzzy relation satisfying (32) and $g \in CR(\varphi)$. For every $a \in A$, there exists some $a' \in A'$ such that $f(\theta_a) = \vartheta_{a'}$, from which we have:

$$g(a) = f_\vartheta(f(f_\theta(a))) = f_\vartheta(f(\theta_a)) = f_\vartheta(\vartheta_{a'}) \in \widehat{\vartheta_{a'}},$$

which means that:

$$f_\varphi(\theta_a) = \vartheta_{g(a)} = \vartheta_{a'} = f(\theta_a).$$

The above equalities hold for every $a \in A$, hence $f_\varphi = f$. From Theorem 3.7, we conclude that φ is a uniform λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, thus (a) follows. □

4. Heterotypic approximate bisimulations

This section is devoted to studying λ-ABFBs. Dual results can be obtained for λ-AFBBs in a similar way and are omitted. The first difference between homotypic λ-approximate bisimulations and heterotypic ones can be seen in Theorem 3.1, because it does not hold in such a formulation for heterotypic λ-approximate bisimulations. Namely, we cannot prove that φ is a partial fuzzy function when φ is a λ-ABFB. Thus, only the following holds, which is exactly [61, Theorem III.6].

Theorem 4.1. *For fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$, if there exists at least one λ-ABFB between $\mathcal{A}$ and $\mathcal{A}'$, then there exists the greatest λ-ABFB between $\mathcal{A}$ and $\mathcal{A}'$.*

Also, a result analogous to Lemma 3.2 also holds for λ-ABFBs. In the sequel, we provide some additional properties of λ-ABFBs, analogous to the one provided for λ-AFBs in the previous section. But first, we state a simple lemma which is needed in the sequel.

Lemma 4.2. *Let $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A \rangle$ be a fuzzy automaton and χ a fuzzy equivalence on A. Then:*

(a) *χ is a λ-AFB on $\mathcal{A}$ if and only if:*

$$S(\chi \circ \delta_x^A \circ \chi, \delta_x^A \circ \chi) \geqslant \lambda; \tag{34}$$
$$S(\chi \circ \tau, \tau) \geqslant \lambda. \tag{35}$$

(b) *χ is a λ-ABB on $\mathcal{A}$ if and only if:*

$$S(\chi \circ \delta_x^A \circ \chi, \chi \circ \delta_x^A) \geqslant \lambda; \tag{36}$$
$$S(\sigma \circ \chi, \sigma) \geqslant \lambda. \tag{37}$$

Proof. We prove only the part (a). Assume that a fuzzy equivalence $\chi \in H^{A \times A}$ is a λ-AFB on $\mathcal{A}$. Then (35) follows immediately from (15), while from (14) and the fact that χ is reflexive we have $\lambda \wedge \chi \circ \delta_x^A \circ \chi \leqslant \delta_x^A \circ \chi \circ \chi = \delta_x^A \circ \chi$, thus (34) follows. Conversely, if (34) and (35) hold, then it follows from (34) that $\lambda \wedge \chi \circ \delta_x^A \leqslant \lambda \wedge \chi \circ \delta_x^A \circ \chi \leqslant \delta_x^A \circ \chi$, which implies (14), and (15) follows from (35). Note that (13) holds since χ is reflexive, and (10) – (12) follow from (13) – (15) since $\chi = \chi^{-1}$. □

The following two theorems are analogous results to Theorems 3.7 and 3.8 for the case of heterogeneous approximate bisimulations. These results establish a full distinction between homogeneous and heterogeneous approximate bisimulations.

Theorem 4.3. *Let $\mathcal{A}$ and $\mathcal{A}'$ be fuzzy automata and $\varphi \in H^{A \times A'}$ a uniform fuzzy relation. Then, φ is a λ-ABFB between $\mathcal{A}$ and $\mathcal{A}'$ if and only if the following conditions are satisfied:*

(a) $\ker \varphi$ *is a λ-AFB on $\mathcal{A}$;*
(b) $\operatorname{coker} \varphi$ *is a λ-ABB on $\mathcal{A}'$;*
(c) *if f_φ is a mapping defined by (9), then f_φ is a λ-approximate homomorphism between $\mathcal{A}/\ker \varphi$ and $\mathcal{A}'/\operatorname{coker} \varphi$.*

Proof. Denote again $\theta = \ker \varphi$ and $\vartheta = \operatorname{coker} \varphi$. Consider the "only if" direction and let φ be a uniform fuzzy relation between $\mathcal{A}$ and $\mathcal{A}'$. Recall again that θ and ϑ are fuzzy equivalences on A and A', respectively, such that $\theta = \varphi \circ \varphi^{-1}$ and $\vartheta = \varphi^{-1} \circ \varphi$.

First, assume that φ is a λ-ABFB. Then, we have:

$$\begin{aligned}
\lambda \wedge \theta \circ \delta_x^A \circ \theta &= \lambda \wedge \varphi \circ \varphi^{-1} \circ \delta_x^A \circ \varphi \circ \varphi^{-1} \leqslant \lambda \wedge \varphi \circ \varphi^{-1} \circ \varphi \circ \delta_x^{A'} \circ \varphi^{-1} = \lambda \wedge \varphi \circ \delta_x^{A'} \circ \varphi^{-1} \\
&\leqslant \lambda \wedge \delta_x^A \circ \varphi \circ \varphi^{-1} = \lambda \wedge \delta_x^A \circ \theta \leqslant \delta_x^A \circ \theta, \\
\lambda \wedge \theta \circ \tau^A &= \lambda \wedge \varphi \circ \varphi^{-1} \circ \tau^A \leqslant \lambda \wedge \varphi \circ \varphi^{-1} \circ \varphi \circ \tau^{A'} = \lambda \wedge \varphi \circ \tau^{A'} = \tau^A.
\end{aligned}$$

In other words, (34) and (35) hold. According to the Lemma 4.2, θ is a λ-AFB on $\mathcal{A}$, thus the assertion (a) follows. Analogously as for (a), it can be shown that $\lambda \wedge \vartheta \circ \delta_x^{A'} \circ \vartheta \leqslant \vartheta \circ \delta_x^{A'}$ and $\lambda \wedge \sigma^{A'} \circ \vartheta \leqslant \sigma^{A'}$, and thus ϑ is a λ-ABB on $\mathcal{A}'$, thus (b) follows. The assertion (c) can be proved analogously as in the proof of Theorem 3.7.

The proof of the "if" direction is similar to the one for the proof of Theorem 3.7, but again using Lemma 4.2. □

Theorem 4.4. *Let $\lambda \in H$ and let $\mathcal{A}$ and $\mathcal{A}'$ be fuzzy automata, $\theta \in H^{A\times A}$ a λ-AFB on $\mathcal{A}$, and $\vartheta \in H^{A'\times A'}$ a λ-ABB on $\mathcal{A}'$. Then, the following conditions are equivalent:*

(a) *there exists a uniform λ-ABFB $\varphi \in H^{A\times A'}$ between $\mathcal{A}$ and $\mathcal{A}'$ such that*

$$\ker \varphi = \theta \quad \textit{and} \quad \operatorname{coker} \varphi = \vartheta;$$

(b) *there exists a λ-approximate isomorphism $f : A/\theta \to A'/\vartheta$ such that, for every $a_1, a_2 \in A$,*

$$\widetilde{\theta}(\theta_{a_1}, \theta_{a_2}) = \widetilde{\vartheta}(f(\theta_{a_1}), f(\theta_{a_2})).$$

The proof is omitted since it is similar to the proof of Theorem 3.8.

5. Computing approximate forward bisimulations for fuzzy automata over the Gödel structure

In this section, we adapt the algorithm developed in [48] to obtain a new one for computing the greatest λ-AFB between finite fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$ when $\mathcal{H}$ is the Gödel structure, if it exists. We also specify an algorithm for computing the greatest $\lambda \in [0,1]$ such that there exists a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$.

Let $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A\rangle$ and $\mathcal{A}' = \langle A', \delta^{A'}, \sigma^{A'}, \tau^{A'}\rangle$ be finite fuzzy automata over an alphabet X. Let i, f, i' and f' be new elements, not belonging to A and A'. Our intention is to use i (respectively, f) as the new unique initial (respectively, terminal) state for $\mathcal{A}$. Similarly, i' (respectively, f') is intended to be used as the new unique initial (respectively, terminal) state for $\mathcal{A}'$. We denote $A_\oplus = A \cup \{i, f\}$ and $A'_\oplus = A' \cup \{i', f'\}$. For $x \in X$, we define the fuzzy relations $E_x : A_\oplus \times A_\oplus \to H$ and $E'_x : A'_\oplus \times A'_\oplus \to H$ as follows:

$$E_x(a,b) = \begin{cases} \delta_x^A(a,b) & \text{if } a,b \in A,\\ \sigma^A(b) & \text{if } a = i \text{ and } b \in A,\\ \tau^A(a) & \text{if } a \in A \text{ and } b = f,\\ 0 & \text{otherwise.}\end{cases} \qquad E'_x(a',b') = \begin{cases} \delta_x^{A'}(a',b') & \text{if } a',b' \in A',\\ \sigma^{A'}(b') & \text{if } a' = i' \text{ and } b' \in A',\\ \tau^{A'}(a') & \text{if } a' \in A' \text{ and } b' = f',\\ 0 & \text{otherwise.}\end{cases}$$

Lemma 5.1. *Given $\varphi : A \times A' \to H$, let $Z : A_\oplus \times A'_\oplus \to H$ be defined as follows: if $\langle a, a'\rangle \in A \times A'$, then $Z(a,a') = \varphi(a,a')$, else if $\langle a, a'\rangle = \langle i, i'\rangle$ or $\langle a, a'\rangle = \langle f, f'\rangle$, then $Z(a,a') = 1$, else $Z(a,a') = 0$. Then, φ is a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ iff the following two conditions hold for all $x \in X$:*

$$\lambda \wedge E_x^{-1} \circ Z \leqslant Z \circ {E'_x}^{-1} \tag{38}$$

$$\lambda \wedge Z \circ E'_x \leqslant E_x \circ Z. \tag{39}$$

Note that (38) is equivalent to

$$\lambda \wedge Z^{-1} \circ E_x \leqslant E'_x \circ Z^{-1}. \tag{40}$$

Roughly speaking, (40) is similar to (11) and covers (10)–(12), whereas (39) is similar to (14) and covers (13)–(15).

Proof. Suppose φ satisfies the conditions (10)–(12). We prove that Z satisfies the condition (38). Let $b \in A_\oplus$ and $a' \in A'_\oplus$. We need to prove that $\lambda \wedge (E_x^{-1} \circ Z)(b,a') \leqslant (Z \circ {E'_x}^{-1})(b,a')$. It is sufficient to show that, for every $a \in A_\oplus$, $\lambda \wedge E_x(a,b) \wedge Z(a,a') \leqslant (Z \circ {E'_x}^{-1})(b,a')$. If $E_x(a,b) = 0$ or $Z(a,a') = 0$, then the inequality clearly holds. Suppose that $E_x(a,b) > 0$ and $Z(a,a') > 0$. There are the following cases.

- Case $a, b \in A$ and $a' \in A'$: We have $E_x(a,b) = \delta_x^A(a,b)$ and $Z(a,a') = \varphi(a,a')$. Thus, $\lambda \wedge E_x(a,b) \wedge Z(a,a') \leqslant (\lambda \wedge \varphi^{-1} \circ \delta_x^A)(a',b)$. By (11), it follows that $\lambda \wedge E_x(a,b) \wedge Z(a,a') \leqslant (\delta_x^{A'} \circ \varphi^{-1})(a',b)$. Since $(\delta_x^{A'} \circ \varphi^{-1})(a',b) = (\varphi \circ (\delta_x^{A'})^{-1})(b,a') = (Z \circ {E'_x}^{-1})(b,a')$, we can derive that $\lambda \wedge E_x(a,b) \wedge Z(a,a') \leqslant (Z \circ {E'_x}^{-1})(b,a')$.

- Case $a = i, b \in A$ and $a' = i'$: We have $E_x(a,b) = \sigma^A(b)$ and $Z(a,a') = 1$. Thus, $\lambda \wedge E_x(a,b) \wedge Z(a,a') = (\lambda \wedge \sigma^A)(b)$. By (10), it follows that $\lambda \wedge E_x(a,b) \wedge Z(a,a') \leqslant (\sigma^{A'} \circ \varphi^{-1})(b)$. Since $(\sigma^{A'} \circ \varphi^{-1})(b) = (\varphi \circ \sigma^{A'})(b) = (Z \circ {E'_x}^{-1})(b,a')$, we can derive that $\lambda \wedge E_x(a,b) \wedge Z(a,a') \leqslant (Z \circ {E'_x}^{-1})(b,a')$.

- Case $a \in A$, $b = f$ and $a' \in A'$: We have $E_x(a,b) = \tau^A(a)$ and $Z(a,a') = \varphi(a,a')$. Thus, $\lambda \wedge E_x(a,b) \wedge Z(a,a') = \lambda \wedge \varphi^{-1}(a',a) \wedge \tau^A(a) \leqslant (\lambda \wedge \varphi^{-1} \circ \tau^A)(a')$. By (12), it follows that $\lambda \wedge E_x(a,b) \wedge Z(a,a') \leqslant \tau^{A'}(a')$. Since $\tau^{A'}(a') = Z(b,f') \wedge E'_x(a',f') \leqslant (Z \circ {E'_x}^{-1})(b,a')$, we can derive that $\lambda \wedge E_x(a,b) \wedge Z(a,a') \leqslant (Z \circ {E'_x}^{-1})(b,a')$.

We have proved that (10)–(12) imply (38) and therefore also (40). By duality, it can be seen that (13)–(15) imply (39). The proof is similar.

Suppose Z satisfies the condition (40). We prove that φ satisfies the conditions (10)–(12).

Consider the condition (10) and let $a \in A$. We need to prove that $\lambda \wedge \sigma^A(a) \leqslant (\sigma^{A'} \circ \varphi^{-1})(a)$. Let x be an arbitrary element of X. We have $\lambda \wedge \sigma^A(a) = \lambda \wedge Z^{-1}(i',i) \wedge E_x(i,a) \leqslant \lambda \wedge (Z^{-1} \circ E_x)(i',a)$. By (40), it follows that $\lambda \wedge \sigma^A(a) \leqslant (E'_x \circ Z^{-1})(i',a) = (\sigma^{A'} \circ \varphi^{-1})(a)$.

Consider the condition (11). Let $a' \in A'$ and $b \in A$. We need to prove that $\lambda \wedge (\varphi^{-1} \circ \delta_x^A)(a',b) \leqslant (\delta_x^{A'} \circ \varphi^{-1})(a',b)$. We have $\lambda \wedge (\varphi^{-1} \circ \delta_x^A)(a',b) \leqslant \lambda \wedge (Z^{-1} \circ E_x)(a',b)$. By (40), it follows that $\lambda \wedge (\varphi^{-1} \circ \delta_x^A)(a',b) \leqslant (E'_x \circ Z^{-1})(a',b)$. Observe that $(E'_x \circ Z^{-1})(a',b) = (\delta_x^{A'} \circ \varphi^{-1})(a',b)$. Hence, $\lambda \wedge (\varphi^{-1} \circ \delta_x^A)(a',b) \leqslant (\delta_x^{A'} \circ \varphi^{-1})(a',b)$.

Consider the condition (12) and let $a' \in A'$. We need to prove that $\lambda \wedge (\varphi^{-1} \circ \tau^A)(a') \leqslant \tau^{A'}(a')$. We have $\lambda \wedge (\varphi^{-1} \circ \tau^A)(a') \leqslant (Z^{-1} \circ E_x)(a',f)$. By (40), it follows that $\lambda \wedge (\varphi^{-1} \circ \tau^A)(a') \leqslant (E'_x \circ Z^{-1})(a',f) = \tau^{A'}(a')$.

We have proved that (40) implies (10)–(12). By duality, it can be seen that (39) implies (13)–(15). The proof is similar. □

Corollary 5.2. *Suppose that $Z : A_\oplus \times A'_\oplus \to H$ is the greatest fuzzy relation satisfying the conditions (38) and (39) (for all $x \in X$) as well as the following ones:*

$$Z(i,a') = 0 \text{ for all } a' \in A' \setminus \{i'\}, \tag{41}$$
$$Z(f,a') = 0 \text{ for all } a' \in A' \setminus \{f'\}, \tag{42}$$
$$Z(a,i') = 0 \text{ for all } a \in A \setminus \{i\}, \tag{43}$$
$$Z(a,f') = 0 \text{ for all } a \in A \setminus \{f\}. \tag{44}$$

Let $\varphi : A \times A' \to H$ be the restriction of Z to $A \times A'$. If $Z(i,i') = 1$, then φ is the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. If $Z(i,i') < 1$, then the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ does not exist.

Proof. We must have $Z(f,f') = 1$, because Z is the greatest one and increasing $Z(f,f')$ does not affect the other requirements about Z.

Suppose $Z(i,i') = 1$. Thus, by Lemma 5.1, φ is a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. To show that φ is the greatest one, let φ_2 be a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ such that $\varphi_2 \geqslant \varphi$. We show that $\varphi_2 = \varphi$. Let $Z_2 : A_\oplus \times A'_\oplus \to H$ be defined as Z in Lemma 5.1 when using φ_2 instead of φ. Since $\varphi_2 \geqslant \varphi$, we have $Z_2 \geqslant Z$. On the other hand, by Lemma 5.1, Z_2 satisfies the conditions (38) and (39) with Z replaced by Z_2 (for all $x \in X$), and therefore, $Z_2 \leqslant Z$. Hence, $Z_2 = Z$, which implies that $\varphi_2 = \varphi$.

Consider the case where $Z(i,i') < 1$. For a contradiction, suppose that φ_2 is the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. Let $Z_2 : A_\oplus \times A'_\oplus \to H$ be defined as Z in Lemma 5.1 when using φ_2 instead of φ. We have $Z_2(i,i') = 1$ and, by Lemma 5.1, $Z_2 \leqslant Z$. These contradict the assumption $Z(i,i') < 1$. □

The algorithm ComputeFuzzyBisimulation developed in [48] constructs the greatest fuzzy bisimulation between two finite fuzzy interpretations $\mathcal{I}$ and $\mathcal{I}'$ in the fuzzy description logic $f\mathcal{ALC}$ under the Gödel

semantics. We adapt this algorithm to obtain a new algorithm, called ComputeAFB, for computing the greatest λ-AFB between finite fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$ when $\mathcal{H}$ is the Gödel structure (if it exists) as follows:

- The initialization Initialize$(\mathcal{I}, \mathcal{I}')$ is replaced by InitializeAFB$(\mathcal{A}, \mathcal{A}')$, where this latter procedure is obtained from the former as follows:
  - $\mathbf{R}$, $\Delta^{\mathcal{I}}$ and $\Delta^{\mathcal{I}'}$ are replaced by X, $A_{\oplus}$ and $A'_{\oplus}$, respectively;
  - the statements 1–3 of Initialize$(\mathcal{I}, \mathcal{I}')$ with the following contents

    for each $x \in \Delta^{\mathcal{I}}$ *and* $x' \in \Delta^{\mathcal{I}'}$*:*
    $Z[x, x'] := \min\{B^{\mathcal{I}}(x) \Leftrightarrow B^{\mathcal{I}'}(x') \mid B \in \mathbf{C}\}$*;*
    done$Z[x, x'] :=$ *false;*

    are replaced by the following ones:

    for each $a \in A_{\oplus}$ *and* $a' \in A'_{\oplus}$*:*
    if $a \in A$ *and* $a' \in A'$ *then* $Z[a, a'] := 1$ *else* $Z[a, a'] := 0$*;*
    done$Z[a, a'] :=$ *false;*
    $Z[i, i'] := 1$, $Z[f, f'] := 1$*;*

- The main loop is modified by adding to the place after the statement

  choose a tuple t *with a smallest weight*(t) *from* $\mathbb{Q}$*;*

  the following statement

  if weight$(t) \geqslant \lambda$ *then break;*

- The statement "*return* Z" is replaced by the following statement:

  if $Z[i, i'] = 1$ *then return the restriction of* Z *to* $A \times A'$*, else terminate with the information that the greatest* λ*-AFB between* $\mathcal{A}$ *and* $\mathcal{A}'$ *over the Gödel structure does not exist;*

A full specification of the algorithm ComputeAFB is provided in the Supplemental Material together with its explanations and a proof of the following theorem, which exploits Corollary 5.2.

Theorem 5.3. *The algorithm* ComputeAFB *always terminates and returns a correct result. It can be implemented to run in time* $O((m + n)n)$*, where* n *is the number of states and* m *is the number of non-zero transitions of the given finite fuzzy automata.*

We have implemented the algorithm ComputeAFB in C++ and shared the codes publicly [45].

Consider the problem of computing the greatest $\lambda \in [0, 1]$ such that there exists a λ-AFB between finite fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$, which is equivalent to that there exists the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ (by Theorem 3.1). We now specify a new algorithm, named ComputeAFBD (*compute AFB degree*), for this task. It is obtained from modifying the algorithm ComputeFuzzyBisimulation [48] as follows:

- The initialization Initialize$(\mathcal{I}, \mathcal{I}')$ is replaced by InitializeAFB$(\mathcal{A}, \mathcal{A}')$, which was specified above.

- The main loop is modified by adding to the place after the statement

  choose a tuple t *with a smallest weight*(t) *from* $\mathbb{Q}$*;*

  the following statement

  if $t = \langle i, i' \rangle_Z$ *then break;*

- The statement "*return* Z" is replaced by "*return* $Z[i, i']$".

A full specification of the algorithm ComputeAFBD together with a proof of the following theorem is provided in the Supplemental Material. The theorem states that, given finite fuzzy automata $\mathcal{A}$ and $\mathcal{A}'$ over the Gödel structure, the greatest $\lambda \in [0,1]$ such that there is a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ exists and can be efficiently computed.

Theorem 5.4. *The algorithm* ComputeAFBD *always terminates and returns a correct result. It can be implemented to run in time* $O((m+n)n)$*, where* n *is the number of states and* m *is the number of non-zero transitions of the given finite fuzzy automata.*

6. Concluding remarks

In [61], the authors introduced the notions of approximate simulation and bisimulation for fuzzy automata over Heyting algebras. They considered them as fuzzy relations defined between the sets of states of two fuzzy automata. In this paper, we studied particular types of such approximate bisimulations, namely those which are uniform fuzzy relations. The main advantage of uniform approximate bisimulations when compared to approximate bisimulations which are not uniform fuzzy relations is that the former ones can be employed to reduce both fuzzy automata in such a manner that there exists a relaxed type of isomorphism between these two reduced fuzzy automata (cf. Theorem 3.7). Moreover, this kind of isomorphism on reduced fuzzy automata induces a uniform approximate bisimulation on the original fuzzy automata. This is of a particular interest for very large fuzzy automata, since they can be reduced to fuzzy automata with a much smaller number of states, while keeping their structures to a certain extent. In addition, uniform forward bisimulations can be defined in the alternative way given by the conditions in Theorem 3.5. These conditions can be used to define approximate forward bisimulations for fuzzy automata over structures other than Heyting algebras, which will be the topic of our future research.

We paid special attention to approximate bisimulations for fuzzy automata defined over the Gödel structure. In particular, we adapted the algorithm developed in [48] to obtain new and efficient algorithms for computing the greatest λ-approximate forward bisimulation between finite fuzzy automata defined over the Gödel structure, if it exists for a given $\lambda \in [0,1]$, as well as for computing the greatest $\lambda \in [0,1]$ which guarantees that existence. We presented only the differences with respect to the algorithm developed in [48], while the detailed analysis and an illustrative example are given in the Supplemental Material. By Lemma 3.2, our algorithms can easily be adapted to work for approximate uniform forward bisimulations, which was our original attention. It is challenging to provide similar results for fuzzy automata defined over structures which are not idempotent and locally finite, unlike the Gödel structure, but we will aim to find alternative results for such fuzzy automata.

References

[1] G. Bailador and G. Triviño. Pattern recognition using temporal fuzzy automata. *Fuzzy Sets and Systems*, 161(1):37–55, 2010.

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

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

[4] A. Blanco, M. Delgado, and M. Pegalajar. Fuzzy automaton induction using neural network. *International Journal of Approximate Reasoning*, 27(1):1–26, 2001.

[5] Y. Cao and Y. Ezawa. Nondeterministic fuzzy automata. *Information Sciences*, 191:86–97, 2012.

[6] Y. Cao, G. Chen, and E. E. Kerre. Bisimulations for fuzzy-transition systems. *IEEE Transactions on Fuzzy Systems*, 19 (3):540–552, 2011. doi: 10.1109/TFUZZ.2011.2117431.

[7] Y. Cao, S. X. Sun, H. Wang, and G. Chen. A behavioral distance for fuzzy-transition systems. *IEEE Transactions on Fuzzy Systems*, 21(4):735–747, 2013.

[8] X. Chen and H. Xing. Nonblocking check in fuzzy discrete event systems based on observation equivalence. *Fuzzy Sets and Systems*, 269:47–64, 2015.

[9] M. Ćirić, A. Stamenković, J. Ignjatović, and T. Petković. Factorization of fuzzy automata. In *Proceedings of the 16th International Conference on Fundamentals of Computation Theory*, FCT'07, pages 213–225. Springer-Verlag, 2007.

[10] M. Ćirić, J. Ignjatović, and S. Bogdanović. Uniform fuzzy relations and fuzzy functions. *Fuzzy Sets and Systems*, 160(8): 1054 – 1081, 2009.

[11] M. Ćirić, A. Stamenković, J. Ignjatović, and T. Petković. Fuzzy relation equations and reduction of fuzzy automata. *Journal of Computer and System Sciences*, 76(7):609 – 633, 2010.
[12] M. Ćirić, J. Ignjatović, N. Damljanović, and M. Bašić. Bisimulations for fuzzy automata. *Fuzzy Sets and Systems*, 186 (1):100 – 139, 2012.
[13] M. Ćirić, J. Ignjatović, I. Jančić, and N. Damljanović. Computation of the greatest simulations and bisimulations between fuzzy automata. *Fuzzy Sets and Systems*, 208:22–42, 2012.
[14] M. Ćirić, J. Ignjatović, M. Bašić, and I. Jančić. Nondeterministic automata: Equivalence, bisimulations, and uniform relations. *Information Sciences*, 261:185–218, 2014.
[15] N. Damljanović, M. Ćirić, and J. Ignjatović. Bisimulations for weighted automata over an additively idempotent semiring. *Theoretical Computer Science*, 534:86–100, 2014.
[16] M. Demirci. A theory of vague lattices based on many-valued equivalence relations — i: general representation results. *Fuzzy Sets and Systems*, 151(3):437–472, 2005.
[17] W. Deng and D. Qiu. Supervisory control of fuzzy discrete-event systems for simulation equivalence. *IEEE Transactions on Fuzzy Systems*, 23:178–192, 2015.
[18] A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *Theoretical Computer Science*, 311(1):221–256, 2004.
[19] Y. Du and P. Zhu. Labeled fuzzy approximations based on bisimulations. *International Journal of Approximate Reasoning*, 94:43–59, 2018.
[20] D. Dubois and H. Prade. Fuzzy sets and systems: theory and applications. In *Mathematics in science and engineering*, 1980.
[21] L. Esakia. *Heyting algebras: Duality theory*, volume 50. Springer, 2019.
[22] F. Lin and H. Ying. Modeling and control of fuzzy discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(4):408–415, 2002.
[23] T.-F. Fan. Fuzzy bisimulation for Gödel modal logic. *IEEE Transactions on Fuzzy Systems*, 23(6):2387–2396, 2015.
[24] M. Forti and F. Honsell. Set theory with free construction principles. *Annali della Scuola Normale Superiore di Pisa - Classe di Scienze*, Ser. 4, 10(3):493–522, 1983.
[25] R. Gentilini, C. Piazza, and A. Policriti. From bisimulation to simulation: Coarsest partition problems. *Journal of Automated Reasoning*, 31:73–103, 2003.
[26] A. Girard and G. J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798, 2007.
[27] A. Girard and G. J. Pappas. Approximate Bisimulation: A Bridge Between Computer Science and Control Theory. *European Journal of Control*, 17(5-6):568–578, 2011.
[28] A. Girard, A. A. Julius, and G. J. Pappas. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Systems*, 18(2):163–179, 2008.
[29] J. Ignjatović, M. Ćirić, and I. Stanković. Bisimulations in fuzzy social network analysis. In *Proceedings of the 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology*, pages 404–411. Atlantis Press, 2015.
[30] J. Ignjatović, M. Ćirić, and S. Bogdanović. Fuzzy homomorphisms of algebras. *Fuzzy Sets and Systems*, 160(16):2345–2365, 2009.
[31] J. Ignjatović and M. Ćirić. Weakly linear systems of fuzzy relation inequalities and their applications: A brief survey. *Filomat*, 26(2):207–241, 2012.
[32] I. Jančić. Weak bisimulations for fuzzy automata. *Fuzzy Sets and Systems*, 249:49–72, 2014.
[33] Z. Jančić, I. Micić, J. Ignjatović, and M. Ćirić. Further improvement of determinization methods for fuzzy finite automata. *Fuzzy Sets and Systems*, 301:79–102, 2015.
[34] A. Kandel and S. C. Lee. *Fuzzy Switching and Automata: Theory and Applications*. Russak, New York, 1979.
[35] Y. Li. Approximation and robustness of fuzzy finite automata. *International Journal of Approxima*, 47(2):247–257, 2008.
[36] Y. Li. Quantitative model checking of linear-time properties based on generalized possibility measures. *Fuzzy Sets and Systems*, 320:17 – 39, 2017.
[37] Y. Li and J. Wei. Possibilistic fuzzy linear temporal logic and its model checking. *IEEE Transactions on Fuzzy Systems*, pages 1–1, 2020. doi: 10.1109/TFUZZ.2020.2988848.
[38] N. Lynch and F. Vaandrager. Forward and backward simulations. *Information and Computation*, 121(2):214–233, 1995.
[39] D. S. Malik and J. N. Mordeson. *Algebraic Fuzzy Automata*, pages 197–246. Physica-Verlag HD, Heidelberg, 2000. ISBN 978-3-7908-1838-3.
[40] I. Micić, Z. Jančić, J. Ignjatović, and M. Ćirić. Determinization of fuzzy automata by means of the degrees of language inclusion. *IEEE Transactions on Fuzzy Systems*, 23(6):2144–2153, 2015.
[41] I. Micić, Z. Jančić, and S. Stanimirović. Computation of the greatest right and left invariant fuzzy quasi-orders and fuzzy equivalences. *Fuzzy Sets and Systems*, 339:99–118, 2018.
[42] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag, Berlin, Heidelberg, 1982.
[43] R. Milner. *Communication and concurrency*, volume 84. Prentice hall Englewood Cliffs, 1989.
[44] J. N. Mordeson and D. Malik. *Fuzzy automata and languages: Theory and Applications*. Chapman & Hall, CRC Press, Boca Raton, London, 2002.
[45] L. Nguyen. An implementation of the algorithm ComputeAFB in C++. Available at http://www.mimuw.edu.pl/%7Enguyen/compAFB, 2021.
[46] L. Nguyen. Characterizing fuzzy simulations for fuzzy labeled transition systems in fuzzy propositional dynamic logic.

International Journal of Approximate Reasoning, 135:21–37, 2021.
[47] L. Nguyen. Logical characterizations of fuzzy bisimulations in fuzzy modal logics over residuated lattices. *Fuzzy Sets and Systems*, 431:70–93, 2022. doi: https://doi.org/10.1016/j.fss.2021.08.009.
[48] L. Nguyen and D. Tran. Computing fuzzy bisimulations for fuzzy structures under the Gödel semantics. *IEEE Transactions on Fuzzy Systems*, 29(7):1715–1724, 2021.
[49] L. Nguyen, Q.-T. Ha, N.-T. Nguyen, T. Nguyen, and T.-L. Tran. Bisimulation and bisimilarity for fuzzy description logics under the Gödel semantics. *Fuzzy Sets and Systems*, 388:146–178, 2020.
[50] H. Pan, M. Zhang, and Y. Chen. Approximate simulation for metric hybrid input/output automata. In *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement - Companion*, pages 53–59, 2011.
[51] H. Pan, M. Zhang, Y. Chen, and H. Wu. Approximate bisimulation for metric doubly labeled transition system. In *2011 Fifth International Conference on Theoretical Aspects of Software Engineering*, pages 108–114, 2011.
[52] H. Pan, Y. Li, Y. Cao, and P. Li. Nondeterministic fuzzy automata with membership values in complete residuated lattices. *International Journal of Approximate Reasoning*, 82:22–38, 2017.
[53] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, pages 167–183, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg.
[54] S. Qiao and P. Zhu. Limited approximate bisimulations and the corresponding rough approximations. *International Journal of Approximate Reasoning*, 130:50–82, 2021.
[55] J. Raftery. Representable idempotent commutative residuated lattices. *Transactions of the American Mathematical Society*, 359(9):4405–4427, 2007.
[56] K. Ravi, A. Choubey, and K. Tripati. Intuitionistic fuzzy automaton for approximate string matching. *Fuzzy Information and Engineering*, 6(1):29–39, 2014.
[57] G. G. Rigatos. Fault detection and isolation based on fuzzy automata. *Information Sciences*, 179(12):1893–1902, 2009.
[58] M. Roggenbach and M. Majster-Cederbaum. Towards a unified view of bisimulation: a comparative study. *Theoretical Computer Science*, 238(1):81–130, 2000.
[59] A. Stamenković, M. Ćirić, and J. Ignjatović. Reduction of fuzzy automata by means of fuzzy quasi-orders. *Information Sciences*, 275:168 – 198, 2014.
[60] S. Stanimirović, M. Ćirić, and J. Ignjatović. Determinization of fuzzy automata by factorizations of fuzzy states and right invariant fuzzy quasi-orders. *Information Sciences*, 469:79–100, 2018.
[61] S. Stanimirović, I. Micić, and M. Ćirić. Approximate bisimulations for fuzzy automata over complete heyting algebras. *IEEE Transactions on Fuzzy Systems*, 30:437–447, 2022.
[62] F. Steimann and K.-P. Adlassnig. Clinical monitoring with fuzzy automata. *Fuzzy Sets and Systems*, 61(1):37–42, 1994.
[63] D. D. Sun, Y. M. Li, and W. W. Yang. Bisimulation relations for fuzzy finite automata. *Fuzzy Systems and Mathematics*, 23:92–100, 2009 (In Chinese).
[64] M. Thomason. Finite fuzzy automata, regular fuzzy languages, and pattern recognition. *Pattern Recognition*, 5(4): 383–390, 1973.
[65] J. van Benthem. *Modal correspondence theory*. PhD thesis, Universiteit van Amsterdam, Instituut voor Logica en Grondslagenonderzo ek van Exacte Wetenschappen, 1976.
[66] H. Wu and Y. Deng. Logical characterizations of simulation and bisimulation for fuzzy transition systems. *Fuzzy Sets and Systems*, 301:19–36, 2016.
[67] H. Wu and Y. Deng. Distribution-based behavioral distance for nondeterministic fuzzy transition systems. *IEEE Transactions on Fuzzy Systems*, 26(2):416–429, 2018.
[68] H. Wu, T. Chen, T. Han, and Y. Chen. Bisimulations for fuzzy transition systems revisited. *International Journal of Approximate Reasoning*, 99:1–11, 2018.
[69] H. Wu, Y. Chen, T. Bu, and Y. Deng. Algorithmic and logical characterizations of bisimulations for non-deterministic fuzzy transition systems. *Fuzzy Sets and Systems*, 333:106–123, 2018.
[70] Q. Wu, Z. Han, and Q. E. Wu. Application of fuzzy automata decision-making system in target control. *Journal of Computer and Communications*, 05(10):16–25, 2017.
[71] Q.-E. Wu, X.-M. Pang, and Z.-Y. Han. Fuzzy automata system with application to target recognition based on image processing. *Computers & Mathematics with Applications*, 61(5):1267–1277, 2011.
[72] H. Xing, Q. Zhang, and K. Huang. Analysis and control of fuzzy discrete event systems using bisimulation equivalence. *Theoretical Computer Science*, 456:100–111, 2012.
[73] Y. Cao and M. Ying. Observability and decentralized control of fuzzy discrete-event systems. *IEEE Transactions on Fuzzy Systems*, 14(2):202–216, 2006.
[74] C. Yang and Y. Li. ϵ-bisimulation relations for fuzzy automata. *IEEE Transactions on Fuzzy Systems*, 26(4):2017–2029, 2018.
[75] C. Yang and Y. Li. Approximate bisimulation relations for fuzzy automata. *Soft Computing*, 22(14):4535–4547, 2018.
[76] C. Yang and Y. Li. Approximate bisimulations and state reduction of fuzzy automata under fuzzy similarity measures. *Fuzzy Sets and Systems*, 391:72 – 95, 2020.
[77] M. Ying. Bisimulation indexes and their applications. *Theoretical Computer Science*, 275(1):1–68, 2002.
[78] M. Ying and M. Wirsing. Approximate bisimilarity. In T. Rus, editor, *Algebraic Methodology and Software Technology*, pages 309–322, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
[79] J. Zhang and Z. Zhu. A modal characterization of alternating approximate bisimilarity. *Form. Methods Syst. Des.*, 44(3): 240–263, 2014.

Supplementary Material

This supplementary material is created by taking the related text of [48] and making necessary changes to obtain the algorithms ComputeAFB and ComputeAFBD together with proofs of Theorems 5.3 and 5.4. The first algorithm computes the greatest λ-AFB between two finite fuzzy automata when $\mathcal{H}$ is the Gödel structure, if it exists, whereas the second one computes the greatest $\lambda \in [0,1]$ such that there exists a λ-AFB between the automata. Our contribution in this supplement is our adaptation of the related text, results and proofs of [48], which relies on dealing with λ-approximation and reformulating the text by using the notation of fuzzy automata instead of the notation of fuzzy description logic. The main change with respect to the proof in [48] is our Lemma A.11 together with its proof and the proofs of Theorems 5.3 and 5.4 themselves. In addition, we also provide a new illustrative example.

Like for Section 5, in this supplement, $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A \rangle$ and $\mathcal{A}' = \langle A', \delta^{A'}, \sigma^{A'}, \tau^{A'} \rangle$ are given finite fuzzy automata over an alphabet X, $\mathcal{H}$ is the Gödel structure, i.e. $H = [0,1]$ and $x \wedge y = \min\{x, y\}$ for $x, y \in H$. We will use the notation introduced in Section 5.

A.1. Data Structures

We denote:

$$Edges = \{\langle a, x, b\rangle \mid a, b \in A_\oplus, x \in X, E_x(a,b) > 0\},$$
$$Edges' = \{\langle a', x, b'\rangle \mid a', b' \in A'_\oplus, x \in X, E'_x(a',b') > 0\}.$$

For $x \in X$, $a, b \in A_\oplus$ and $a', b' \in A'_\oplus$, we define:

- $Next_x(a) = \{b \mid \langle a, x, b\rangle \in Edges\}$, $Prev_x(b) = \{a \mid \langle a, x, b\rangle \in Edges\}$,
- $Next'_x(a') = \{b' \mid \langle a', x, b'\rangle \in Edges'\}$, $Prev'_x(b') = \{a' \mid \langle a', x, b'\rangle \in Edges'\}$.

The algorithm ComputeAFB uses the following data structures:

- $Z : A_\oplus \times A'_\oplus \to [0,1]$
- $doneZ : A_\oplus \times A'_\oplus \to \{true, false\}$
- $EZ : X \to (A_\oplus \times A'_\oplus \to [0,1])$
- $countEZ : X \to (A_\oplus \times A'_\oplus \to \mathbb{N})$
- $doneEZ : X \to (A_\oplus \times A'_\oplus \to \{true, false\})$
- $ZrE' : X \to (A_\oplus \times A'_\oplus \to [0,1])$
- $countZrE' : X \to (A_\oplus \times A'_\oplus \to \mathbb{N})$
- $doneZrE' : X \to (A_\oplus \times A'_\oplus \to \{true, false\})$
- for each $e \in Edges$:
  - $e.doneV : A'_\oplus \to \{true, false\}$
  - $e.done : \{true, false\}$
- for each $e' \in Edges'$:
  - $e'.doneV' : A_\oplus \to \{true, false\}$
  - $e'.done' : \{true, false\}$

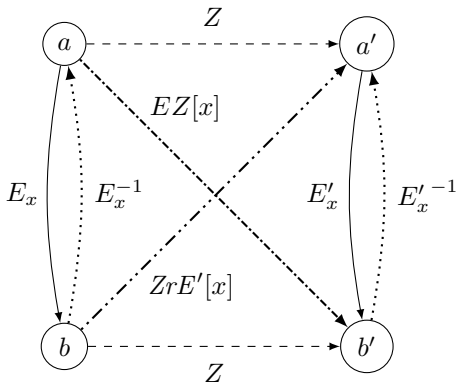

Figure 1: An illustration for the data structures described in Section A.1.

The intuition behind the data structures is given below.

At the end of the computation, if $Z[i, i'] = 1$, then Z restricted to $A \times A'$ gives the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$, else such a bisimulation does not exist.

For $x \in X$, $EZ[x]$ represents $E_x \circ Z$, whereas $ZrE'[x]$ represents $Z \circ {E'_x}^{-1}$. The name EZ stands for the "sequential composition" of the set of "edges" of $\mathcal{A}$ and Z. The name ZrE' stands for the "sequential composition" of Z and the set of "reversed edges" of $\mathcal{A}'$. The data structures EZ and ZrE' are illustrated in Figure 1, with $a, b \in A_{\oplus}$, $a', b' \in A'_{\oplus}$ and $x \in X$.

Consider the remaining data structures. A fact $doneZ[a, a'] = true$ means that the pair $\langle a, a' \rangle$ has been "processed" for Z and the value of $Z[a, a']$ has been computed and is final. The data structures $doneEZ$, $doneZrE'$, $e.doneV$, $e'.doneV'$ and the attributes $e.done$ and $e'.done'$, for $e \in Edges$ and $e' \in Edges'$, have a similar meaning. For example, $doneEZ[x][a, b'] = true$ means that the tuple $\langle a, x, b' \rangle$ has been "processed" for EZ and the value of $EZ[x][a, b']$ has been computed and is equal to the final value of $(E_x \circ Z)(a, b')$.

The value of $countEZ[x][a, b']$, for $x \in X$, $a \in A_{\oplus}$ and $b' \in A'_{\oplus}$, means the number of elements $b \in Next_x(a)$ that have not yet been "processed" for the tuple $\langle a, x, b' \rangle$ with respect to EZ. When $countEZ[x][a, b']$ becomes 0, the tuple $\langle a, x, b' \rangle$ is ready to be "processed" for EZ. The data structure $countZrE'$ is similar to $countEZ$.

For $x \in X$, we will write EZ_x to denote $EZ[x]$, and similarly for $countEZ_x$, $doneEZ_x$, ZrE'_x, $countZrE'_x$ and $doneZrE'_x$. In the following, the subscripts Z, E, E', EZ and ZrE' of tuples are constants, they are used to distinguish each from the others. Let:

- $\mathbb{Z}$ denote $\{\langle a, a' \rangle_Z \mid a \in A_{\oplus}, a' \in A'_{\oplus}, \neg doneZ[a, a']\}$,
- $\mathbb{E}$ denote $\{\langle a, x, b \rangle_E \mid e = \langle a, x, b \rangle \in Edges, \neg e.done\}$,
- $\mathbb{E}'$ denote $\{\langle a', x, b' \rangle_{E'} \mid e' = \langle a', x, b' \rangle \in Edges', \neg e'.done'\}$,
- $\mathbb{EZ}$ denote $\{\langle a, x, b' \rangle_{EZ} \mid x \in X, a \in A_{\oplus}, b' \in A'_{\oplus}, countEZ_x[a, b'] = 0, \neg doneEZ_x[a, b']\}$,
- $\mathbb{Z}r\mathbb{E}'$ denote $\{\langle b, x, a' \rangle_{ZrE'} \mid x \in X, b \in A_{\oplus}, a' \in A'_{\oplus}, countZrE'_x[b, a'] = 0, \neg doneZrE'_x[b, a']\}$,
- $\mathbb{Q}$ denote $\mathbb{Z} \cup \mathbb{E} \cup \mathbb{E}' \cup \mathbb{EZ} \cup \mathbb{Z}r\mathbb{E}'$.

For a tuple $t \in \mathbb{Q}$, the *weight* of t, denoted by $weight(t)$, is defined to be:

- $Z[a, a']$ if $t = \langle a, a' \rangle_Z$,
- $E_x(a, b)$ if $t = \langle a, x, b \rangle_E$,

Procedure InitializeAFB($\mathcal{A}, \mathcal{A}'$)

foreach $a \in A_\oplus$ *and* $a' \in A'_\oplus$ **do**
  if $a \in A$ *and* $a' \in A'$ **then** $Z[a, a'] := 1$;
  else $Z[a, a'] := 0$;
  $doneZ[a, a'] := false$;
$Z[i, i'] := 1$, $Z[f, f'] := 1$;
foreach $x \in X$, $a \in A_\oplus$ *and* $b' \in A'_\oplus$ **do**
  $EZ_x[a, b'] := 0$, $doneEZ_x[a, b'] := false$, $countEZ_x[a, b'] := |Next_x(a)|$;
foreach $x \in X$, $b \in A_\oplus$ *and* $a' \in A'_\oplus$ **do**
  $ZrE'_x[b, a'] := 0$, $doneZrE'_x[b, a'] := false$, $countZrE'_x[b, a'] := |Next'_x(a')|$;
foreach $e = \langle a, x, b\rangle \in Edges$ **do**
  $e.done := false$;
  foreach $b' \in A'_\oplus$ **do** $e.doneV[b'] := false$;
foreach $e' = \langle a', x, b'\rangle \in Edges'$ **do**
  $e'.done' := false$;
  foreach $b \in A_\oplus$ **do** $e'.doneV'[b] := false$;

- $E'_x(a', b')$ if $t = \langle a', x, b'\rangle_{E'}$,
- $EZ_x[a, b']$ if $t = \langle a, x, b'\rangle_{EZ}$,
- $ZrE'_x[b, a']$ if $t = \langle b, x, a'\rangle_{ZrE'}$.

We treat $\mathbb{Q}$ as a *priority queue* of tuples to be processed, where the smaller the weight, the higher the priority. The data structures are initialized by the procedure `InitializeAFB`$(\mathcal{A}, \mathcal{A}')$ (on page 3).

A.2. The Algorithm ComputeAFB

As the main part, the algorithm ComputeAFB computes the fuzzy relation Z specified in Corollary 5.2. Recall that the final Z must satisfy the conditions (38), (39) and (41)–(44) (for all $x \in X$). The algorithm ComputeAFB uses the data structures specified in Section A.1 and initializes them by using the procedure `InitializeAFB`$(\mathcal{A}, \mathcal{A}')$. Thus, Z is initialized to the greatest fuzzy relation that satisfies (41)–(44), and EZ_x and ZrE'_x (for $x \in X$) are initialized to the empty fuzzy relations. After that, fuzzy values are propagated backward through (the graph representing) the data structures as follows:

- The weights of tuples from Z are used to update ZrE'_x and EZ_x (for $x \in X$) in order to guarantee that $ZrE'_x = Z \circ {E'_x}^{-1}$ and $EZ_x = E_x \circ Z$, respectively.
- The weights of tuples from ${E'_x}^{-1}$ are used to update ZrE'_x in order to guarantee that $ZrE'_x = Z \circ {E'_x}^{-1}$.
- The weights of tuples from E_x are used to update EZ_x in order to guarantee that $EZ_x = E_x \circ Z$.
- The weights of tuples from ZrE'_x are used to update Z in order to realize the requirement $\lambda \wedge E_x^{-1} \circ Z \leqslant ZrE'_x$.
- The weights of tuples from EZ_x are used to update Z in order to realize the requirement $\lambda \wedge Z \circ E'_x \leqslant EZ_x$.
- Each time only one tuple is processed, it is the tuple with a smallest weight taken from the priority queue $\mathbb{Q}$. In this way, the weights of tuples from ZrE'_x and EZ_x may change only to bigger ones, and the weights of tuples from Z may change only to smaller ones. When the weight of the tuple taken from $\mathbb{Q}$ is greater than or equal to λ, the main loop is terminated because no further changes can be made for Z by realizing the requirements $\lambda \wedge E_x^{-1} \circ Z \leqslant ZrE'_x$ and $\lambda \wedge Z \circ E'_x \leqslant EZ_x$.

Algorithm 1: ComputeAFB

Input: finite fuzzy automata $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A \rangle$ and $\mathcal{A}' = \langle A', \delta^{A'}, \sigma^{A'}, \tau^{A'} \rangle$ over an alphabet X and the Gödel structure, as well as $\lambda \in (0, 1]$.

Output: the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$ over the Gödel structure, if it exists.

```
1  InitializeAFB(A, A');
2  while ℚ is not empty do
3      choose a tuple t with a smallest weight(t) from ℚ;
4      if weight(t) ⩾ λ then break ;
5      if t = ⟨b, b'⟩_Z then ProcessTupleZ(b, b');
6      else if t = ⟨a, x, b⟩_E then ProcessTupleE(a, x, b);
7      else if t = ⟨a', x, b'⟩_E' then ProcessTupleE'(a', x, b');
8      else if t = ⟨a, x, b'⟩_EZ then ProcessTupleEZ(a, x, b');
9      else if t = ⟨b, x, a'⟩_ZrE' then ProcessTupleZrE'(b, x, a');
10 if Z[i, i'] = 1 then
11     let φ : A × A' → [0, 1] be the restriction of Z to A × A';
12     return φ;
13 else
14     terminate with the information that the greatest λ-AFB between A and A' over the Gödel
         structure does not exist;
```

Procedure ProcessTupleZ(b, b')

```
1  foreach x ∈ X and a' ∈ Prev_x(b') do
2      let e' = ⟨a', x, b'⟩;
3      if ¬e'.doneV'[b] then
4          ZrE'_x[b, a'] := Z[b, b'];
5          decrease countZrE'_x[b, a'] by 1;
6          e'.doneV'[b] := true;
7  foreach x ∈ X and a ∈ Prev_x(b) do
8      let e = ⟨a, x, b⟩;
9      if ¬e.doneV[b'] then
10         EZ_x[a, b'] := Z[b, b'];
11         decrease countEZ_x[a, b'] by 1;
12         e.doneV[b'] := true;
13 doneZ[b, b'] := true;
```

Procedure ProcessTupleE(a, x, b)

```
1 let e = ⟨a, x, b⟩;
2 foreach b' ∈ A'_⊕ do
3     if ¬e.doneV[b'] then
4         EZ_x[a, b'] := E_x(a, b);
5         decrease countEZ_x[a, b'] by 1;
6         e.doneV[b'] := true;
7 e.done := true;
```

Procedure ProcessTupleE$'(a', x, b')$

```
1 let e' = ⟨a', x, b'⟩;
2 foreach b ∈ A_⊕ do
3     if ¬e'.doneV'[b] then
4         ZrE'_x[b, a'] := E'_x(a', b');
5         decrease countZrE'_x[b, a'] by 1;
6         e'.doneV'[b] := true;
7 e'.done' := true;
```

Procedure ProcessTupleEZ(a, x, b')

```
1 foreach a' ∈ Prev'_x(b') do
2     if ¬doneZ[a, a'] and E'_x(a', b') > EZ_x[a, b'] then
3         Z[a, a'] := EZ_x[a, b'];
4 doneEZ_x[a, b'] := true;
```

Procedure ProcessTupleZrE$'(b, x, a')$

```
1 foreach a ∈ Prev_x(b) do
2     if ¬doneZ[a, a'] and E_x(a, b) > ZrE'_x[b, a'] then
3         Z[a, a'] := ZrE'_x[b, a'];
4 doneZrE'_x[b, a'] := true;
```

- The Boolean attributes with a name beginning with "done" (like $doneZ$, $doneEZ_x$, ...) are updated and exploited to avoid redundant computations.

- The value of $EZ_x[a, b']$ is final and ready to be used only when $countEZ_x[a, b']$ becomes 0. This counter is updated appropriately, reflecting the computation $(E_x \circ Z)(a, b') = \sup\{E_x(a, b) \wedge Z[b, b'] \mid b \in Next_x(a)\}$. For example, when an element $b' \in A'_\oplus$ has been "processed" for an edge $\langle a, x, b\rangle$, $countEZ_x[a, b']$ is decreased by 1. Similar explanations can be said for ZrE'_x and $countZrE'_x$.

The pseudocode of the algorithm ComputeAFB is given on page 4. It uses some subroutines, which are defined on pages 4 and 5. In this pseudocode, we assume that $\mathbb{Q}$ is updated automatically according to its definition. For example, $\mathbb{Q}$ can be updated before each of its uses at the statement 2 in the straightforward way according to its definition. As mentioned in the paper, we have implemented the algorithm ComputeAFB in C++ and shared the codes publicly [45].

In the example given below, for a fuzzy set $\alpha : \Gamma \to [0, 1]$, we write $\alpha = \{x_1 : p_1, \ldots, x_n : p_n\}$ to denote that $\alpha(x_i) = p_i$, for $1 \leq i \leq n$, and $\alpha(x) = 0$ for $x \in \Gamma \setminus \{x_1, \ldots, x_n\}$.

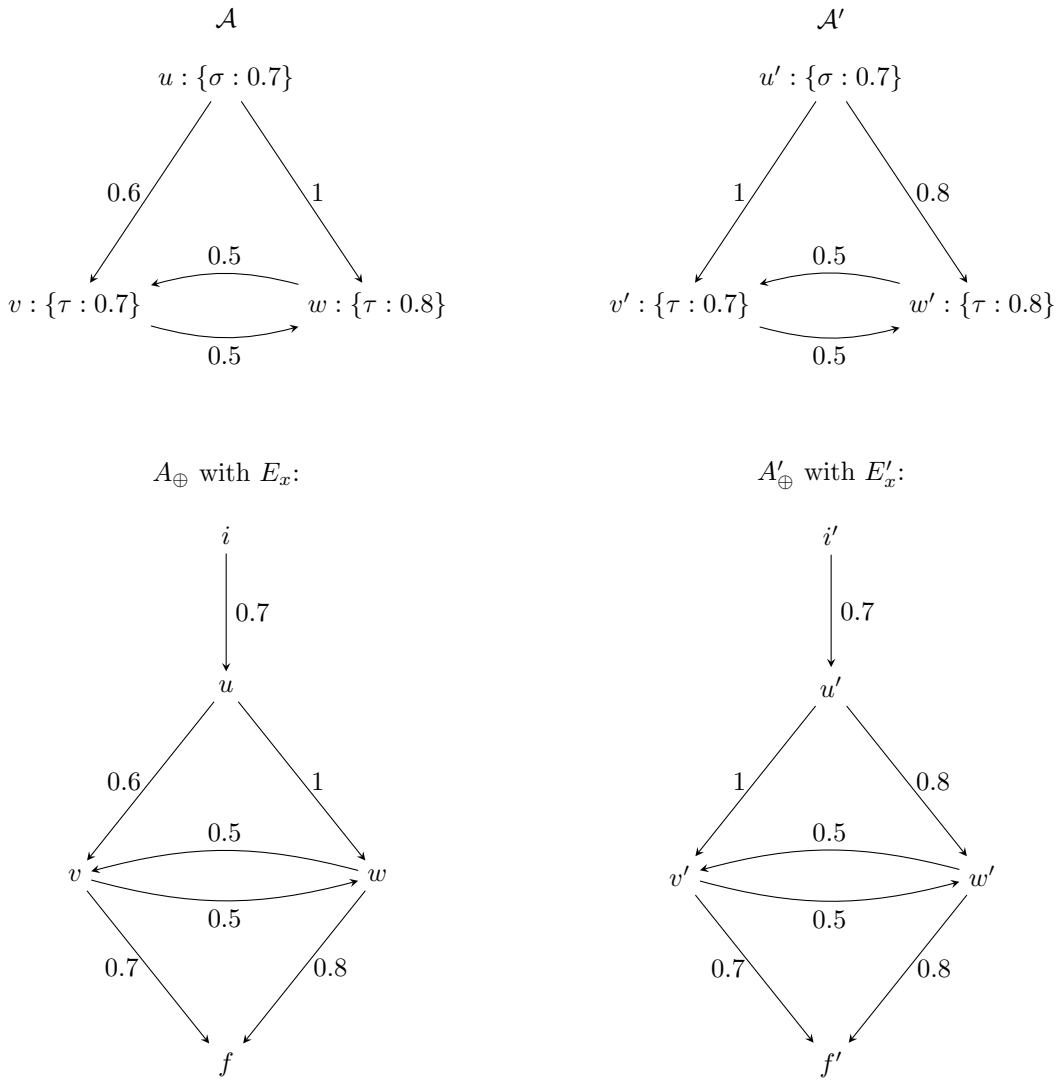

Figure 2: An illustration for Example A.1.

Example A.1. Let $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A \rangle$ and $\mathcal{A}' = \langle A', \delta^{A'}, \sigma^{A'}, \tau^{A'} \rangle$ be the finite fuzzy automata over $X = \{x\}$ that are specified below and illustrated in the upper part of Figure 2.

- $A = \{u, v, w\}$, $\sigma^A = \{u{:}0.7\}$, $\tau^A = \{v{:}0.7, w{:}0.8\}$,
  $\delta_x^A = \{\langle u, v\rangle{:}0.6, \langle u, w\rangle{:}1, \langle v, w\rangle{:}0.5, \langle w, v\rangle{:}0.5\}$;
- $A' = \{u', v', w'\}$, $\sigma^{A'} = \{u'{:}0.7\}$, $\tau^{A'} = \{v'{:}0.7, w'{:}0.8\}$,
  $\delta_x^{A'} = \{\langle u', v'\rangle{:}1, \langle u', w'\rangle{:}0.8, \langle v', w'\rangle{:}0.5, \langle w', v'\rangle{:}0.5\}$.

The sets $A_\oplus$, $A'_\oplus$ and the fuzzy relations E_x, E'_x are illustrated in the lower part of Figure 2. Let's trace the execution of the algorithm ComputeAFB for $\mathcal{A}$, $\mathcal{A}'$ and $\lambda \in \{0.7, 0.8\}$.[1] We will not report the changes made for the arrays $doneZ$, $doneEZ_x$, $doneZrE'_x$ and the attributes $done$, $doneV$, $done'$, $doneV'$ of edges.

- After initialization and executing iterations of the main loop for all tuples from $\mathbb{Q}$ with the weight 0, we have that:
  - the fuzzy relations EZ_x and ZrE'_x are still empty, but the arrays $countEZ_x$ and $countZrE'_x$ have been updated appropriately;
  - $Z[v, u']$, $Z[w, u']$, $Z[u, v']$ and $Z[u, w']$ have been set to 0 by the main loop and therefore $Z = \{\langle i, i'\rangle{:}1, \langle f, f'\rangle{:}1, \langle u, u'\rangle{:}1, \langle v, v'\rangle{:}1, \langle v, w'\rangle{:}1, \langle w, v'\rangle{:}1, \langle w, w'\rangle{:}1\}$.

  For example, $Z[v, u']$ and $Z[w, u']$ were set to 0 by the call ProcessTupleZrE′(f, x, u'). The tuple $\langle f, x, u'\rangle_{ZrE'}$ from $\mathbb{Q}$ was processed because both the tuples $\langle f, v'\rangle_Z$ and $\langle f, w'\rangle_Z$ had been processed, which had reduced $countZrE'_x[f, u']$ to 0.
- Processing the tuple $\langle v, x, w\rangle_E$, whose weight is 0.5,
  - $EZ_x[v, v']$ is increased from 0 to 0.5 and $countEZ_x[v, v']$ is reduced from 1 to 0,
  - $EZ_x[v, w']$ is increased from 0 to 0.5 and $countEZ_x[v, w']$ is reduced from 1 to 0.
- Processing the tuple $\langle w, x, v\rangle_E$, whose weight is 0.5,
  - $EZ_x[w, v']$ is increased from 0 to 0.5 and $countEZ_x[w, v']$ is reduced from 1 to 0,
  - $EZ_x[w, w']$ is increased from 0 to 0.5 and $countEZ_x[w, w']$ is reduced from 1 to 0.
- Processing the tuple $\langle v', x, w'\rangle_{E'}$, whose weight is 0.5,
  - $ZrE'_x[v, v']$ is increased from 0 to 0.5 and $countZrE'_x[v, v']$ is reduced from 1 to 0,
  - $ZrE'_x[w, v']$ is increased from 0 to 0.5 and $countZrE'_x[w, v']$ is reduced from 1 to 0.
- Processing the tuple $\langle w', x, v'\rangle_{E'}$, whose weight is 0.5,
  - $ZrE'_x[v, w']$ is increased from 0 to 0.5 and $countZrE'_x[v, w']$ is reduced from 1 to 0,
  - $ZrE'_x[w, w']$ is increased from 0 to 0.5 and $countZrE'_x[w, w']$ is reduced from 1 to 0.
- Processing the tuples $\langle v, x, v'\rangle_{EZ}$, $\langle v, x, w'\rangle_{EZ}$, $\langle w, x, v'\rangle_{EZ}$, $\langle w, x, w'\rangle_{EZ}$, $\langle v, x, v'\rangle_{ZrE'}$, $\langle v, x, w'\rangle_{ZrE'}$, $\langle w, x, v'\rangle_{ZrE'}$ and $\langle w, x, w'\rangle_{ZrE'}$, whose weights are 0.5, only the arrays $doneEZ_x$ and $doneZrE'_x$ are updated.
- Processing the tuple $\langle u, x, v\rangle_E$, whose weight is 0.6,
  - $EZ_x[u, v']$ is increased from 0 to 0.6 and $countEZ_x[u, v']$ is reduced from 2 to 1,

[1] A full trace of that execution can be obtained by running the implementation [45] for the input files in5.txt and in5p.txt. The states u, v, w, i, and f (respectively, u', v', w', i' and f') are denoted in that trace as 0, 1, 2, 3 and 4, respectively.

– $EZ_x[u, w']$ is increased from 0 to 0.6 and $countEZ_x[u, w']$ is reduced from 2 to 1.

If $\lambda = 0.7$, then when the next tuple is taken from $\mathbb{Q}$, as its weight is 0.7, the main loop of the algorithm terminates and the restriction of Z to $A \times A'$ is returned, which is $\{\langle u, u'\rangle{:}1, \langle v, v'\rangle{:}1, \langle v, w'\rangle{:}1, \langle w, v'\rangle{:}1, \langle w, w'\rangle{:}1\}$.

If $\lambda = 0.8$, then the main loop of the algorithm continues with the following steps.

- Processing the tuple $\langle v, x, f\rangle_E$, whose weight is 0.7, $EZ_x[v, f']$ is increased from 0 to 0.7 and $countEZ_x[v, f']$ is reduced from 1 to 0.
- Processing the tuple $\langle v, x, f'\rangle_{EZ}$, whose weight is 0.7, $Z[v, w']$ is reduced from 1 to 0.7.
- Processing the tuple $\langle v, w'\rangle_Z$, whose weight is 0.7, $ZrE'_x[v, u']$ is increased from 0 to 0.7 and $countZrE'_x[v, u']$ is reduced from 2 to 1.
- Processing the tuple $\langle i, x, u\rangle_E$, whose weight is 0.7, $EZ_x[i, u']$ is increased from 0 to 0.7 and $countEZ_x[i, u']$ is reduced from 1 to 0.
- Processing the tuple $\langle i', x, u'\rangle_{E'}$, whose weight is 0.7, $ZrE'_x[u, i']$ is increased from 0 to 0.7 and $countZrE'_x[u, i']$ is reduced from 1 to 0.
- Processing the tuple $\langle v', x, f'\rangle_{E'}$, whose weight is 0.7, $ZrE'_x[f, v']$ is increased from 0 to 0.7 and $countZrE'_x[f, v']$ is reduced from 1 to 0.
- Processing the tuple $\langle f, x, v'\rangle_{ZrE'}$, whose weight is 0.7, $Z[w, v']$ is reduced from 1 to 0.7.
- Processing the tuple $\langle w, v'\rangle_Z$, whose weight is 0.7,
  - $EZ_x[u, v']$ is increased from 0.6 to 0.7 and $countEZ_x[u, v']$ is reduced from 1 to 0,
  - $ZrE'_x[w, u']$ is increased from 0 to 0.7 and $countZrE'_x[w, u']$ is reduced from 2 to 1.
- Processing the tuple $\langle u, x, v'\rangle_{EZ}$, whose weight is 0.7, $Z[u, u']$ is reduced from 1 to 0.7.
- Processing the tuples $\langle u, u'\rangle_Z$, $\langle i, x, u'\rangle_{EZ}$ and $\langle u, x, i'\rangle_{ZrE'}$, whose weights are 0.7, only the arrays $doneZ$, $doneEZ_x$ and $doneZrE'_x$ are updated.

When the next tuple is taken from $\mathbb{Q}$, as its weight is 0.8 (the same as λ), the main loop of the algorithm terminates and the restriction of Z to $A \times A'$ is returned, which is $\{\langle u, u'\rangle : 0.7, \langle v, v'\rangle : 1, \langle v, w'\rangle : 0.7, \langle w, v'\rangle{:}0.7, \langle w, w'\rangle{:}1\}$. □

A.3. Proof of Theorem 5.3

To prove Theorem 5.3, we need a number of lemmas, which together with their corollaries state essential properties of the algorithm ComputeAFB.

Lemma A.2. *The algorithm* ComputeAFB *always terminates.*

Proof. Each tuple that may belong to $\mathbb{Q}$ at some step is processed only once. This is controlled by the Boolean attributes with a name beginning with "done" like $doneZ$. As the number of all possible tuples that may belong $\mathbb{Q}$ is bounded, the algorithm ComputeAFB always terminates. □

Lemma A.3. *The following properties are invariants of the main loop of the algorithm* ComputeAFB *for every $x \in X$:*

1. *For every $a \in A_\oplus$ and $b' \in A'_\oplus$, $countEZ_x[a, b'] = |\{b \in Next_x(a) : \neg e.doneV[b']$, where $e = \langle a, x, b\rangle\}|$.*
2. *For every $b \in A_\oplus$ and $a' \in A'_\oplus$, $countZrE'_x[b, a'] = |\{b' \in Next'_x(a') : \neg e'.doneV'[b]$, where $e' = \langle a', x, b'\rangle\}|$.*

3. *For every* $a, b \in A_{\oplus}$ *and* $b' \in A'_{\oplus}$*, if* $e = \langle a, x, b\rangle \in Edges$*, then* $e.doneV[b'] = true$ *iff* $e.done = true$ *or* $doneZ[b, b'] = true$*.*
4. *For every* $a', b' \in A'_{\oplus}$ *and* $b \in A_{\oplus}$*, if* $e' = \langle a', x, b'\rangle \in Edges'$*, then* $e'.doneV'[b] = true$ *iff* $e'.done' = true$ *or* $doneZ[b, b'] = true$*.*

The proof of this lemma is straightforward.

Corollary A.4. *When* $\mathbb{Q}$ *is empty, we have* $countEZ_x[a, b'] = 0$ *and* $countZrE'_x[b, a'] = 0$ *for all* $x \in X$*,* $a, b \in A_{\oplus}$ *and* $a', b' \in A'_{\oplus}$*.*

This corollary immediately follows from Lemma A.3 and the definition of $\mathbb{Q}$.

Lemma A.5. *If* $Z[a, a']$ *changes from* v_1 *to* v_2 *during an iteration of the main loop of the algorithm* ComputeAFB*, then* v_2 *is the weight of the tuple chosen by the statement 3 of the algorithm during that iteration and* $v_2 \leq v_1$*.*

The proof of this lemma is straightforward.

Lemma A.6. *If* $\langle a, x, b'\rangle_{EZ}$ *(respectively,* $\langle b, x, a'\rangle_{ZrE'}$*) is implicitly added to* $\mathbb{Q}$ *during an iteration of the main loop of the algorithm* ComputeAFB *because* $countEZ_x[a, b']$ *(respectively,* $countZrE'_x[b, a']$*) becomes 0, then* $EZ_x[a, b']$ *(respectively,* $ZrE'_x[b, a']$*) is equal to the weight of the tuple chosen by the statement 3 of the algorithm during that iteration.*

The proof of this lemma is straightforward.

Corollary A.7. *Let* v_1 *(respectively,* v_2*) be the weight of the tuple chosen by the statement 3 of the algorithm* ComputeAFB *at an iteration* k_1 *(respectively,* k_2*) of the main loop. If* $k_1 < k_2$*, then* $v_1 \leq v_2$*.*

This corollary follows from Lemmas A.5 and A.6.

Corollary A.8. *If* $EZ_x[a, b']$ *(respectively,* $ZrE'_x[b, a']$*) changes from* v_1 *to* v_2 *during an iteration of the main loop of the algorithm* ComputeAFB*, then* v_2 *is the weight of the tuple chosen by the statement 3 of the algorithm during that iteration and* $v_2 \geq v_1$*.*

This corollary follows from Corollary A.7.

Lemma A.9. *The following properties are invariants of the main loop of the algorithm* ComputeAFB *for every* $x \in X$*:*

1. *For every* $a \in A_{\oplus}$ *and* $b' \in A'_{\oplus}$*, if* $e.doneV[b'] = false$ *for all* $e = \langle a, x, b\rangle \in Edges$*, then* $EZ_x[a, b'] = 0$*, else* $EZ_x[a, b'] = E_x(a, b) \wedge Z[b, b']$*, where* b *is the element such that* $e = \langle a, x, b\rangle \in Edges$ *and* $e.doneV[b']$ *was changed to true last.*
2. *For every* $b \in A_{\oplus}$ *and* $a' \in A'_{\oplus}$*, if* $e'.doneV'[b] = false$ *for all* $e' = \langle a', x, b'\rangle \in Edges'$*, then* $ZrE'_x[b, a'] = 0$*, else* $ZrE'_x[b, a'] = Z[b, b'] \wedge {E'_x}^{-1}(b', a')$*, where* b' *is the element such that* $e' = \langle a', x, b'\rangle \in Edges'$ *and* $e'.doneV'[b]$ *was changed to true last.*

Proof. We prove the first assertion and omit the second one, as its proof is similar. Clearly, if $e.doneV[b'] = false$ for all $e = \langle a, x, b\rangle \in Edges$, then $EZ_x[a, b'] = 0$. Suppose b is the element such that $e = \langle a, x, b\rangle \in Edges$ and $e.doneV[b']$ was changed to *true* last, and it was done at a moment k.

- Consider the case where $e.doneV[b']$ was changed to *true* by the statement 12 of the procedure `ProcessTupleZ`. At the moment k, we had that:
  - $EZ_x[a, b'] = Z[b, b']$ and this was the weight of the tuple $\langle b, b'\rangle_Z$ chosen by the statement 3 of the main loop of the algorithm ComputeAFB during the current iteration;
  - the tuple $\langle a, x, b\rangle_E$ was still in $\mathbb{Q}$ (because, immediately before the moment k, $e.doneV[b'] = false$ and, by the assertion 3 of Lemma A.3, $e.done = false$), and thus $E_x(a, b) \geqslant Z[b, b']$;

– therefore, $EZ_x[a, b'] = E_x(a, b) \wedge Z[b, b']$.

After the moment k, $doneZ[b, b']$ was set to *true* by the statement 13 of the procedure `ProcessTupleZ`. Hence, $Z[b, b']$ did not change after the moment k. $EZ_x[a, b']$ did not change after the moment k either. Therefore, $EZ_x[a, b'] = E_x(a, b) \wedge Z[b, b']$ still holds.

- Consider the case where $e.doneV[b']$ was changed to *true* by the statement 6 of the procedure `ProcessTupleE`. At the moment k, we had that:

  – $EZ_x[a, b'] = E_x(a, b)$ and this was the weight of the tuple $\langle a, x, b\rangle_E$ chosen by the statement 3 of the main loop of the algorithm ComputeAFB during the current iteration;

  – the tuple $\langle b, b'\rangle_Z$ was still in $\mathbb{Q}$ (because, immediately before the moment k, $e.doneV[b'] = false$, and by the assertion 3 of Lemma A.3, $doneZ[b, b'] = false$), and thus $Z[b, b'] \geqslant E_x(a, b)$;

  – therefore, $EZ_x[a, b'] = E_x(a, b) \wedge Z[b, b']$.

  After the moment k, $Z[b, b']$ may have changed to a smaller value, but by Lemma A.5 and Corollary A.7, it would always be greater than or equal to $E_x(a, b)$. As $EZ_x[a, b']$ did not change after the moment k, we still have that $EZ_x[a, b'] = E_x(a, b) \wedge Z[b, b']$.

□

Corollary A.10. *The following properties are invariants of the main loop of the algorithm* ComputeAFB *for every $x \in X$, $a, b \in A_\oplus$ and $a', b' \in A'_\oplus$:*

1. *If $countEZ_x[a, b'] = 0$, then $EZ_x[a, b'] = (E_x \circ Z)(a, b')$.*
2. *If $countZrE'_x[b, a'] = 0$, then $ZrE'_x[b, a'] = (Z \circ {E'_x}^{-1})(b, a')$.*

Proof. The first assertion follows from the assertion 1 of Lemma A.9, Corollary A.8 and the assertion 1 of Lemma A.3. Similarly, the second assertion follows from the assertion 2 of Lemma A.9, Corollary A.8 and the assertion 2 of Lemma A.3. □

Lemma A.11. *When the algorithm* ComputeAFB *has executed the statements 1–9, Z satisfies the conditions (38), (39) and (41)–(44) (for all $x \in X$).*

Proof. Let $x \in X$ and consider the moment when the algorithm ComputeAFB has executed the statements 1–9. Z satisfies the conditions (41)–(44) due to the initialization of Z (by the statement 3 of the procedure `InitializeAFB`) and by Lemma A.5.

We now prove the condition (39). Let $a \in A_\oplus$ and $a', b' \in A'_\oplus$ be chosen arbitrarily. It is sufficient to show that

$$\lambda \wedge Z[a, a'] \wedge E'_x(a', b') \leqslant (E_x \circ Z)(a, b'). \tag{45}$$

There are the following cases:

- Case $countEZ_x[a, b'] > 0$: By Lemma (A.3), there exists $b \in Next_x(a)$ such that $e.doneV[b'] = false$, where $e = \langle a, x, b\rangle$, and consequently, $e.done = false$ and $doneZ[b, b'] = false$. Thus, the tuples $\langle a, x, b\rangle_E$ and $\langle b, b'\rangle_Z$ are still in the queue $\mathbb{Q}$. Therefore, the weights of these tuples are greater than or equal to λ. That is, $E_x(a, b) \geqslant \lambda$ and $Z[b, b'] \geqslant \lambda$. Hence, $(E_x \circ Z)(a, b') \geqslant \lambda$ and (45) holds.

- Case $countEZ_x[a, b'] = 0$ and the tuple $\langle a, x, b'\rangle_{EZ}$ is still in the queue $\mathbb{Q}$: Thus, the weight of the tuple is greater than or equal to λ, which means $EZ_x[a, b'] \geqslant \lambda$. By Lemma A.9, there exists $b \in A_\oplus$ such that $E_x(a, b) \wedge Z[b, b'] = EZ_x[a, b'] \geqslant \lambda$. Therefore, $(E_x \circ Z)(a, b') \geqslant \lambda$ and (45) holds.

- Case $countEZ_x[a,b'] = 0$ and the tuple $\langle a,x,b'\rangle_{EZ}$ is not in the queue $\mathbb{Q}$: Thus, $doneEZ[a,b'] = true$ and the weight of the tuple is less than or equal to λ, which means $EZ_x[a,b'] \leqslant \lambda$. Since $doneEZ[a,b'] = true$, `ProcessTupleEZ`(a,x,b') has been called. Due to the statement 3 of the procedure `ProcessTupleEZ` and by Lemma A.5, we have that $Z[a,a'] \leqslant EZ_x[a,b']$ if $E'_x(a',b') > EZ_x[a,b']$ (the condition of the "if" statement in the procedure `ProcessTupleEZ`). Therefore, $Z[a,a'] \wedge E'_x(a',b') \leqslant EZ_x[a,b'] \leqslant \lambda$. Since $countEZ_x[a,b'] = 0$, by Corollary A.10, $EZ_x[a,b'] = (E_x \circ Z)(a,b')$. It follows that (45) holds.

The condition (38) can be proved analogously. □

Lemma A.12. *Let $Z_2 : A_{\oplus} \times A'_{\oplus} \to [0,1]$ be an arbitrary fuzzy relation that satisfies the conditions (38), (39) and (41)–(44) with Z replaced by Z_2 (for all $x \in X$). It is an invariant of the main loop of the algorithm* ComputeAFB *that $Z_2 \leq Z$.*

This lemma follows from Corollary A.10 and the way Z is initialized and updated. Recall that, when Z is updated, the weight of the tuple taken from the queue $\mathbb{Q}$ is less than λ.

We are now ready to give below a proof of Theorem 5.3.

Proof. By Lemma A.2, the algorithm ComputeAFB always terminates. By Lemmas A.11, A.12 and Corollary 5.2, the algorithm returns a correct result.

In [48] Nguyen and Tran specified an implementation of the algorithm ComputeFuzzyBisimulation and proved that it has a complexity of order $O((m+n)n)$ (see Theorem 15 of [48]), where n is the number of individuals and m is the number of non-zero instances of roles in the given fuzzy interpretations $\mathcal{I}$ and $\mathcal{I}'$. Using our notation, $n = |A_{\oplus}| + |A'_{\oplus}|$ and $m = |Edges| + |Edges'|$.

Apart from the initialization and the post-processing for returning the result, the algorithm ComputeAFB differs from the algorithm ComputeFuzzyBisimulation only in that the main loop may be terminated earlier by the statement 4 (**if** $weight(t) \geqslant \lambda$ **then break**). The initialization and the post-processing do not affect the complexity analysis given in [48] for the algorithm ComputeFuzzyBisimulation.

By Theorem 15 of [48] and the above discussion, we conclude that the algorithm ComputeAFB can be implemented (as specified in Section II.E of [48]) to run in time $O((m+n)n)$, where n is the number of states and m is the number of non-zero transitions of the given finite fuzzy automata. Here, we change m from $|Edges| + |Edges'|$ to $|\delta^A| + |\delta^{A'}|$, where $|\delta^A| = |\{\langle a,x,b\rangle : a,b \in A, x \in X, \delta^A_x(a,b) > 0\}|$ and $|\delta^{A'}|$ is defined similarly, but the change does not affect the complexity order $O((m+n)n)$. □

Algorithm 2: ComputeAFBD

Input: finite fuzzy automata $\mathcal{A} = \langle A, \delta^A, \sigma^A, \tau^A\rangle$ and $\mathcal{A}' = \langle A', \delta^{A'}, \sigma^{A'}, \tau^{A'}\rangle$ over an alphabet X and the Gödel structure.

Output: the greatest $\lambda \in [0,1]$ such that there exists a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$.

```
1  InitializeAFB(A, A');
2  while Q is not empty do
3  |  choose a tuple t with a smallest weight(t) from Q;
4  |  if t = <i, i'>_Z then break ;
5  |  if t = <b, b'>_Z then ProcessTupleZ(b, b');
6  |  else if t = <a, x, b>_E then ProcessTupleE(a, x, b);
7  |  else if t = <a', x, b'>_E' then ProcessTupleE'(a', x, b');
8  |  else if t = <a, x, b'>_EZ then ProcessTupleEZ(a, x, b');
9  |  else if t = <b, x, a'>_ZrE' then ProcessTupleZrE'(b, x, a');
10 return Z[i, i'];
```

A.4. Proof of Theorem 5.4

The algorithm ComputeAFBD is explicitly specified on page 11. It differs from the algorithm ComputeAFB in that the main loop is broken when the tuple $\langle i, i' \rangle_Z$ is chosen from $\mathbb{Q}$ and then the weight of this tuple is returned as the result.

Using a similar argumentation as for the proof of Theorem 5.3, we reach the conclusion that the algorithm ComputeAFBD always terminates and can be implemented to run in time $O((m+n)n)$, where n is the number of states and m is the number of non-zero transitions of $\mathcal{A}$ and $\mathcal{A}'$.

For the correctness of the algorithm ComputeAFBD, let λ be the result returned by this algorithm. Consider the run of the algorithm ComputeAFB for $\mathcal{A}$ and $\mathcal{A}'$ using that λ. We can assume that the main loop of that run behaves in the same way as the algorithm ComputeAFBD until being broken. By Corollary A.7 and Lemma A.5, the loop is broken before the tuple $\langle i, i' \rangle_Z$ is processed and the weight of this tuple remains 1 as initialized by the call `InitializeAFB`$(\mathcal{A}, \mathcal{A}')$. Therefore, the result returned by that run is the greatest λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$. Let $\lambda' \in (\lambda, 1]$ and consider the run of the algorithm ComputeAFB for $\mathcal{A}$ and $\mathcal{A}'$ using λ'. We can assume that it behaves in the same way as the algorithm ComputeAFBD until the tuple $\langle i, i' \rangle_Z$ with the weight λ is chosen from $\mathbb{Q}$. By Corollary A.7 and Lemma A.5, when the main loop of that run terminates, $\mathit{weight}(\langle i, i' \rangle_Z) \leq \lambda < 1$. Therefore, the algorithm ComputeAFB terminates with the information that the greatest λ'-AFB between $\mathcal{A}$ and $\mathcal{A}'$ over the Gödel structure does not exist. By Theorem 3.1, this means that there does not exist any λ'-AFB between $\mathcal{A}$ and $\mathcal{A}'$. Thus, λ is the greatest value from $[0, 1]$ such that there exists a λ-AFB between $\mathcal{A}$ and $\mathcal{A}'$.