Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 348 (2020) 43–60

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# Introducing Synchrony in Fuzzy Automata [1]

Leandro Gomes [2]

*INESC TEC*
*Univ. Minho*
*Braga, Portugal*

Alexandre Madeira[3]

*CIDMA*
*Univ. Aveiro*
*Aveiro, Portugal*

Luis Soares Barbosa[4]

*INESC TEC*
*Univ. Minho*
*Braga, Portugal*
*United Nations University*
*UNU-EGOV*
*Guimarães, Portugal*

**Abstract**

This paper introduces a sort of automata and associated languages, often arising in modelling natural phenomena, in which both *vagueness* and *simultaneity* are taken as first class citizens. This requires a fuzzy semantics assigned to transitions and a precise notion of a synchronous product to enforce the simultaneous occurrence of actions. The expected relationships between automata and languages are revisited in this setting; in particular it is shown that any subset of a fuzzy synchronous language with the suitable signature forms a synchronous Kleene algebra.

*Keywords:* fuzzy automata, fuzzy languages, synchronous languages.

[1] This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within projects POCI-01-0145-FEDER-030947 and UID/MAT/04106/2019. The second author is supported in the scope of the framework contract foreseen in the numbers 4, 5 and 6 of the article 23, of the Decree-Law 57/2016, of August 29, changed by Portuguese Law 57/2017, of July 19. This paper is also a result of the project SmartEGOV: Harnessing EGOV for Smart Governance (Foundations, Methods, Tools) NORTE-01-0145-FEDER-000037, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (EFDR). It received further support from the PT-FLAD Chair in Smart Cities & Smart Governance.

[2] Email: leandro.r.gomes@inesctec.pt

[3] Email: madeira@ua.pt

[4] Email: lsb@di.uminho.pt

https://doi.org/10.1016/j.entcs.2020.02.004

# 1 Introduction

The notion of an automaton [7] as the *de facto* mathematical abstraction of a computational process over discrete spaces, is being constantly revisited to capture different sorts of computational behaviour in the most varied contexts, either prescribed in a program or discovered in Nature. Already in 1997 Robin Milner [15] emphasised that

> *from being a prescription for how to do something – in Turing's terms a 'list of instructions', software becomes much more akin to a description of behaviour, not only programmed on a computer, but also occurring by hap or design inside or outside it.*

Over time different kinds of automata have been proposed accordingly, *generating* (or *recognising*, depending on the perspective) such behaviours (or the languages that express them) [2,3,21]. In this context, Kleene algebra was introduced [8] as an algebraic structure to capture axiomatically the basic properties of regular expressions.

This paper focus on a specific sort of automata and languages, often arising in modelling natural phenomena, in which two extra ingredients cannot be overlooked. The first is *vagueness*. In a biological network [4], for example, this expresses the possibility of a certain enzyme being absent or scarce in certain configurations. The other is *simultaneity*, i.e. the fact that certain events (for example the flow of some reagents in a chemical reaction) are required to happen at the same time, instead of, as usually considered in interleaving models of concurrency, in a non deterministic alternation.

The first ingredient — *vagueness* — is formalised by the notion of a *fuzzy* finite-state automata (FFA), a structure introduced in the Sixties [23] to give a formal semantics to vagueness inherent to several computational systems. Variants of this idea, e.g. incorporating fuzziness to either states or transitions, or both, are well documented in the literature [3,10,12]. In any case, *fuzzy* languages [9,1] are recognised by a FFA only up to a certain membership degree. Applications are transversal to several domains [11,16,17,24]. Probabilistic automata [19], another approach to handle uncertainty, fixes the interpretation of the latter as a probability, always enforcing the production of an outcome (as expressed by the requirement that the outgoing probabilities always sum 1). Such is not the case in the fuzzy framework adopted in this paper.

On its turn, *simultaneity*, our second ingredient, was suitably formalised in what Milner called the 'synchronous version of CCS' — the SCCS [14] calculus, a variant of CCS [13] where arbitrary actions can run *synchronously*. This very same idea of synchronous evolution appears in the work of C. Priscariu on synchronous Kleene algebra (SKA) [18]. Kleene algebras are idempotent, and thus partially ordered, semirings endowed with a closure operator. Models for SKA, as well as for its

variant with tests (SKAT), are given in terms of sets of synchronous strings and finite automata accepting synchronous strings. These structures found application, for instance, in variants of deontic logic to formalise contract languages [20,22], and of Hoare logic to reason about parallel synchronous programs with shared variables [18].

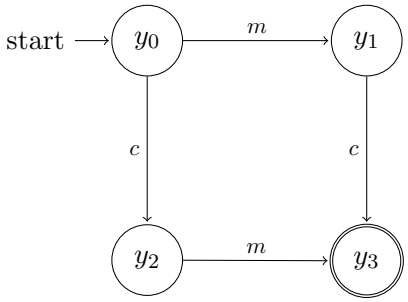


Fig. 1. Interleaving two automata representing deterministic flows.

Both of these ingredients are combined in this paper. The sort of systems we have in mind is illustrated in Fig. 1. Suppose that two automata represent the flow of two reagents $c$ and $m$ into a solution. The scheme of Fig. 1 represents their interleaving as two alternative sequential compositions. Our objective, however, has a different focus. First we intend to record that elementary steps are 'uncertain', in the sense that each individual flow may exhibit failures or interruptions. The transitions in the automata are thus labelled with the flow identifier and a 'certainty' degree measuring how certain it is for each flow (*event*) effectively flowing. Second, their combination makes sure that both flows (*actions*) occur simultaneously combining their 'certainty' degrees into the 'certainty' degree of their joint flow. These features are expressed in the *fuzzy synchronous* automata that this paper proposes to add to the broad family of finite-state automata mentioned above.

In order to formalise such a behaviour, the paper introduces a synchronous product construction between a variant of fuzzy transition automata in the spirit of reference [12] where, depending on the application scenario, "vagueness" can be modelled in an arbitrary, either discrete or continuous, domain. This is captured by a complete Heyting algebra introduced as a parameter in the model. The notion of synchronous sets in reference [18] is generalised to that of *fuzzy synchronous languages*, and some operators over them suitably defined. A map that interprets the terms of SKA as fuzzy synchronous languages is provided. Then we prove that the terms of a SKA can be used to generate a $\mathcal{H}$-NFA accepting precisely the fuzzy synchronous language that constitutes its interpretation. Obtaining a regular expression from a $\mathcal{H}$-NFA proceeds by state elimination as in the classical case [6]. The procedure results in a $\mathcal{H}$-NFA with a single transition from the initial to the final state, labelled by an action $\alpha$ of SKA such that its interpretation is the language recognised by that $\mathcal{H}$-NFA.

This paper is organised as follows. Section 2 recaps some fundamental concepts required later. Section 3 introduces fuzzy synchronous languages and defines some suitable operators over them. Section 4 introduces a method for constructing

the synchronous product of two nondeterministic automata with fuzzy transitions. Moreover, it is proved that the algebra constituted by any set of fuzzy synchronous languages and the signature previously defined forms a SKA. Finally, Section 5 concludes and enumerates some topics for future research.

## 2 Preliminaries

**Definition 2.1 (Kleene algebra)** *A* Kleene algebra $(K, +, \cdot, ^*, \mathbf{0}, \mathbf{1})$ *is an idempotent semiring with an extra unary operator* $^*$ *satisfying the axioms* (1) − (13) *of Fig. 2. A partial order* $\leq$ *is defined as* $\alpha \leq \beta \Leftrightarrow \alpha + \beta = \beta$.

The operators $+$, $\cdot$ and $*$ are typically understood as *nondeterministic choice*, *sequence* and *iteration*, respectively. Actions $\mathbf{0}$ and $\mathbf{1}$ represent *fail* and *skip*, respectively.

Well-known examples of Kleene algebras are the algebra of binary relations over a set $X$, the set of all languages over $\Sigma^*$, and the $(min, +)$ algebra, also known as the tropical algebra.

Extending the original definition with an operation $\times$ to capture the synchronous execution of actions[5] lead to the notion of a *synchronous Kleene algebra* [18].

**Definition 2.2 (Synchronous Kleene algebra)** *A synchronous Kleene algebra (SKA) is a tuple*

$$\mathbf{S} = (\mathcal{A}, +, \cdot, \times, ^*, \mathbf{0}, \mathbf{1}, \mathcal{A}_B)$$

*where* $\mathcal{A}_B$ *is a finite discrete set of* basic actions *and* $A$ *a (possible infinite) set of* composed actions*, satisfying the axioms in Fig. 2.*

The sets of actions $\mathcal{A}$ and $\mathcal{A}_B$ are structured by $\mathcal{A}_B \subseteq \mathcal{A}_B^{\times} \subset \mathcal{A}$, where $\mathcal{A}_B$ is the set of basic actions and $\mathcal{A}_B^{\times}$ is its closure under $\times$.

We denote by $T_{SKA}$ the term algebra of SKA, generated by the grammar:

$$\alpha ::= a_b \mid \mathbf{0} \mid \mathbf{1} \mid \alpha + \alpha \mid \alpha \cdot \alpha \mid \alpha \times \alpha \mid \alpha^*$$

where $a_b \in A_B$. Following a common practice, we write $a_b b_b$, rather than $a_b \cdot b_b$, for $a_b$, $b_b \in A_B$. The elements of $\mathcal{A}_B^{\times}$ are called $\times$-*actions* (e.g. $a$, $a \times b \in \mathcal{A}_B^{\times}$ but $a + b$, $a \times b + c$, $\mathbf{0}$, $\mathbf{1} \notin \mathcal{A}_B^{\times}$).

**Definition 2.3 (Complete Heyting algebra)** *A complete Heyting algebra (CHA) is a tuple*

$$\mathcal{H} = (H, +, ;, \mathbf{0}, \mathbf{1}, \rightarrow)$$

*which satisfies axioms (1)-(9) in Fig. 2, replacing* $\cdot$ *by* ; *and, additionally, the following axioms:*

[5] Following [18], the symbol $\times$ stands for the synchronous product; any possible confusion with the same symbol used for Cartesian product is desambiguated by context.

$$+(\beta+\gamma) = (\alpha+\beta)+\gamma \tag{1}$$
$$\alpha+\beta = \beta+\alpha \tag{2}$$
$$\alpha+\alpha = \alpha \tag{3}$$
$$\alpha+\mathbf{0} = \mathbf{0}+\alpha = \alpha \tag{4}$$
$$\alpha\cdot(\beta\cdot\gamma) = (\alpha\cdot\beta)\cdot\gamma \tag{5}$$
$$\alpha\cdot\mathbf{1} = \mathbf{1}\cdot\alpha = \alpha \tag{6}$$
$$\alpha\cdot(\beta+\gamma) = (\alpha\cdot\beta)+(\alpha\cdot\gamma) \tag{7}$$
$$(\alpha+\beta)\cdot\gamma = (\alpha\cdot\gamma)+(\beta\cdot\gamma) \tag{8}$$
$$\alpha\cdot\mathbf{0} = \mathbf{0}\cdot\alpha = \mathbf{0} \tag{9}$$
$$\mathbf{1}+(\alpha\cdot\alpha^*) = \alpha^* \tag{10}$$
$$\mathbf{1}+(\alpha^*\cdot\alpha) = \alpha^* \tag{11}$$
$$\beta+\alpha\cdot\gamma\leq\gamma \Rightarrow \alpha^*\cdot\beta\leq\gamma \tag{12}$$
$$\beta+\gamma\cdot\alpha\leq\gamma \Rightarrow \beta\cdot\alpha^*\leq\gamma \tag{13}$$
$$\alpha\times(\beta\times\gamma) = (\alpha\times\beta)\times\gamma \tag{14}$$
$$\alpha\times\beta = \beta\times\alpha \tag{15}$$
$$\alpha\times\mathbf{1} = \mathbf{1}\times\alpha = \alpha \tag{16}$$
$$\alpha\times\mathbf{0} = \mathbf{0}\times\alpha = \mathbf{0} \tag{17}$$
$$a_b\times a_b = a_b \qquad a_b\in\mathcal{A}_B \tag{18}$$
$$\alpha\times(\beta+\gamma) = (\alpha\times\beta)+(\alpha\times\gamma) \tag{19}$$
$$(\alpha+\beta)\times\gamma = (\alpha\times\gamma)+(\beta\times\gamma) \tag{20}$$
$$(\alpha_\times\cdot\alpha)\times(\beta_\times\cdot\beta) = (\alpha_\times\times\beta_\times)\cdot(\alpha\times\beta) \qquad \alpha_\times,\beta_\times\in\mathcal{A}_B^\times \tag{21}$$

Fig. 2. Axiomatisation of SKA from [18].

$$h_1;h_2 = h_2;h_1 \tag{22}$$
$$h;h = h \tag{23}$$
$$h_1+(h_1;h_2) = h_1 \tag{24}$$
$$h_1;h_2\leq h_3 \Leftrightarrow h_2\leq h_1\to h_3 \tag{25}$$
$$h;\left(\sum_{i\in I}h_i\right) = \sum_{i\in I}(h;h_i) \tag{26}$$
$$\left(\sum_{i\in I}h_i\right);h = \sum_{i\in I}(h_i;h) \tag{27}$$

*with $\Sigma$ denoting the iterated version of the associative operator $+$, and $I$ being an (possible infinite) index set.*

We assume $\mathcal{H}$ to be complete to ensure that all suprema exist when characterising operators $\cdot$, $\times$ and $^*$ on fuzzy synchronous languages as (possible) infinite sums. Such property, together with axiom (25) ensure that every suprema distributes over arbitrary infima, which is used to prove Theorem 4.6. The examples below illustrate

this structure.

**Example 2.4 (2- the Boolean algebra)** *A first example is the well-known binary structure*

$$\mathbf{2} = (\{\top, \bot\}, \vee, \wedge, \bot, \top, \rightarrow)$$

*with the standard interpretation of Boolean connectives.*

**Example 2.5** *A second example is the three-valued Gödel chain, which introduces an explicit denotation u for "unknown" (or "undefined").*

$$\mathbf{G_3} = (\{\top, u, \bot\}, \vee, \wedge, \bot, \top, \rightarrow)$$

*where*

| $\vee$ | $\bot$ | $u$ | $\top$ |
|---|---|---|---|
| $\bot$ | $\bot$ | $u$ | $\top$ |
| $u$ | $u$ | $u$ | $\top$ |
| $\top$ | $\top$ | $\top$ | $\top$ |

| $\wedge$ | $\bot$ | $u$ | $\top$ |
|---|---|---|---|
| $\bot$ | $\bot$ | $\bot$ | $\bot$ |
| $u$ | $\bot$ | $u$ | $u$ |
| $\top$ | $\bot$ | $u$ | $\top$ |

| $\rightarrow$ | $\bot$ | $u$ | $\top$ |
|---|---|---|---|
| $\bot$ | $\top$ | $\top$ | $\top$ |
| $u$ | $u$ | $\top$ | $\top$ |
| $\top$ | $\bot$ | $u$ | $\top$ |

**Example 2.6 (Gödel algebra)** *Another example is given by the standard Gödel algebra*

$$\mathbf{G} = ([0,1], max, min, 0, 1, \rightarrow)$$

*where*

$$x \rightarrow y = \begin{cases} 1, & \text{if } x \leq y \\ y, & \text{if } y < x \end{cases}$$

## 3 Fuzzy synchronous languages

This section introduces a notion of *fuzzy synchronous language*, based on C. Prisacariu proposal for the crisp synchronous case [18]. A number of operations over fuzzy synchronous languages are also defined. Finally, a map that interprets each term of $T_{SKA}$ as a fuzzy synchronous language is formalised. The reader is referred to any classical introduction to fuzzy logic, e.g. [1], for the standard definitions of fuzzy sets and fuzzy languages used in the sequel.

**Definition 3.1 ($\mathcal{H}$-Fuzzy synchronous language)** *Let $\mathcal{A}_B$ be a set of basic actions and $\mathcal{H}$ a CHA over carrier $H$, and $\Sigma = \mathcal{P}(\mathcal{A}_B) \setminus \{\emptyset\}$ the alphabet of all the nonempty subsets of $\mathcal{A}_B$ (denoted by $x, y$). Sequences $u, v, \ldots \in \Sigma^*$ are called* $\mathcal{A}_B$-synchronous strings*, with notation $\epsilon$ standing for the empty string. A $\mathcal{H}$-fuzzy synchronous language over $\mathcal{A}_B$ is an element of $\mathcal{H}^{\Sigma^*}$, i.e. a function $\mathcal{L} : \Sigma^* \rightarrow H$.*

We can then generalise for this setting, the standard operators from regular language theory. For any $\mathcal{H}$-fuzzy synchronous languages $\mathcal{L}$, $\mathcal{L}_1$, $\mathcal{L}_2$, and for all $w \in \Sigma^*$, we define the following operations:

- $\varnothing(w) = \mathbf{0}$, for all $w \in \Sigma^*$

- $\chi(w) = \begin{cases} \mathbf{1} & \text{if } w = \epsilon \\ \mathbf{0} & \text{otherwise} \end{cases}$
- $(\mathcal{L}_1 \cup \mathcal{L}_2)(w) = \mathcal{L}_1(w) + \mathcal{L}_2(w)$
- $(\mathcal{L}_1 \cdot \mathcal{L}_2)(w) = \sum_{u,v} \mathcal{L}_1(u); \mathcal{L}_2(v)$, with $w = uv$ standing for the concatenation of strings $u$ and $v$
- $\mathcal{L}^*(w) = \sum_{i \geq 0} \mathcal{L}_1^i(w)$, with $\mathcal{L}^0(w) = \chi(w)$, $\mathcal{L}^{(i+1)}(w) = (\mathcal{L} \cdot \mathcal{L}^i)(w)$
- $(\mathcal{L}_1 \times \mathcal{L}_2)(w) = \sum_{u,v} \mathcal{L}_1(u); \mathcal{L}_2(v)$, with $w = u \times v$ defined by
  - $u \times \epsilon = u = \epsilon \times u$
  - $u \times v = (x \cup y)(u' \times v')$ where $u = xu'$ and $v = yv'$, with $x, y \in \Sigma$.

One may notice that the expressions that define operators $\cdot$ and $\times$ seem related. Note, however, that operator $\cdot$ ranges over all possible ways to construct the word $w$ by concatenation of the smaller words $u$ and $v$, while operator $\times$ looks over all the possible constructions by "classical" synchronous product of words $u \times v$ defined above.

**Definition 3.2 (Basic $\mathcal{H}$-fuzzy and $\times$ – $\mathcal{H}$-fuzzy synchronous languages)** *A* basic $\mathcal{H}$-fuzzy synchronous language*, denoted by $\mathcal{L}_B$, is a $\mathcal{H}$-fuzzy synchronous language such that $\mathcal{L}_B(w) = \mathbf{0}$ whenever $w \notin \mathcal{A}_B$. A* $\times$ – $\mathcal{H}$-fuzzy synchronous language*, denoted by $\mathcal{L}^\times$, is a $\mathcal{H}$-fuzzy synchronous language such that $\mathcal{L}^\times(w) = \mathbf{0}$ whenever $w \notin \mathcal{A}_B^\times$.*

Note that $\mathcal{A}_B^\times$ does not contain any action built from operator $\cdot$ (e.g. for $\mathcal{A}_B = \{a, b, c\}$, $abc \notin \mathcal{A}_B^\times$).
Without loss of generality, we write $a_b$ for the singleton set $\{a_b\}$, for any $a_b \in \mathcal{A}_B$. Moreover, expression $a_1 \dots a_n$, for $n \geq 1$ will denote in the sequel a synchronous string where $a_i \in \Sigma$, with $1 \leq i \leq n$.

Similarly to the homomorphism used to interpret SKA as synchronous sets [18], we define a map to interpret term actions of $\alpha \in T_{SKA}$ as $\mathcal{H}$-fuzzy synchronous languages.

**Definition 3.3 (Fuzzy interpretation)** *Consider a map $FI_{SKA} : \mathcal{A}_B \cup \{\mathbf{0}, \mathbf{1}\} \to H^{\Sigma^*}$ such that*

- $FI_{SKA}(a_b) = \mathcal{L}_B$
- $FI_{SKA}(\mathbf{0}) = \varnothing$
- $FI_{SKA}(\mathbf{1}) = \chi$

*where $\mathcal{L}_B$ is a basic $\mathcal{H}$-fuzzy synchronous language such that $\mathcal{L}_B(w) = \mathbf{0}$ for all $w \neq a_b$.*
*Its extension $\widehat{FI}_{SKA} : T_{SKA} \to H^{\Sigma^*}$ over the term algebra is called a* fuzzy interpretation *of SKA and defined as*

$$\widehat{FI}_{SKA}(\alpha) = FI_{SKA}(\alpha), \forall \alpha \in \mathcal{A}_B \cup \{\mathbf{0}, \mathbf{1}\}$$
$$\widehat{FI}_{SKA}(\alpha + \beta) = \widehat{FI}_{SKA}(\alpha) \cup \widehat{FI}_{SKA}(\beta)$$
$$\widehat{FI}_{SKA}(\alpha \cdot \beta) = \widehat{FI}_{SKA}(\alpha) \cdot \widehat{FI}_{SKA}(\beta)$$

$\widehat{FI}_{SKA}(\alpha \times \beta) = \widehat{FI}_{SKA}(\alpha) \times \widehat{FI}_{SKA}(\beta)$
$\widehat{FI}_{SKA}(\alpha^*) = \widehat{FI}_{SKA}(\alpha)^*$

# 4 Synchronous product of fuzzy automata

This section presents our main results. First a new type of fuzzy automata is defined on top of a CHA which models the space of possible membership values for fuzzy transitions. An appropriate notion of a synchronous product for these sort of automata then is presented. The section ends with the generalisation of two classical results:

- for every term $\alpha$ of $T_{SKA}$, there is a $\mathcal{H}$-NFA which accepts precisely $\widehat{FI}_{SKA}(\alpha)$;
- given a $\mathcal{H}$-NFA $\mathcal{M}$, there is a function $f$ mapping $\mathcal{M}$ to $\alpha$ of SKA such that $\widehat{FI}_{SKA}(\alpha) = \mathcal{L}(\mathcal{M})$.

**Definition 4.1 (Nondeterministic automata with fuzzy transitions)** *For a CHA over $H$, a set $\mathcal{A}_B$ of basic actions, a* nondeterministic finite-state automaton with fuzzy transitions ($\mathcal{H}$-NFA) *is a tuple $\mathcal{M} = (X, \Sigma, x_0, F, \delta)$ where:*

- *$X$ is a finite set of states;*
- *$\Sigma = \mathcal{P}(\mathcal{A}_B) \setminus \{\emptyset\}$ is the input alphabet (i.e. the powerset of the set of basic actions minus the empty set);*
- *$x_0$ is the initial state;*
- *$F$ is the set of final states;*
- *$\delta : X \times \Sigma \times X \to H$ is the fuzzy transition function.*

Intuitively, $\delta(x_1, a, x_2)$, for $a \in \Sigma$, can be interpreted as the truth degree of "*input $a$ causing a transition from $x_1$ to $x_2$*".

The fuzzy transition relation can be inductively extended to the free monoid $\Sigma^*$ over $\Sigma$ through a function $\delta^* : X \times \Sigma^* \times X \to H$ such that, for any $x_1, x_2 \in X$,

$$\delta^*(x_1, \epsilon, x_2) = \begin{cases} \mathbf{1} & \text{if } x_1 = x_2 \\ \mathbf{0} & \textit{otherwise} \end{cases}$$

and, for any $x_1, x_2 \in X, w \in \Sigma^*$ and $a \in \Sigma$,

$$\delta^*(x_1, aw, x_2) = \sum_{x' \in X} \delta(x_1, a, x'); \delta^*(x', w, x_2)$$

For any states $x_1, x_2 \in X$ and any word $w \in \Sigma^*$, $\delta^*(x_1, w, x_2)$ can be interpreted as the truth degree of "*word $w$ causes a transition from $x_1$ to $x_2$*".

Given a residuated lattice $\mathbf{A}$ with support set $A$, a fuzzy language over an alphabet $\Sigma$ is classically defined as a fuzzy subset of $\Sigma^*$, that is, a function $\lambda : \Sigma^* \to A$ [9]. Thus,

**Definition 4.2** *Given a CHA over $H$ and a $\mathcal{H}$-NFA $\mathcal{M} = (X, \Sigma, x_0, F, \delta)$, the fuzzy synchronous language recognised by $\mathcal{M}$ is a function $\mathcal{L}(\mathcal{M}) : \Sigma^* \to H$ defined as*

$$\mathcal{L}(\mathcal{M})(w) = \sum_{x \in F} \delta^*(x_0, w, x)$$

*for* $w \in \Sigma^*$.

We can interpret $\mathcal{L}(\mathcal{M})(w)$ as the truth degree of "*the word* $w$ *causes a transition from an initial state to a final state in* $\mathcal{M}$". $\mathcal{L}(\mathcal{M})(w)$ is the degree of recognition of $w$ by $\mathcal{M}$.

Now we prove a Kleene theorem for $\mathcal{H}$-NFA and fuzzy synchronous languages. The proof proceeds by taking a class of $\mathcal{H}$-NFA denoted $\mathcal{M}_\alpha$ whenever the automaton is a $\mathcal{H}$-NFA associated to an action $\alpha \in T_{SKA}$.

**Theorem 4.3** *For any action* $\alpha \in SKA$ *there exists a* $\mathcal{H}$*-NFA* $\mathcal{M}_\alpha$ *which accepts precisely* $\widehat{FI}_{SKA}(\alpha)$.

**Proof.** We construct a $\mathcal{H}$-NFA $\mathcal{M}_\alpha$ for each regular expression built from a basic action $a_b \in \mathcal{A}_B$ and operators $+$, $\cdot$ and $^*$. Then we provide a construction similar to the one in [18] for the synchronous operator $\times$. Each transition of the automaton is labelled by a pair $(\alpha, \delta(x_i, \alpha, x_j))$, $0 \leq i, j \leq n$ where $\alpha \in T_{SKA}$ is the action relating to the input that causes a transition between states $x_i$ and $x_j$, and $\delta(x_i, \alpha, x_j) \in \mathcal{H}$ the is "weight" of the transition. Slightly abusing the notation, let $\alpha \in T_{SKA}$ represent the input of the automaton that relates to action $\alpha$, a convention that allows a clearer presentation of the inductive proof. Thus,

*Base case*:

The automata corresponding to $a_b \in \mathcal{A}_B$, $\mathbf{0}$ and $\mathbf{1}$, i.e. $\mathcal{M}_{a_b}$, $\mathcal{M}_{\mathbf{0}}$ and $\mathcal{M}_{\mathbf{1}}$, are depicted in Figure 3 from top to bottom, respectively. By Definition 4.2 it is easy to see that the fuzzy synchronous language recognized by each one of these automata coincides precisely with $\widehat{FI}_{SKA}(a_b)$, $\widehat{FI}_{SKA}(\mathbf{0})$ and $\widehat{FI}_{SKA}(\mathbf{1})$, respectively.

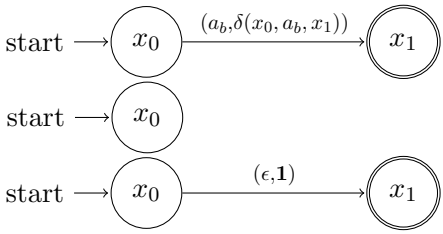


Fig. 3. Automata representing actions $a \in \mathcal{A}_B$, $\mathbf{0}$ and $\mathbf{1}$.

By Definition 4.2, the fuzzy synchronous language recognized by $\mathcal{M}_{a_b}$ is given by

$$\mathcal{L}(\mathcal{M}_{a_b})(a_b) = \delta^*(x_0, a_b, x_1) = \delta(x_0, a_b, x_1)$$

and $\mathcal{L}(\mathcal{M}_{a_b})(w) = \mathbf{0}$, for all $w \neq a_b$. Thus, $\mathcal{L}(\mathcal{M}_{a_b}) = \widehat{FI}_{SKA}(a_b)$. The fuzzy synchronous language recognized by $\mathcal{M}_{\mathbf{0}}$ is given by $\mathcal{L}(\mathcal{M}_{\mathbf{0}})(w) = \mathbf{0}$, for all $w \in \Sigma^*$. That is exactly $\widehat{FI}_{SKA}(\mathbf{0})$.

Analogously, the fuzzy synchronous language recognized by $\mathcal{M}_{\mathbf{1}}$ is defined as $\mathcal{L}(\mathcal{M}_{\mathbf{1}})(\epsilon) = \mathbf{1}$ and
$\mathcal{L}(\mathcal{M}_{\mathbf{1}})(w) = \mathbf{0}$, for all $w \neq \epsilon$. Clearly $\mathcal{L}(\mathcal{M}_{\mathbf{1}}) = \widehat{FI}_{SKA}(\mathbf{1})$.

*Inductive case*:

The automata $\mathcal{M}_{\alpha+\beta}$, $\mathcal{M}_{\alpha\cdot\beta}$ and $\mathcal{M}_{\alpha^*}$ depicted in Figures 4, 5 and 6, respectively, correspond to terms $\alpha+\beta$, $\alpha\cdot\beta$ and $\alpha^*$. Their construction is the standard one [6].

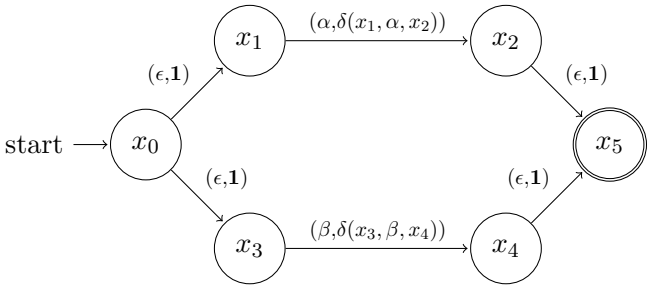


Fig. 4. Automata representing action $\alpha+\beta$.

The fuzzy synchronous language recognized by $\mathcal{M}_{\alpha+\beta}$ is given by:

$$\begin{aligned}\mathcal{L}(\mathcal{M}_{\alpha+\beta})(\epsilon\alpha\epsilon) &= \delta^*(x_0,\epsilon\alpha\epsilon,x_5) = \delta(x_0,\epsilon,x_1);\delta^*(x_1,\alpha\epsilon,x_5) + \delta(x_0,\epsilon,x_3);\delta^*(x_3,\alpha\epsilon,x_5)\\ &= \mathbf{1};\delta^*(x_1,\alpha,x_2);\delta(x_2,\epsilon,x_5) + \mathbf{1};\delta^*(x_3,\alpha,x_4);\delta(x_4,\epsilon,x_5)\\ &= \delta^*(x_1,\alpha,x_2);\mathbf{1}+\mathbf{0};\mathbf{1} = \delta^*(x_1,\alpha,x_2)\end{aligned}$$

and analogously for word $\epsilon\beta\epsilon$,

$$\mathcal{L}(\mathcal{M}_{\alpha+\beta})(\epsilon b\epsilon) = \delta^*(x_2,\beta,x_3)$$

On the other hand, $\widehat{FI}_{SKA}(\alpha+\beta) = \mathcal{L}_\alpha \cup \mathcal{L}_\beta$ and $(\mathcal{L}_\alpha \cup \mathcal{L}_\beta)(\alpha) = \delta^*(x_1,\alpha,x_2)$, $(\mathcal{L}_\alpha \cup \mathcal{L}_\beta)(\beta) = \delta^*(x_2,\beta,x_3)$ and $(\mathcal{L}_\alpha \cup \mathcal{L}_\beta)(w) = \mathbf{0}$ for $w \neq \alpha,\beta$. Thus, $\mathcal{L}(\mathcal{M}_{\alpha+\beta}) = \widehat{FI}_{SKA}(\alpha+\beta)$.

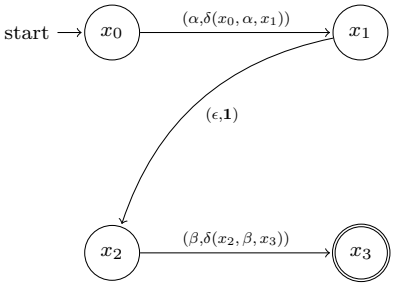


Fig. 5: Automata representing action $\alpha\cdot\beta$.

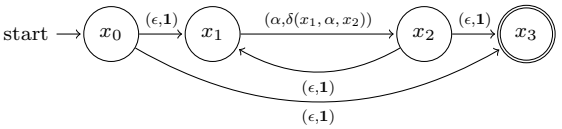


Fig. 6: Automata representing action $\alpha^*$.

The fuzzy synchronous language recognised by $\mathcal{M}_{\alpha\cdot\beta}$ is defined as

$$\begin{aligned}\mathcal{L}(\mathcal{M}_{\alpha\cdot\beta})(\alpha\epsilon\beta) &= \delta^*(x_0,\alpha\epsilon\beta,x_3) = \delta^*(x_0,\alpha,x_1);\delta^*(x_1,\epsilon b,x_3)\\ &= \delta^*(x_0,\alpha,x_1);\delta(x_1,\epsilon,x_2);\delta^*(x_2,\beta,x_3) = \delta^*(x_0,\alpha,x_1);\mathbf{1};\delta^*(x_2,\beta,x_3)\\ &= \delta^*(x_0,\alpha,x_1);\delta^*(x_2,\beta,x_3)\end{aligned}$$

Analogously as before, $\widehat{FI}_{SKA}(\alpha\cdot\beta) = \mathcal{L}_\alpha\cdot\mathcal{L}_\beta$ with $(\mathcal{L}_\alpha\cdot\mathcal{L}_\beta)(w) = \delta^*(x_0,\alpha,x_1);\delta^*(x_2,\beta,x_3)$ if $w=\alpha\cdot\beta$ and $\mathbf{0}$ otherwise. Hence, $\mathcal{L}(\mathcal{M}_{\alpha\cdot\beta}) = \widehat{FI}_{SKA}(\alpha\cdot\beta)$.

Finally, automaton $\mathcal{M}_{\alpha^*}$ recognizes the fuzzy synchronous language given by

$$\begin{aligned}\mathcal{L}(\mathcal{M}_{\alpha^*})(\epsilon\alpha\alpha^*\epsilon) &= \delta(x_0,\epsilon,x_1);\delta^*(x_1,\alpha\alpha^*\epsilon,x_3) = \mathbf{1};\delta^*(x_1,\alpha,x_2);\delta^*(x_2,\alpha^*\epsilon,x_3)\\ &= \delta^*(x_1,\alpha,x_2);\delta^*(x_2,\alpha^*,x_2);\delta(x_2,\epsilon,x_3) = \delta^*(x_1,\alpha,x_2);\delta^*(x_2,\alpha^*,x_2);\mathbf{1}\\ &= \delta^*(x_1,\alpha,x_2);\delta^*(x_2,\alpha^*,x_2)\end{aligned}$$

and for word $\epsilon$, $\mathcal{L}(\mathcal{M}_{\alpha^*})(\epsilon) = \delta(x_0,\epsilon,x_3) = \mathbf{1}$. $\widehat{FI}_{SKA}(\alpha^*) = \widehat{FI}_{SKA}(\alpha)^* = \mathcal{L}_\alpha^*$ where

$$\begin{aligned}\mathcal{L}_\alpha^*(w) &= \sum_{i\geq 0}\mathcal{L}_\alpha^i(w) = \chi(w)+\mathcal{L}_\alpha(w)+\mathcal{L}_\alpha^2(w)+\ldots\\ &= \mathcal{L}_\alpha(\alpha\alpha^*)+\mathcal{L}_\alpha^2(\alpha\alpha^*)+\ldots = \mathcal{L}_\alpha(\alpha\alpha^*)+\mathcal{L}_\alpha(\alpha\alpha^*);\mathcal{L}_\alpha(\alpha\alpha^*)+\ldots\\ &= \mathcal{L}_\alpha(\alpha\alpha^*) = \mathcal{L}_\alpha(\alpha);\mathcal{L}_\alpha(\alpha^*) = \delta^*(x_1,\alpha,x_2);\delta^*(x_2,\alpha^*,x_2)\end{aligned}$$

$\mathcal{L}_\alpha^*(w) = \chi(\epsilon) = \mathbf{1}$ if $w=\epsilon$ and $\mathbf{0}$ otherwise. Therefore, $\mathcal{L}(\mathcal{M}_{\alpha^*}) = \widehat{FI}_{SKA}(\alpha^*)$.

□

The *synchronous product* of two $\mathcal{H}$-NFA $\mathcal{M}_\alpha = (X^\alpha, \mathcal{P}(\mathcal{A}_B^\alpha)\setminus\{\emptyset\}, x_0^\alpha, F^\alpha, \delta^\alpha)$ and
$\mathcal{M}_\beta = (X^\beta, \mathcal{P}(\mathcal{A}_B^\beta)\setminus\{\emptyset\}, x_0^\beta, F^\beta, \delta^\beta)$ is

$$\mathcal{M}_{\alpha\times\beta} = (X^\alpha\times X^\beta, \mathcal{P}(\mathcal{A}_{\mathcal{B}}^\alpha\cup\mathcal{A}_{\mathcal{B}}^\beta)\setminus\{\emptyset\}, (x_0^\alpha,x_0^\beta), F^\alpha\times F^\beta, \delta^{\alpha\times\beta})$$

where

$$\delta^{\alpha\times\beta} : (X^\alpha\times X^\beta)\times(\mathcal{P}(\mathcal{A}_B^\alpha\cup\mathcal{A}_B^\beta)\setminus\{\emptyset\})\times(X^\alpha\times X^\beta)\to H$$

is defined, for $u\in\mathcal{P}(\mathcal{A}_B^\alpha)\setminus\{\emptyset\}$ and $v\in\mathcal{P}(\mathcal{A}_B^\beta)\setminus\{\emptyset\}$, $w=u\cup v$, by

$$\delta^{\alpha\times\beta}((x^\alpha,x^\beta),w,(y^\alpha,y^\beta)) = \begin{cases}\delta^\alpha(x^\alpha,u,y^\alpha) & \text{if } x^\beta=y^\beta\in F^\beta\\ \delta^\beta(x^\beta,v,y^\beta) & \text{if } x^\alpha=y^\alpha\in F^\alpha\\ \sum_{u,v}\delta^\alpha(x^\alpha,u,y^\alpha);\delta^\beta(x^\beta,v,y^\beta) & \text{otherwise}\end{cases}$$

The corresponding construction is illustrated in Fig. 7.

**Definition 4.4** *Let* $\mathcal{M}_\alpha = (X^\alpha, \mathcal{P}(\mathcal{A}_B^\alpha)\setminus\{\emptyset\}, x_0^\alpha, F^\alpha, \delta^\alpha)$ *and* $\mathcal{M}_\beta = (X^\beta, \mathcal{P}(\mathcal{A}_B^\beta)\setminus\{\emptyset\}, x_0^\beta, F^\beta, \delta^\beta)$ *be two* $\mathcal{H}$*-NFA and* $\mathcal{M}_{\alpha\times\beta} = (X^\alpha\times X^\beta, \mathcal{P}(\mathcal{A}_{\mathcal{B}}^\alpha\cup\mathcal{A}_{\mathcal{B}}^\beta)\setminus$

$\{\emptyset\}, (x_0^\alpha, x_0^\beta), F^\alpha \times F^\beta, \delta^{\alpha\times\beta})$ *its synchronous product. The fuzzy synchronous language recognised by* $\mathcal{M}_{\alpha\times\beta}$ *is the function* $\mathcal{L}(M_{\alpha\times\beta}) : \mathcal{P}(\mathcal{A}_B^\alpha \cup \mathcal{A}_B^\beta)\setminus\{\emptyset\} \to H$ *defined by*

$$\mathcal{L}(\mathcal{M}_{\alpha\times\beta})(w) = \sum_{\substack{x_f^\alpha \in F^\alpha \\ x_f^\beta \in F^\beta}} \left(\delta^{\alpha\times\beta}\right)^*((x_0^\alpha, x_0^\beta), w, (x_f^\alpha, x_f^\beta)).$$

Analogously to other cases, we prove that $\mathcal{M}_{\alpha\times\beta}$ recognizes the fuzzy synchronous language $\widehat{FI}_{SKA}(\alpha \times \beta)$:

$$\begin{aligned}\mathcal{L}(\mathcal{M}_{\alpha\times\beta})(\alpha\times\beta) &= (\delta^{\alpha\times\beta})^*((x_0^\alpha, x_0^\beta), \alpha\times\beta, (x_f^\alpha, x_f^\beta)) = \delta^{\alpha\times\beta}((x_0^\alpha, x_0^\beta), \alpha\times\beta, (x_f^\alpha, x_f^\beta)) \\ &= \delta^\alpha(x_0^\alpha, \alpha, x_f^\alpha); \delta^\beta(x_0^\beta, \beta, x_f^\beta)\end{aligned}$$

But $\widehat{FI}_{SKA}(\alpha\times\beta) = \mathcal{L}_\alpha \times \mathcal{L}_\beta$ such that $(\mathcal{L}_\alpha \times \mathcal{L}_\beta)(w) = \delta^\alpha(x_0^\alpha, \alpha, x_f^\beta); \delta^\beta(x_0^\alpha, \beta, x_f^\beta)$ if $w = \alpha \times \beta$ and $\mathbf{0}$ otherwise. Hence, $\mathcal{L}(\mathcal{M}_{\alpha\times\beta})(\alpha\times\beta) = \widehat{FI}_{SKA}(\alpha\times\beta)$.

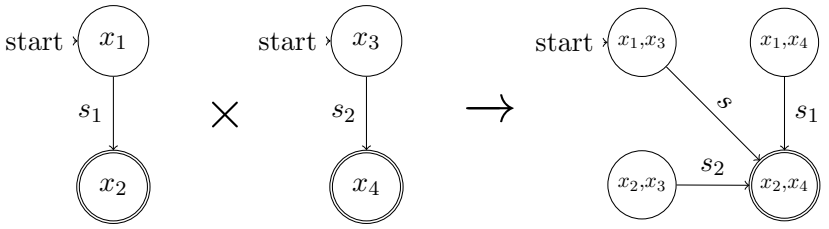


Fig. 7. Example of the automaton construction corresponding to $\alpha \times \beta$.

where $s_1$ denotes the label $(\alpha,\delta^\alpha(x_1, \alpha, x_2))$, $s_2$ the label $(\beta,\delta^\beta(x_3, \beta, x_4))$ and $s$ the label
$(\alpha\times\beta, \delta^{\alpha\times\beta}((x_1, x_3), \alpha\times\beta, (x_2, x_4)))$ corresponding to the synchronous action $\alpha\times\beta$.

The proof of completeness of SKA w.r.t. the fuzzy interpretation proceeds by eliminating states which generates a regular expression. Consider a function $f$ which takes a $\mathcal{H}$-NFA $\mathcal{M}_\alpha$ and returns an action $\alpha \in SKA$. The weight associated with this action is computed accordingly, depending on the weight of each transition of the automaton. Note that this procedure considers actions of SKA as labels for the automaton transitions, rather than as elements of the input alphabet $\Sigma$.

**Theorem 4.5** *For all* $\alpha \in T_{SKA}$, $f(\mathcal{M}_\alpha)$ *results in an action* $\alpha$ *such that* $\widehat{FI}_{SKA}(\alpha) = \mathcal{L}(\mathcal{M}_\alpha)$.

**Proof.** The proof uses induction on the structure of the actions.

*Base case*:

Let us consider the automata $\mathcal{M}_a$, $\mathcal{M}_\mathbf{0}$ and $\mathcal{M}_\mathbf{1}$ of Figure 3. Applying $f$, we obtain the actions $a$, $\mathbf{0}$ and $\mathbf{1}$ with weights $\delta(x_0, a, x_1)$, $\mathbf{0}$ and $\mathbf{1}$, respectively.

*Inductive case*:

*Case* $\alpha = \alpha_1 + \alpha_2$. The automaton $\mathcal{M}_{\alpha_1+\alpha_2}$ is obtained with the construction for $+$ of Theorem 4.3 from the automata $\mathcal{M}_{\alpha_1}$ and $\mathcal{M}_{\alpha_2}$. Then, $f$ eliminates states

$x_1$ and $x_2$, obtaining a single transition labelled with the action $1 \cdot \alpha_1 \cdot 1 \equiv \alpha_1$, with weight $\mathbf{1}; \delta(x_1, \alpha_1, x_2); \mathbf{1} = \delta(x_1, \alpha_1, x_2)$, and states 3 and 4 obtaining a single transition labelled by the action $1 \cdot \alpha_2 \cdot 1 \equiv \alpha_2$ with weight $\mathbf{1}; \delta(x_3, \alpha_2, x_4); \mathbf{1} = \delta(x_3, \alpha_2, x_4)$. Finally it combines the two transitions into one labelled by the action $\alpha_1 + \alpha_2$ with weight $\delta(x_1, \alpha_1, x_2) + \delta(x_3, \alpha_2, x_4)$.

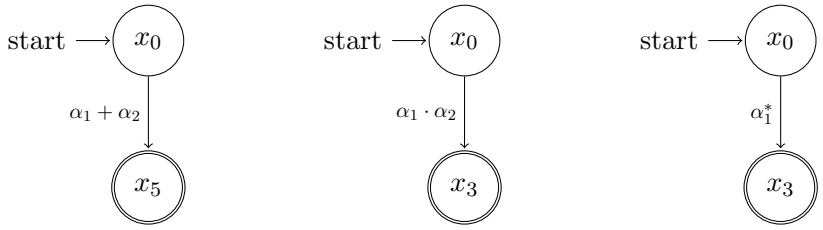


Fig. 8. Application of $f$ in automata $\mathcal{M}_{\alpha+\beta}$, $\mathcal{M}_{\alpha\cdot\beta}$ and $\mathcal{M}_{\alpha^*}$.

*Case* $\alpha = \alpha_1 \cdot \alpha_2$.

The automaton $\mathcal{M}_{\alpha_1\cdot\alpha_2}$ is obtained from $\mathcal{M}_{\alpha_1}$ and $\mathcal{M}_{\alpha_2}$ by the process of Theorem 4.3. By eliminating intermediate states $x_1$ and $x_2$ we obtain a single transition labelled $\alpha_1 \cdot \mathbf{1} \cdot \alpha_2 \equiv \alpha_1 \cdot \alpha_2$ with weight $\delta(x_0, \alpha_1, x_1); \mathbf{1}; \delta(x_2, \alpha_2, x_3) = \delta(x_0, \alpha_1, x_1); \delta(x_2, \alpha_2, x_3)$.

*Case* $\alpha = \alpha_1^*$.

Using the same procedure, $f$ eliminates states $x_1$ and $x_2$ of $\mathcal{M}_{\alpha_1^*}$, obtaining an automaton with a single transition labelled by $\mathbf{1} \cdot \alpha_1 \cdot (\mathbf{1} \cdot \alpha_1)^* \cdot \mathbf{1} + \mathbf{1} \equiv \alpha_1^*$ with weight $\mathbf{1}; \delta(x_1, \alpha_1, x_2); (\mathbf{1}; \delta(x_1, \alpha_1^*, x_2)); \mathbf{1} + \mathbf{1} = \delta(x_1, \alpha_1, x_2); \delta(x_1, \alpha_1^*, x_2) + \mathbf{1}$.
The resulting automata obtained by the procedure of the cases above are shown in Figure 8.

*Case* $\alpha = \alpha_1 \times \alpha_2$.

Analogously, function $f$ eliminates states $(x_1, x_4)$ and $(x_2, x_3)$, obtaining an automaton with a single transition labelled by $\alpha \times \beta$ with weight $\delta^{\alpha_1}(x_1, \alpha_1, x_2); \delta^{\alpha_2}(x_3, \alpha_2, x_4) = \delta^{\alpha_1 \times \alpha_2}((x_1, x_3), \alpha_1 \times \alpha_2, (x_2, x_4))$.

□

Next we characterise the set of fuzzy synchronous languages as a SKA.

**Theorem 4.6** *Any set of fuzzy synchronous languages containing* $\varnothing$ *and* $\chi$ *and closed under the operations of Definition 3.1 is a synchronous Kleene algebra, for any CHA.*

**Proof.** The proofs of (1)-(13) are analogous to [5]. Note that in [5], instead of (12) and (13), the proofs of the equivalent axioms $\alpha \cdot \gamma \leq \gamma \Rightarrow \alpha^* \cdot \gamma \leq \gamma$ and $\gamma \cdot \alpha \leq \gamma \Rightarrow \gamma \cdot \alpha^* \leq \gamma$ are presented. We present only the proof for axioms dealing with operator $\times$, for a given word $a_1 \dots a_n \in \Sigma^*$, with $n \geq 1$.

Axiom (14):

$$
\begin{aligned}
& (\mathcal{L}_1 \times (\mathcal{L}_2 \times \mathcal{L}_3))(a_1 \dots a_n) \\
= & \quad \{ \text{ definition of } \times \} \\
& \textstyle\sum_{k\geq 1} (\mathcal{L}_1(a_1 \dots a_n); (\mathcal{L}_2 \times \mathcal{L}_3)(a_1 \dots a_k) + \mathcal{L}_1(a_1 \dots a_k); (\mathcal{L}_2 \times \mathcal{L}_3)(a_1 \dots a_n)) \\
= & \quad \{ \text{ definition of } \times \} \\
& \sum_{k\geq 1} \Big( \mathcal{L}_1(a_1 \dots a_n); \big( \sum_{l\geq 1} (\mathcal{L}_2(a_1 \dots a_n); \mathcal{L}_3(a_1 \dots a_l) + \mathcal{L}_2(a_1 \dots a_l); \mathcal{L}_3(a_1 \dots a_n)) \big) \\
& + \mathcal{L}_1(a_1 \dots a_k); \big( \sum_{l\geq 1} (\mathcal{L}_2(a_1 \dots a_n); \mathcal{L}_3(a_1 \dots a_l) + \mathcal{L}_2(a_1 \dots a_l); \mathcal{L}_3(a_1 \dots a_n)) \big) \Big) \\
= & \quad \{ (26) \text{ and } (5) \} \\
& \sum_{k\geq 1} \Big( \sum_{l\geq 1} \big( (\mathcal{L}_1(a_1 \dots a_n); (\mathcal{L}_2(a_1 \dots a_n)); \mathcal{L}_3(a_1 \dots a_l) + (\mathcal{L}_1(a_1 \dots a_n); \mathcal{L}_2(a_1 \dots a_l)); \mathcal{L}_3(a_1 \dots a_n) \big) \\
& + \sum_{l\geq 1} \big( (\mathcal{L}_1(a_1 \dots a_k); (\mathcal{L}_2(a_1 \dots a_n)); \mathcal{L}_3(a_1 \dots a_l) + (\mathcal{L}_1(a_1 \dots a_k); \mathcal{L}_2(a_1 \dots a_l)); \mathcal{L}_3(a_1 \dots a_n) \big) \Big) \\
= & \quad \{ (27) \text{ and change indexes without loss of generality} \} \\
& \sum_{k\geq 1} \Big( \sum_{l\geq 1} \big( (\mathcal{L}_1(a_1 \dots a_k); \mathcal{L}_2(a_1 \dots a_l) + \mathcal{L}_1(a_1 \dots a_l); \mathcal{L}_2(a_1 \dots a_k)); \mathcal{L}_3(a_1 \dots a_n) \big) \\
& + \sum_{l\geq 1} \big( (\mathcal{L}_1(a_1 \dots a_n); \mathcal{L}_2(a_1 \dots a_l) + (\mathcal{L}_1(a_1 \dots a_l); \mathcal{L}_2(a_1 \dots a_n)) \\
& ; \mathcal{L}_3(a_1 \dots a_k)); \mathcal{L}_3(a_1 \dots a_k)) \big) \Big) \\
= & \quad \{ \text{ definition of } \times \} \\
& \textstyle\sum_{k\geq 1} \big( (\mathcal{L}_1 \times \mathcal{L}_2)(a_1 \dots a_k); \mathcal{L}_3(a_1 \dots a_n) + (\mathcal{L}_1 \times \mathcal{L}_2)(a_1 \dots a_n); \mathcal{L}_3(a_1 \dots a_k) \big) \\
= & \quad \{ \text{ definition of } \times \} \\
& ((\mathcal{L}_1 \times \mathcal{L}_2) \times \mathcal{L}_3)(a_1 \dots a_n)
\end{aligned}
$$

Axiom (15):

$$
\begin{aligned}
& (\mathcal{L}_1 \times \mathcal{L}_2)(a_1 \dots a_n) \\
= & \quad \{ \text{ definition of } \times \} \\
& \sum_{k\geq 1} (\mathcal{L}_1(a_1 \dots a_n); \mathcal{L}_2(a_1 \dots a_k) + \mathcal{L}_1(a_1 \dots a_k); \mathcal{L}_2(a_1 \dots a_n)) \\
= & \quad \{ (2) \text{ and } (22) \} \\
& \sum_{k\geq 1} (\mathcal{L}_2(a_1 \dots a_n); \mathcal{L}_1(a_1 \dots a_k) + \mathcal{L}_2(a_1 \dots a_k); \mathcal{L}_1(a_1 \dots a_n)) \\
= & \quad \{ \text{ definition of } \times \} \\
& (\mathcal{L}_2 \times \mathcal{L}_1)(a_1 \dots a_n)
\end{aligned}
$$

Axiom (16):

$$
\begin{aligned}
& (\mathcal{L} \times \chi)(a_1 \dots a_n) \\
= & \quad \{ \text{ definition of } \times \} \\
& \sum_{k\geq 1} (\mathcal{L}(a_1 \dots a_n); \chi(a_1 \dots a_k) + \mathcal{L}(a_1 \dots a_k); \chi(a_1 \dots a_n)) \\
= & \quad \{ \text{ definition of } \chi \text{ and } (6) \} \\
& \sum_{k\geq 1} (\mathcal{L}(a_1 \dots a_n) + \mathcal{L}(a_1 \dots a_k)) = \mathcal{L}(a_1 \dots a_n) \qquad \text{definition of } \times
\end{aligned}
$$

$\chi \times \alpha$ is proved analogously.

Axiom (17):

$$
\begin{aligned}
& (\mathcal{L} \times \varnothing)(a_1 \dots a_n) \\
= \quad & \{ \text{ definition of } \times \} \\
& \sum_{k \geq 1} (\mathcal{L}(a_1 \dots a_n); \varnothing(a_1 \dots a_k) \\
& + \mathcal{L}(a_1 \dots a_k); \varnothing(a_1 \dots a_n)) \\
= \quad & \{ \text{ definition of } \varnothing \text{ and } (9) \} \\
& \sum_{k \geq 1} \mathbf{0} \\
= \quad & \{ \text{ definition of } \times \} \\
& \varnothing(a_1 \dots a_n)
\end{aligned}
$$

Axiom (18):
This axiom applies only to basic fuzzy synchronous languages $\mathcal{L}_B$. So, given a basic fuzzy synchronous language $\mathcal{L}_B$,

$$
\begin{aligned}
& (\mathcal{L}_B \times \mathcal{L}_B)(a_b) \\
= \quad & \{ \text{ definition of } \times \} \\
& \textstyle\sum_{k \geq 1} \mathcal{L}_B(a_b); \mathcal{L}_B(a_b) \\
= \quad & \{ (23) \} \\
& \textstyle\sum_{k \geq 1} \mathcal{L}_B(a_b) \\
= \quad & \{ \text{ definition of } \times \} \\
& \mathcal{L}_B(a_b)
\end{aligned}
$$

The proof of $\varnothing \times \alpha$ uses a similar reasoning.

Axiom (19):

$$
\begin{aligned}
& (\mathcal{L}_1 \times (\mathcal{L}_2 \cup \mathcal{L}_3))(a_1 \dots a_n) \\
= \quad & \{ \text{ definition of } \times \text{ and } \cup \} \\
& \sum_{k \geq 1} (\mathcal{L}_1(a_1 \dots a_n); (\mathcal{L}_2(a_1 \dots a_k) + \mathcal{L}_3(a_1 \dots a_k)) \\
& + \mathcal{L}_1(a_1 \dots a_k); (\mathcal{L}_2(a_1 \dots a_n) + \mathcal{L}_3(a_1 \dots a_n))) \\
= \quad & \{ (7) \} \\
& \sum_{k \geq 1} (\mathcal{L}_1(a_1 \dots a_n); \mathcal{L}_2(a_1 \dots a_k) + \mathcal{L}_1(a_1 \dots a_n); \mathcal{L}_3(a_1 \dots a_k) \\
& + \mathcal{L}_1(a_1 \dots a_k); \mathcal{L}_2(a_1 \dots a_n) + \mathcal{L}_1(a_1 \dots a_k); \mathcal{L}_3(a_1 \dots a_n)) \\
= \quad & \{ (2) \} \\
& \sum_{k \geq 1} (\mathcal{L}_1(a_1 \dots a_n); \mathcal{L}_2(a_1 \dots a_k) \\
& + \mathcal{L}_1(a_1 \dots a_k); \mathcal{L}_2(a_1 \dots a_n) \\
& + \mathcal{L}_1(a_1 \dots a_n); \mathcal{L}_3(a_1 \dots a_k) \\
& + \mathcal{L}_1(a_1 \dots a_k); \mathcal{L}_3(a_1 \dots a_n)) \\
= \quad & \{ \text{ definition of } \times \text{ and } \cup \} \\
& ((\mathcal{L}_1 \times \mathcal{L}_2) \cup (\mathcal{L}_1 \times \mathcal{L}_3))(a_1 \dots a_n)
\end{aligned}
$$

Axiom (20): Analogously to (19) but using (8).

Axiom (21): This proof is done by considering $\times$-fuzzy synchronous languages.

$$
\begin{aligned}
& ((\mathcal{L}_1^{\times} \cdot \mathcal{L}_1) \times (\mathcal{L}_2^{\times} \cdot \mathcal{L}_2))(a_1 \dots a_n) \\
= \quad & \{ \text{ definition of } \times \} \\
& \textstyle\sum_{k \geq 1} ((\mathcal{L}_1^{\times} \cdot \mathcal{L}_1)(a_1 \dots a_n); (\mathcal{L}_2^{\times} \cdot \mathcal{L}_2)(a_1 \dots a_k) \; + (\mathcal{L}_1^{\times} \cdot \mathcal{L}_1)(a_1 \dots a_k); (\mathcal{L}_2^{\times} \cdot \mathcal{L}_2)(a_1 \dots a_n)) \\
= \quad & \{ \text{ definition of } \cdot \} \\
& \sum_{k \geq 1} \Big( (\sum_{l \geq 0} \mathcal{L}_1^{\times}(a_1 \dots a_l); \mathcal{L}_1(a_{l+1} \dots a_n)); (\sum_{l \geq 0} \mathcal{L}_2^{\times}(a_1 \dots a_l); \mathcal{L}_2(a_{l+1} \dots a_k)) \\
& + (\sum_{l \geq 0} \mathcal{L}_1^{\times}(a_1 \dots a_l); \mathcal{L}_1(a_{l+1} \dots a_k)); (\sum_{l \geq 0} \mathcal{L}_2^{\times}(a_1 \dots a_l); \mathcal{L}_2(a_{l+1} \dots a_n)) \Big) \\
= \quad & \{ \mathcal{L}^{\times}(a_1 \dots a_k) = \mathbf{0} \text{ for } k \neq 1 \} \\
& \textstyle\sum_{k \geq 1} ((\mathcal{L}_1^{\times}(a_1); \mathcal{L}_1(a_2 \dots a_n)); (\mathcal{L}_2^{\times}(a_1); \mathcal{L}_2(a_2 \dots a_k)) + (\mathcal{L}_1^{\times}(a_1); \mathcal{L}_1(a_2 \dots a_k)); (\mathcal{L}_2^{\times}(a_1); \mathcal{L}_2(a_2 \dots a_n))) \\
= \quad & \{ (5) \text{ and } (22) \} \\
& \textstyle\sum_{k \geq 1} ((\mathcal{L}_1^{\times}(a_1); \mathcal{L}_2^{\times}(a_1)); (\mathcal{L}_1(a_2 \dots a_n); \mathcal{L}_2(a_2 \dots a_k)) + (\mathcal{L}_1^{\times}(a_1); \mathcal{L}_2^{\times}(a_1)); (\mathcal{L}_1(a_2 \dots a_k); \mathcal{L}_2(a_2 \dots a_n))) \\
= \quad & \{ (7) \} \\
& \textstyle\sum_{k \geq 1} ((\mathcal{L}_1^{\times}(a_1); \mathcal{L}_2^{\times}(a_1)); (\mathcal{L}_1(a_2 \dots a_n); \mathcal{L}_2(a_2 \dots a_k) + \mathcal{L}_1(a_2 \dots a_k); \mathcal{L}_2(a_2 \dots a_n))) \\
= \quad & \{ \text{ definition of } \times \text{ and } \mathcal{L}^{\times}(a_1 \dots a_k) = \mathbf{0} \text{ for } k \neq 1 \} \\
& \textstyle\sum_{k \geq 1} ((\mathcal{L}_1^{\times} \times \mathcal{L}_2^{\times})(a_1 \dots a_n); (\mathcal{L}_1 \times \mathcal{L}_2)(a_2 \dots a_n)) \\
= \quad & \{ \text{ definition of } \cdot \} \\
& ((\mathcal{L}_1^{\times} \times \mathcal{L}_2^{\times}) \cdot (\mathcal{L}_1 \times \mathcal{L}_2))(a_1 \dots a_n)
\end{aligned}
$$

□

Let us revisit the example mentioned in the Introduction, concerning the joint fuzzy flow of two reagents. The fuzzyness in a flow represents potential malfunctions in the control apparatus. In order to model the confidence values of execution, we assume the structure **G** of Example 2.6. Consider, for instance, that the machine releases the reagents $c$ and $m$ with certainty values 0.95 and 0.93, respectively. We model such situation by taking the action corresponding to adding $c$ with certainty 0.95 and the action of adding $m$ with certainty 0.93 by the two $\mathcal{H}$-NFA depicted in Fig. 9, where $c$ abbreviates the label $(c, \delta^c(x_0, c, x_1))$ and, analogously, $m$ the label $(m, \delta^m(x_0, m, x_1))$.


Fig. 9. Two $\mathcal{H}$-FA representing the basic actions $c$ and $m$.

Let us consider a machine able of execute both actions $c$ and $m$ simultaneously. Its behaviour is modelled by the synchronous product of the automata above, the result being depicted in Fig. 10, with $c, m$ abbreviating label $(\{c, m\}, \delta^{c\times m}((x_1, x_3), \{c, m\}, (x_2, x_4)))$.

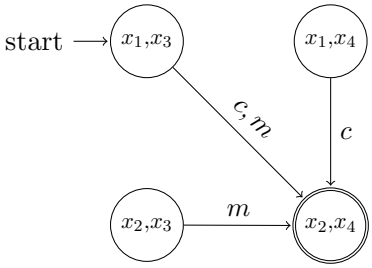


Fig. 10. The synchronous product.

The weight of action $c \times m$, corresponding to the certainty of obtaining the mix of both reagents, is given by

$$\delta^{c\times m}(((x_1, x_3), \{c, m\}, (x_2, x_4))) = \delta^c(x_1, c, x_2); \delta^m(x_3, m, x_4) = min\{0.95, 0.93\} = 0.93$$

## 5 Conclusions

In this work we defined the concept of a fuzzy synchronous language, a number of operators over such languages, and a synchronous product construction of two $\mathcal{H}$-NFA. A generalisation of two classic results was proved: for every term $\alpha$ of $T_{SKA}$, it is possible to construct a $\mathcal{H}$-NFA which accepts precisely $\widehat{FI}_{SKA}(\alpha)$; and, for all $\alpha \in T_{SKA}$, there exists a function $f$ mapping $\mathcal{M}_\alpha$ into $\alpha$ of SKA such that $\widehat{FI}_{SKA}(\alpha) = \mathcal{L}(\mathcal{M}_\alpha)$. Finally, we have shown that any set of fuzzy synchronous languages enriched with the fuzzy operators previously defined is a SKA.

Note that some axioms of Figure 2, namely (10)-(13), may have different representations in some literature. Even the very axiomatisation here presented for Kleene algebra is not minimal ((3) and (4) may be omitted). However, the axiomatisation from [18] was maintained, since we intend to present the algebra of fuzzy languages as a model of SKA. One may notice also that operator $\rightarrow$ is absent from the automata constructions presented in the paper. Its role is however related with the proof of Theorem 4.6, as it assures, together with the complete property of the Heyting algebra, the infinite distribution of ";" over arbitrary suprema.

The construction of FFA with membership degrees in a lattice-ordered monoid $\mathcal{L}$ [10] is studied in an analogous context of this work, based on the concept of $\mathcal{L}$-*fuzzy regular expression*. Such expressions are defined as regular expressions from an alphabet $X$ with a scalar $\lambda \in \mathcal{L}$ multiplication, using the monoid multiplication operator. It is precisely this scalar that attributes the weight to a transition in the automaton. In the approach presented in this paper, on the other hand, automata are built using standard regular expressions instead of fuzzy regular expressions. Regular expressions are then interpreted as fuzzy languages accepted by a fuzzy automaton, using the interpretation map $\widehat{FI}$.

Most of the results presented in the context of fuzzy languages are constructed using either the real interval $[0, 1]$ or a generic residuated lattice $\mathcal{L}$ to model the (possible) many valued membership values. However, one of the main results of this paper, Theorem 4.6 relies on properties provided by a specific characterisation of a lattice ordered structure: the operator ; of the parameter must be idempotent and commutative. The definition presented for $\mathcal{H}$-NFA differs from [12] in the semantic structure used for membership values. Although the original definition uses the unit interval $[0, 1]$, we consider values from a more generic structure, a complete Heyting algebra.

A set of possible directions for future work emerge. The extension of SKA to tests, known as the synchronous Kleene algebra with tests, SKAT [18], modelled by a notion of *fuzzy guarded synchronous languages* is worth to be discussed. This entails the need for defining guarded $\mathcal{H}$-NFA and extending the synchronous product construction accordingly. Another extension worth to be considered is to study a relaxation of SKA. Considering, for instance, the structure $\boldsymbol{R} = (R_+ \cup \{\infty\}, min, +, \infty, 0, \rightarrow)$ with $x \rightarrow y = max\{y - x, 0\}$, $\forall x, y \in R_+ \cup \{\infty\}$, known as the tropical semiring, as a parameter, would make possible to address situations where the experimenter could choose the desired proportion of reagents $c$ and $m$ involved. The synchronous action $c \times m$ would then represent the sum of the respective quantities. Such extension would not only broaden the number of applications of the approach proposed in this paper, but also open the discussion on which implications the more generic algebra would have in the proven results.

# References

[1] L A. Zadeh. Fuzzy languages and their relation to human and machine intelligence. *Proc. Int. Conf. on Man and Computer*, pages 148–179, 05 1996.

[2] Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. On the average size of

glushkov and equation automata for KAT expressions. In Leszek Gasieniec and Frank Wolter, editors, *Fundamentals of Computation Theory - 19th International Symposium, FCT 2013, Liverpool, UK, August 19-21, 2013. Proceedings*, volume 8070 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2013.

[3] Mansoor Doostfatemeh and Stefan C. Kremer. New directions in fuzzy automata. *Int. J. Approx. Reasoning*, 38(2):175–214, 2005.

[4] Daniel Figueiredo, Manuel A. Martins, and Madalena Chaves. Applying differential dynamic logic to reconfigurable biological networks. *Mathematical Biosciences*, 291:10–20, 2017.

[5] Leandro Gomes, Alexandre Madeira, and Luís Soares Barbosa. Generalising KAT to verify weighted computations. Technical report, HASLab INESC TEC - Univ. of Minho, Portugal, Department of Informatics, 2018.

[6] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley, 2003.

[7] S. C. Kleene. Representation of events in nerve nets and finite automata. In Claude Shannon and John McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, NJ, 1956.

[8] Dexter Kozen. On Kleene algebras and closed semirings. In Branislav Rovan, editor, *Mathematical Foundations of Computer Science 1990, MFCS'90, Banská Bystrica, Czechoslovakia, August 27-31, 1990, Proceedings*, volume 452 of *Lecture Notes in Computer Science*, pages 26–47. Springer, 1990.

[9] E. T. Lee and Lotfi A. Zadeh. Note on fuzzy languages. *Inf. Sci.*, 1(4):421–434, 1969.

[10] Yongming Li and Witold Pedrycz. Fuzzy finite automata and fuzzy regular expressions with membership values in lattice-ordered monoids. *Fuzzy Sets and Systems*, 156(1):68–92, 2005.

[11] Feng Lin and Hao Ying. Modeling and control of fuzzy discrete event systems. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 32(4):408–415, 2002.

[12] Alexandru Mateescu, Arto Salomaa, Kai Salomaa, and Sheng Yu. Lexical analysis with a simple finite-fuzzy-automaton model. *J. UCS*, 1(5):292–311, 1995.

[13] R. Milner. *A Calculus of Communicating Systems*. Springer Lect. Notes Comp. Sci. (92), 1980.

[14] Robin Milner. Calculi for synchrony and asynchrony. *Theor. Comput. Sci.*, 25:267–310, 1983.

[15] Robin Milner. *Turing, Computing and Communication*, pages 1–8. Springer Berlin Heidelberg, Berlin, Heidelberg, 01 2006.

[16] John Mordeson and Davender Malik. *Fuzzy automata and languages: theory and applications*. Chapman and Hall/CRC; 1 edition, 2002.

[17] Witold Pedrycz and Adam Gacek. Learning of fuzzy automata. *International Journal of Computational Intelligence and Applications*, 1(1):19–33, 2001.

[18] Cristian Prisacariu. Synchronous Kleene algebra. *J. Log. Algebr. Program.*, 79(7):608–635, 2010.

[19] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230 – 245, 1963.

[20] Krister Segerberg. A deontic logic of action. *Studia Logica*, 41(2):269–282, Jun 1982.

[21] Aleksandar Stamenkovic and Miroslav Ciric. Construction of fuzzy automata from fuzzy regular expressions. *Fuzzy Sets and Systems*, 199:1–27, 2012.

[22] G. H. von Wright. *An Essay in Deontic Logic and the General Theory of Action with a Bibliography of Deontic and Imperative Logic.* –. North-Holland Pub. Co, 1968.

[23] William Wee and K S. Fu. A formulation of fuzzy automata and its application as a model of learning systems. *IEEE Trans. Systems Science and Cybernetics*, 5:215–223, 01 1969.

[24] Mingsheng Ying. A formal model of computing with words. *IEEE Trans. Fuzzy Systems*, 10(5):640–652, 2002.