8

Fuzzy Computation

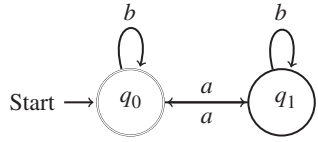
Computation is about calculating or enumerating mechanically. Typically, the word mechanically means that one builds a device and sets it in motion in order to perform the desired calculation or enumeration. Many and different real or conceptual devices capable of performing computations have been proposed. Most of them operate in a crisp environment and in a crisp manner. However, there are some devices that profit from the use of vagueness in their overall operation. These devices and the related theory are described in this chapter. The material presented in this chapter is based on [220, 274].

8.1 Automata, Grammars, and Machines

A *finite automaton* can be seen as a machine equipped with scanning head that can read the contents of sequence of cells, while the head can move in only one direction. At any moment, the machine is in a state. Initially, the scanning head is positioned on the leftmost cell, while a number of symbols are printed on consecutive cells starting with the leftmost cell. Also, the machine enters a default initial state. Each machine is associated with a number of *transition rules*. A transition rule has the general form $q_i \xrightarrow{s} q_j$, where q_i and q_j are states and s is a symbol. The meaning of this rule is that if the automaton is in state q_i, it will enter state q_j only if the next symbol is s. When the machine starts, it reads the first symbol and if there is a transition rule that includes this symbol and the current state, then the scanning head moves to the right and the machine enters a new state. If the machine enters the final state, then it accepts the input and terminates. The machine may suspend without completion when no transition rule applies. The alphabet of the machine consists of the symbols that the scanning head can

A Modern Introduction to Fuzzy Mathematics, First Edition.
Apostolos Syropoulos and Theophanes Grammenos.

recognize. The following figure shows a finite automaton with two states whose alphabet is $\{a, b\}$.



The symbols in the circles are the states, the symbol in the double circle is the accepting state, and the symbols over the arcs are the symbols that the automaton consumes. Thus, this is a compact way to write the various transition rules. This automaton determines if an input sequence of symbols over its alphabet contains an even number of a's. For example, if the input is the sequence "$abba$", then the following table shows what has to be done in order to have this sequence accepted by this automaton.

Current state	Unread symbols	Transition rule
q_0	$abba$	$q_0 \xrightarrow{a} q_1$
q_1	bba	$q_1 \xrightarrow{b} q_1$
q_1	ba	$q_1 \xrightarrow{b} q_1$
q_1	a	$q_1 \xrightarrow{a} q_0$
q_0		Input accepted!

More generally, automata are able to examine whether character sequences or just *strings* belong to some formal language.

Definition 8.1.1 Assume that Σ is an arbitrary set of symbols, which is called *alphabet*, and that ϵ denotes the *empty string*. Then,

$$\Sigma^* = \{\epsilon\} \cup \Sigma \cup \Sigma \times \Sigma \cup \Sigma \times \Sigma \times \Sigma \cup \cdots .$$

is the set of all finite strings over Σ. A *formal language* L, or just language L, is a subset of Σ^*, that is, $L \subseteq \Sigma^*$.

Typically, a language is defined by a grammar:

Definition 8.1.2 A grammar is defined to be a quadruple $G = (V_N, V_T, S, \Phi)$ where V_T and V_N are disjoint sets of terminal and nonterminal (syntactic class) symbols, respectively; S, a distinguished element of V_N, is called the starting symbol. Φ is a finite nonempty relation from $(V_T \cup V_N)^* V_N (V_T \cup V_N)^*$ to $(V_T \cup V_N)^*$. In general, an element (α, β) is written as $\alpha \to \beta$ and is called a production or rewriting rule [287].

Grammars are classified as follows:

Unrestricted grammars. There are no restrictions on the form of the production rules.

Context-sensitive grammars. The relation Φ contains only productions of the form $\alpha \to \beta$, where $|\alpha| \leq |\beta|$, and in general, $|\gamma|$ is the length of the string γ.

Context-free grammars. The relation Φ contains only productions of the form $\alpha \to \beta$, where $|\alpha| = 1$ and $\alpha \in V_{\mathrm{N}}$.

Regular grammars. The relation Φ contains only productions of the form $\alpha \to \beta$, where $|\alpha| \leq |\beta|$, $\alpha \in V_{\mathrm{N}}$, and β has the form aB or a, where $a \in V_{\mathrm{T}}$ and $B \in V_{\mathrm{N}}$.

Syntactically complex languages can be defined by means of grammars. To each class of languages, there is a class of automata (machines) that *accept* (i.e., they can answer the decision problem "$s \in L$?," where s is a string and L is a language) this class of languages, which are generated by the respective grammars. In particular, *finite automata* accept languages generated by regular grammars, *push-down automata* accept languages generated by context-free grammars, *linear bounded automata* accept languages generated by context-sensitive grammars, and *Turing machines* accept *recursive* languages, that is, a subclass of the class of languages generated by unrestricted grammars.

A Turing machine is a conceptual computing device consisting of an *infinite tape*, a *controlling device*, and a *scanning head* (see Figure 8.1). The tape is divided into an infinite number of cells. The scanning head can read and write symbols in each cell. The symbols are elements of a set $\Sigma = \{S_1, \ldots, S_n\}$, $n \geq 1$, which is called the *alphabet*. Usually, there is an additional symbol, $\llcorner\!\lrcorner$, called the *blank* symbol,

The Turing machine's scanning head moves back and forth along the tape. The number that the scanning head displays is its current state, which changes as it proceeds.

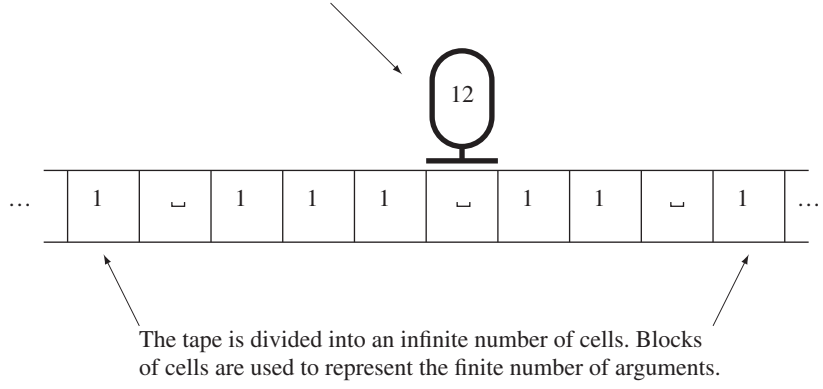


Figure 8.1 A typical Turing machine.

and when this symbol is written on a cell by the scanning head, the effect of this operation is the erasure of the symbol that was printed on this particular cell. At any moment, the machine is in a *state* q_i, which is a member of a finite set $Q = \{q_0, q_1, \dots, q_r\}, r \geq 0$. The controlling device is actually a look-up table that is used to determine what the machine has to do next at any given moment. In particular, the action a machine has to take depends on its current state and the symbol that is printed on the cell the scanning head has just finished scanning. If no action has been specified for a particular combination of state and symbol, the machine halts. Usually, the control device is specified by a finite set of *quadruples*, which are special cases of expressions.

Definition 8.1.3 An *expression* is a string of symbols chosen from the list q_0, q_1,…; ⎵, S_1, …; R, L.

A quadruple can have one of the following forms:

$$q_i S_j S_k q_l \tag{8.1}$$

$$q_i S_j L q_l \tag{8.2}$$

$$q_i S_j R q_l \tag{8.3}$$

Note that $L, R \notin \Sigma$ and $S_j, S_k \in \Sigma \cup \{\_\}$. The quadruple (8.1) specifies that if the machine is in state q_i and the cell that the scanning head scans contains the symbol S_j, then the scanning head replaces S_j by S_k and the machine enters state q_l. The quadruples (8.2) and (8.3) specify that if the machine is in state q_i and the cell that the scanning head scans contains the symbol S_j, then the scanning head moves to the cell to the left of the current cell, or to the cell to the right of the current cell, respectively, and the machine enters the state q_l. Sometimes the following quadruple is also considered:

$$q_i S_j q_k q_l. \tag{8.4}$$

This quadruple is particularly useful if we want to construct a Turing machine that will compute *relatively computable functions* . These quadruples provide a Turing machine with a means of communicating with an external agency that can give correct answers to questions about a set $A \subset \mathbb{N}$. In particular, when a machine is in state q_i and the cell that the scanning head scans contains the symbol S_j, then the machine can be thought of as asking the question, "Is $n \in A$?" Here n is the number of S_1's that are printed on the tape. If the answer is "yes," then the machine enters state q_k; otherwise it enters state q_l. Turing machines equipped with such an external agency are called *oracle machines* , and the external agency is called an *oracle*.

Turing machines are used to compute the value of functions $f(n_1, \dots, n_m)$ that take values in $\mathbb{N}^m$. Each argument $n_i \in \mathbb{N}$, is represented on the tape by preprinting

the symbol S_1 on $n_i + 1$ consecutive cells. Typically, such a block of cells is denoted by $\overline{n_1}$. Argument representations are separated by a blank cell (i.e., a cell on which the symbol $\_$ is printed), while all other cells are empty (i.e., the symbol S_0 has been preprinted on each cell). It is customary to represent such a block of cell with the expression

$$\overline{(n_1, n_2, \dots, n_m)} = \overline{n_1}\_\overline{n_2}\_\cdots\_\overline{n_m}.$$

If α is an expression, then $\langle\alpha\rangle$ will denote the number of S_1 contained in α. In addition,

$$\overline{\langle m-1\rangle} = m \quad \text{and} \quad \langle\alpha\beta\rangle = \langle\alpha\rangle + \langle\beta\rangle.$$

It is also customary to use the symbol 1 for S_1. Thus, the sequence $3, 4, 2$ will be represented by the following three blocks of 1's:

1111_11111_111

The machine starts at state q_0 and the scanning head is placed atop the leftmost 1 of a sequence of n blocks of 1's. If the machine has reached a situation in which none or more than one quadruple is applicable, the machine halts. Once the machine has terminated, the result of the computation is equal to the number of cells on which the symbol S_1 is printed.

Although the description presented so far is quite formal for our own taste, still fuzzy versions of the Turing machine are extensions of the "standard" formal definition that is given below.

Definition 8.1.4 A Turing machine $\mathscr{M}$ is an octuple $(Q, \Sigma, \Gamma, \delta, \_, \rhd, q_0, H)$, where

- Q is a finite set of states;
- Σ is the input alphabet;
- Γ is an alphabet, called *working alphabet*, where $\Sigma \subseteq \Gamma$;
- $\_ \in \Gamma$ is the blank symbol;
- $\rhd \in \Gamma$ is the *left end symbol*;
- $q_0 \in Q$ is the initial state;
- $H \subseteq Q$ is the set of halting or accepting and rejecting states;
- δ, the *transition* function, is a function from $(Q \setminus H) \times \Gamma$ to $Q \times \Gamma \times \{L, R, N\}$ such that $\mathscr{M}$ may perform an instruction $(q', Y, D) \in Q \times \Gamma \times \{L, R, N\}$, if $\mathscr{M}$ is in state q, the scanning head has just read the symbol X, and $\delta(q, X) = (q', Y, D)$. Depending on the value of D, the machine will move to the left (L), to the right (R), or it will stay still when $D = N$ and the scanning head will overwrite X with Y. Also, $\delta(q, \rhd) = (q, \rhd, R)$, which means that whenever the scanning head has read the symbol $\rhd$, it immediately moves to the right.

Note that here we have defined a machine that has a unidirected tape and not a bidirectional tape. One can get the bidirectional version by eliminating all references to the $\triangleright$ symbol.

A *configuration* of $\mathscr{M}$ is an element from

$$C(\mathscr{M}) = \{\triangleright\}\Gamma^* Q\Gamma^+ \cup Q\{\triangleright\}\Gamma^*,$$

where AB denotes strings that are formed by concatenating a string that belongs to A with a string that belongs to B. If $w_1 q a w_2$ is a configuration, then $w_1 \in \{\triangleright\}\Gamma^*$, $w_2 \in \Gamma^*$, $a \in \Gamma$, and $q \in Q$. Moreover, this configuration means that a machine $\mathscr{M}$ is in state q, the content of the tape is $\triangleright\, w_1 a w_2 \_\_\_ \ldots$, and the scanning head sits atop the $(|w_1| + 1)$th cell, where $|s|$ is the length of string s. The *initial* configuration is $q_0 \triangleright x$, where x is the input fed to the machine. A configuration whose state component is in H is called a *halted* configuration.

A *step* is a relation $\vdash_{\mathscr{M}}$ on the set of configurations defined as follows:

(i) $\cdots x_{i-1} q x_i x_{i+1} \cdots \vdash_{\mathscr{M}} \cdots x_{i-1} q' y x_{i+1} \cdots$, if $\delta(q, x_i) = (q', y, N)$;

(ii) $\cdots x_{i-1} q x_i x_{i+1} \cdots \vdash_{\mathscr{M}} \cdots x_{i-2} q' x_{i-1} y x_{i+1} \cdots$, if $\delta(q, x_i) = (q', y, L)$;

(iii) $\cdots x_{i-1} q x_i x_{i+1} \cdots \vdash_{\mathscr{M}} \cdots x_{i-1} y q' x_{i+1} \cdots$, if $\delta(q, x_i) = (q', y, R)$; and

(iv) $\cdots x_{n-1} q x_n \vdash_{\mathscr{M}} \cdots x_{n-1} y q' \_$, if $\delta(q, x_n) = (q', y, R)$.

A *computation* by $\mathscr{M}$ is a sequence of configurations $C_0, C_1, \ldots, C_n$, for some $n \geq 0$ such that

$$C_0 \vdash_{\mathscr{M}} C_1 \vdash_{\mathscr{M}} C_2 \vdash_{\mathscr{M}} \cdots \vdash_{\mathscr{M}} C_n.$$

A computation from C_0 to C_n can be written compactly as $C_0 \vdash^*_{\mathscr{M}} C_n$.

A computation by $\mathscr{M}$ on input x is a series of actions that start at configuration $C_0 = q_0 \triangleright x$ and is either *infinite* (i.e. nonterminating) or stops at configuration $w_1 q w_2$, where $q \in H$. Assume that $H = \{q_a, q_r\}$, where q_a is the *accepting* state and q_r is the *rejecting* state. Then, a computation is called *accepting* if it finishes in the configuration $w_1 q_a w_2$ and *rejecting* if it finishes in the configuration $w_1 q_r w_2$. Furthermore, if a computation on input x is accepting or rejecting, we say that the corresponding machine *accepts* or *rejects* x, respectively. More generally, the words accepted by $\mathscr{M}$ form a formal language $L(\mathscr{M})$.

8.2 Fuzzy Languages and Grammars

Suppose that Σ is an alphabet. Then, a *fuzzy formal language* (or just fuzzy language) is a fuzzy subset of Σ^*. If we have a set of nonterminal symbols, V_{N}, and

a set of terminal symbols, V_T, such that $V_T \cap V_N = \emptyset$, then a fuzzy language is a fuzzy subset of V_T^*.

Assume that λ_1 and λ_2 are two fuzzy languages over V_T. Then, the *union* of λ_1 and λ_2 is the fuzzy language denoted by $\lambda_1 \cup \lambda_2$ and defined by

$$(\lambda_1 \cup \lambda_2)(x) = \lambda_1(x) \vee \lambda_2(x), \quad \text{for all } x \in V_T^*.$$

Similarly, the *intersection* of λ_1 and λ_2 is the fuzzy language denoted by $\lambda_1 \cap \lambda_2$ and defined by

$$(\lambda_1 \cap \lambda_2)(x) = \lambda_1(x) \wedge \lambda_2(x), \quad \text{for all } x \in V_T^*.$$

The *concatenation* of λ_1 and λ_2 is the fuzzy language denoted by $\lambda_1\lambda_2$ and defined by

$$(\lambda_1\lambda_2)(x) = \bigvee \{\lambda_1(u) \wedge \lambda_2(v) | x = uv,\ u, v \in V_T^*\}.$$

Since the operators $\wedge$ and $\vee$ are distributive, concatenation is associative. Note that $\lambda^2(x) = (\lambda\lambda)(x)$, $\lambda^3(x) = (\lambda\lambda\lambda)(x)$, $\lambda^4(x) = (\lambda\lambda\lambda\lambda)(x)$, etc.

Suppose that λ is a fuzzy language in V_T. Then, the fuzzy subset λ^∞ of V_T^* defined by

$$\lambda^\infty(x) = \bigvee \{\lambda^n(x) | n = 0, 1, 2, \dots\} \quad \text{for all } x \in V_T^*$$

is called the *Kleene closure* of λ.

A fuzzy grammar includes a set of rules for generating the elements of a fuzzy language. More specifically, a fuzzy grammar G is a quadruple (V_N, V_T, S, Φ) where Φ is a set of fuzzy production rules and everything else is as in Definition 8.2.2. The elements of Φ are expressions of the form

$$\alpha \xrightarrow{\rho} \beta, \quad \rho > 0, \tag{8.5}$$

where $\alpha, \beta \in (V_T \cup V_N)^*$ and ρ is the plausibility degree that α can generate β. Also, we write $\rho_{\alpha,\beta}$ to designate that this is the plausibility degree of rule of the form $\alpha \to \beta$. Given the *rewriting rule* (8.5) and two arbitrary strings s and t in $(V_T \cup V_N)^*$, then we have

$$s\alpha t \xrightarrow{\rho} s\beta t$$

and $s\beta t$ is said to be *directly derivable* from $s\alpha t$. Suppose that $r_1, r_2, \dots, r_n \in (V_T \cup V_N)^*$ and

$$r_1 \xrightarrow{\rho_2} r_2, \dots, r_{n-1} \xrightarrow{\rho_n} r_n,$$

where $\rho_2, \dots, \rho_n > 0$, then r_n is *derivable* from r_1 in G. This is usually written as

$$r_1 \underset{G}{\Rightarrow} r_m \quad \text{or simply} \quad r_1 \Rightarrow r_n.$$

The following expression

$$r_1 \xrightarrow{\rho_2} r_2 \to \cdots \to r_{n-1} \xrightarrow{\rho_n} r_n$$

is called a *derivation chain* from r_1 to r_n.

A fuzzy grammar G generates a fuzzy language $L(G)$. Given a string x consisting of terminal symbols, we say that it is in $L(G)$ if and only if x is derivable from S. The membership degree of x in $L(G)$ is given by

$$L(G)(x) = \bigvee(\rho_{S,r_1} \wedge \rho_{r_1,r_2} \wedge \cdots \wedge \rho_{r_n,x}), \tag{8.6}$$

where the least upper bound is taken over all derivation chains from S to x. This means that (8.6) defines $L(G)$ as a fuzzy subset of $(V_T \cup V_N)^*$. Also, if $L(G_1)$ and $L(G_2)$ are equal fuzzy sets, then the grammars G_1 and G_2 are *equivalent*. Naturally, it is important to know whether we can compute $L(G)(x)$ using Eq. (8.6):

Definition 8.2.1 A fuzzy grammar G is *recursive* if there is an *algorithm* that can be used to compute $L(G)(x)$ using Eq. (8.6).

Similar to crisp grammars, fuzzy grammars are classified as follows:

Fuzzy unrestricted grammars. Production rules are of the form $\alpha \xrightarrow{\rho} \beta$, $\rho > 0$, where $\alpha, \beta \in (V_T \cup V_N)^*$.

Fuzzy context-sensitive grammars. Production rules are of the form $\alpha_1 A \alpha_2 \xrightarrow{\rho} \alpha_1 \beta \alpha_2$, $\rho > 0$, where $\alpha_1, \alpha_2, \beta \in (V_T \cup V_N)^*$, $A \in V_N$, and $\beta \neq \epsilon$. The production rule $S \to \epsilon$ is also allowed.

Fuzzy context-free grammars. Production rules are of the form $A \xrightarrow{\rho} \beta, A \in V_N$, $\beta \in (V_T \cup V_N)^*$, $\beta \neq \epsilon$, and $S \to \epsilon$.

Fuzzy regular grammars. Productions are of the form $A \xrightarrow{\rho} \beta B$ or $A \xrightarrow{\rho} \beta$, $\rho > 0$, where $\alpha \in V_T$ and $A, B \in V_N$. In addition, the rule $S \to \epsilon$ is allowed.

The following result states that fuzzy context-sensitive and hence also fuzzy context-free and fuzzy regular grammars are recursive.

Theorem 8.2.1 *If $G = (V_N, V_T, S, \Phi)$ is a fuzzy context-sensitive grammar, then G is recursive.*

Proof: The first thing we need to show is that, for any kind of grammar, the least upper bound in (8.6) may be taken over a subset of the set of all derivation chains from S to x, namely, the subset of chains in which no r_i, $i = 1, \ldots, n$, occurs more than once.

Assume that in a derivation chain C,

$$C = S \xrightarrow{\rho_1} r_1 \xrightarrow{\rho_1} r_2 \cdots \xrightarrow{\rho_n} r_n \xrightarrow{\rho_{n+1}} x,$$

r_i is the same as $r_j, j > i$. Suppose that C' is the chain resulting from replacing the subchain

$$r_i \xrightarrow{\rho_{i+1}} \cdots \xrightarrow{\rho_j} r_j \xrightarrow{\rho_{j+1}} r_{j+1}$$

in C by $r_i \xrightarrow{\rho_{j+1}} r_{j+1}$. Obviously, if C is a derivation chain from S to x, then C' is also a derivation chain. But

$$\bigwedge\{\rho_1, \dots, \rho_i, \rho_{i+1}, \dots, \rho_{j+1}, \dots, \rho_{n+1}\} \leq \bigwedge\{\rho_1, \dots, \rho_i, \rho_{j+1}, \dots, \rho_{n+1}\}$$

and so C may be deleted without affecting the least upper bound in (8.6). This means that we can replace the definition (8.6) for $L(G)(x)$ by

$$L(G)(x) = \bigvee \bigwedge\{\rho_{S,r_1}, \rho_{r_1,r_2}, \dots, \rho_{r_n,x}\}, \tag{8.7}$$

where the least upper bound is taken over all loop-free derivation chains from S to x.

We proceed by showing that for fuzzy context-sensitive grammars, the set over which the least upper bound is taken in (8.7) can be derivation chains of bounded length l_0, where l_0 depends on $|x|$ (i.e. the length of x) and the number of symbols in $V_T \cup V_N$.

If G is fuzzy context-sensitive, then because the production rules in Φ are not contracting, it follows that

$$|r + j| \geq |r_i| \quad \text{if } j > i. \tag{8.8}$$

Suppose that $\text{card}(V_T \cup V_N) = k$. Then, since there are at most k' distinct strings in $(V_T \cup V_N)^*$ having length equal to l, and since the derivation chain is loop-free, it follows from (8.8) that the total length of the chain is bounded by

$$l_0 = 1 + k + \cdots + k^{|x|}.$$

Now let us describe a method that can be used to generate all finite derivation chains from S to x having length less than or equal to l_0. We start with S and, using Φ, generate the set Q_1 of all strings in $(V_T \cup V_N)^*$ having length less than or equal to $|x|$ that are derivable from S in one step. Then, we construct Q_2, that is, the set of all strings in $(V_T \cup V_N)^*$ having length less than or equal to $|x|$ that are derivable from S in two steps. This can be done by noting that Q_2 is identical with the set of all strings in $(V_T \cup V_N)^*$ having length less than or equal to $|x|$ that are directly derivable from strings in Q_1. Next, we construct the sets $Q_3, Q_4, \dots, Q_k$ by following the same process. This process should stop the moment $k = l_0$ or when $Q_k = \emptyset$. Since each Q_i, $i = 1, \dots, k$ is a finite set, we have managed to find, in a finite number of steps, all loop-free derivation chains from S to x having length less than or equal to l_0 and so to compute $L(G)(x)$ from (8.6). Clearly, this makes the whole process an *algorithm*, which may not be an optimal one, that can be used to compute $L(G)(x)$. Here G is recursive. □

8.3 Fuzzy Automata

In general, a fuzzy automaton, or more formally, a *fuzzy finite state automaton* is, a triple $M = (Q, \Sigma, A)$, where Q is the set of states, X is the set of input symbols and A is a fuzzy subset of $Q \times \Sigma \times Q$, that is, $A : Q \times \Sigma \times Q \to [0, 1]$. Both Q and Σ are finite nonempty sets and Σ^* is the set of all finite words of elements of Σ.

Definition 8.3.1 Suppose that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton. Then, $A^* : Q \times \Sigma^* \times Q \to [0, 1]$ is defined as follows:

$$A^*(q, \epsilon, r) = \begin{cases} 1, & \text{if } q = r \\ 0, & \text{if } q \neq r \end{cases}$$

and

$$A^*(q, \sigma a, r) = \bigvee \{A^*(q, \sigma, s) \wedge A(s, a, r) | s \in Q\}, \quad \forall \sigma \in \Sigma^*, a \in \Sigma.$$

Definition 8.3.2 Assume that $M_1 = (Q_1, \Sigma_1, A_1)$ and $M_2 = (Q_2, \Sigma_2, A_2)$ are two fuzzy finite state automata. Then, a pair (α, β) of mappings, $\alpha : Q_1 \to Q_2$ and $\beta : \Sigma_1 \to \Sigma_2$ is called a *homomorphism* written $(\alpha, \beta) : M_1 \to M_2$ if $A_1(q, x, p) \leq A_2(\alpha(q), \beta(x), \alpha(p))$ for all $p, q \in Q_1$ and all $x \in \Sigma_1$.

The pair (α, β) is called a *strong homomorphism* if

$$A_2(\alpha(q), \beta(x), \alpha(p)) = \bigvee \{A_1(q, x, t) | t \in Q_1 \text{ and } \alpha(t) = \alpha(p)\}$$

for all $q, p \in Q_1$ and all $x \in \Sigma_1$.

Suppose that $M_1 = (Q_1, \Sigma_1, A_1)$ and $M_2 = (Q_2, \Sigma_2, A_2)$ are two fuzzy finite state automata. Also, suppose that $\overline{\Sigma}$ is a finite set and $f : \overline{\Sigma} \to \Sigma_1 \times \Sigma_2$ is a function. Assume that π_1 and π_2 are the projection maps of $\Sigma_1 \times \Sigma_2$ onto Σ_i, $i = 1, 2$. Then, we define $A_f : (Q_1 \times Q_2) \times \overline{\Sigma} \times (Q_1 \times Q_2) \to [0, 1]$ as follows:

$$A_f((q_1, q_2), \sigma, (p_1, p_2)) = A_1 \times A_2((q_1, q_2), (\pi_1(f(\sigma)), \pi_2(f(\sigma))),$$

for all $(q_1, q_2), (p_1, p_2) \in Q_1 \times Q_2$ and for all $\sigma \in \overline{\Sigma}$. Then, $(Q_1 \times Q_2, \overline{\Sigma}, A_f)$ is called the *general direct product* of M_1 and M_2 and it is usually denoted by $M_1 * M_2$.

Example 8.3.1 Suppose that $M_1 = (Q_1, \Sigma_1, A_1)$ and $M_2 = (Q_2, \Sigma_2, A_2)$ are two fuzzy finite state automata, where $Q_1 = \{q_1, q_2\}$, $\Sigma_1 = \{a\}$, $Q_2 = \{r_1, r_2\}$, $\Sigma_2 = \{a, b\}$, and A_1 and A_2 are defined as follows:

$$A_1(q_1, a, q_1) = 0 \qquad A_1(q_2, a, q_2) = 0 \qquad A_1(q_1, a, q_2) = 0.3,$$
$$A_1(q_2, a, q_1) = 0.4 \qquad A_2(r_1, b, r_1) = 0.5 \qquad A_2(r_1, b, r_2) = 0,$$
$$A_2(r_2, a, r_1) = 0 \qquad A_2(r_2, a, r_2) = 0.6 \qquad A_2(r_1, a, r_1) = 0,$$
$$A_2(r_2, b, r_1) = 0.9 \qquad A_2(r_1, a, r_2) = 0.2 \qquad A_2(r_2, b, r_2) = 0,$$

Then,

$$A_1 \times A_2((q_1, r_1), (a, b), (q_2, r_2)) = 0.3,$$
$$A_1 \times A_2((q_2, r_1), (a, b), (q_1, r_1)) = 0.4,$$
$$A_1 \times A_2((q_1, r_1), (a, a), (q_2, r_2)) = 0.2,$$
$$A_1 \times A_2((q_2, r_2), (a, b), (q_1, r_1)) = 0.4,$$
$$A_1 \times A_2((q_2, r_1), (a, a), (q_1, r_2)) = 0.2,$$
$$A_1 \times A_2((q_1, r_2), (a, b), (q_2, r_1)) = 0.3,$$
$$A_1 \times A_2((q_1, r_2), (a, a), (q_2, r_2)) = 0.3,$$
$$A_1 \times A_2((q_2, r_2), (a, a), (q_1, r_2)) = 0.4.$$

Assume that $M_1 = (Q_1, \Sigma_1, A_1)$ and $M_2 = (Q_2, \Sigma_2, A_2)$ are two fuzzy finite state automata. Also, assume that $\omega : Q_2 \times \Sigma_2 \to \Sigma_1$ is a function and $Q = Q_1 \times Q_2$. Define

$$A^\omega : Q \times \Sigma_2 \times Q \to [0, 1]$$

as follows: for all $((q_1, q_2), b, (r_1, r_2)) \in Q \times \Sigma_2\ Q$,

$$A^\omega((q_1, q_2), b, (r_1, r_2)) = A_1(q_1, \omega(q_2, b), r_1) \wedge A_2(q_2, b, r_2).$$

Then, $M = (Q, \Sigma_2, A^\omega)$ is a fuzzy finite state automaton, it is called the *cascade product* of M_1 and M_2 and we write $M = M_1 \omega M_2$.

Example 8.3.2 Suppose that $M_1 = (Q_1, \Sigma_1, A_1)$ and $M_2 = (Q_2, \Sigma_2, A_2)$ are two fuzzy finite state automata, where $Q_1 = \{q_1, q_2\}$, $\Sigma_1 = \{a, b\}$, $Q_2 = \{r_1, r_2\}$, $\Sigma_2 = \{a\}$, and A_1 and A_2 are defined as follows:

$$A_1(q_1, a, q_1) = 0 \qquad A_1(q_2, b, q_1) = 0.4 \qquad A_1(q_1, a, q_2) = 0.1,$$
$$A_1(q_2, b, q_2) = 0 \qquad A_1(q_1, b, q_1) = 0.2 \qquad A_2(r_1, a, r_1) = 0,$$
$$A_1(q_1, b, q_2) = 0 \qquad A_2(r_1, a, r_2) = 0.6 \qquad A_2(r_2, a, r_1) = 0.7,$$
$$A_1(q_2, a, q_2) = 0.3 \qquad A_2(r_2, a, r_2) = 0.$$

Suppose that $\omega : Q_2 \times \Sigma_2 \to \Sigma_1$ is defined as follows:

$$\omega(r_1, a) = a \quad \text{and} \quad \omega(r_2, a) = b.$$

Also, suppose that $Q = Q_1 \times Q_2$. Then, $A^\omega : Q \times \Sigma_2 \times Q \to [0, 1]$ is such that

$$A^\omega((q_1, r_1), a, (q_2, r_2)) = A_1(q_1, \omega(r_1, a), q_2) \wedge A_2(r_1, a, r_2) = 0.1$$
$$A^\omega((q_2, r_2), a, (q_1, r_1)) = A_1(q_2, \omega(r_2, a), q_1) \wedge A_2(r_2, a, r_1) = 0.4$$
$$A^\omega((q_1, r_2), a, (q_1, r_1)) = A_1(q_1, \omega(r_2, a), q_1) \wedge A_2(r_2, a, r_1) = 0.2$$
$$A^\omega((q_2, r_1), a, (q_2, r_2)) = A_1(q_2, \omega(r_1, a), q_2) \wedge A_2(r_1, a, r_2) = 0.3$$

and A^ω is 0 elsewhere.

As before, let $M_1 = (Q_1, \Sigma_1, A_1)$ and $M_2 = (Q_2, \Sigma_2, A_2)$ be two fuzzy finite state automata. Also, let $f : Q_2 \to \Sigma_1$ be a function. In addition, let

$$A^0 : Q \times (\Sigma_1^{Q_2} \times \Sigma_2) \to [0, 1]$$

be a function such that for all $((q_1, q_2), (f, b), (r_1, r_2)) \in Q \times (\Sigma_1^{Q_2} \times \Sigma_2) \times Q$,

$$A^0((q_1, q_2), (f, b), (r_1, r_2)) = A_1(q_1, f(q_2), r_1) \wedge A_2(q_2, b, r_2).$$

Then, $M = (Q, \Sigma_1^{Q_2} \times \Sigma_2, A^0)$ is a fuzzy finite state automaton. $M = M_1 \circ M_2$ is called the *wreath product* of M_1 and M_2.

Example 8.3.3 Suppose that M_1 and M_2 are the two fuzzy finite state automata of the previous example. Suppose that $\Sigma_1^{Q_2} = \{f_1, f_2, f_3, f_4\}$, where $f_1(r_1) = f_1(r_2) = a, f_2(r_1) = a, f_2(r_2) = b, f_3(r_1) = b, f_3(r_2) = a$, and $f_4(r_1) = f_4(r_2) = b$. Then,

$$A^0((q_1, r_1), (f_1, a), (q_2, r_2)) = A_1(q_1, f_1(r_1), q_2) \wedge A_2(r_1, a, r_2) = 0.1,$$
$$A^0((q_2, r_1), (f_1, a), (q_2, r_2)) = A_1(q_2, f_1(r_1), q_2) \wedge A_2(r_1, a, r_2) = 0.3,$$
$$A^0((q_1, r_2), (f_1, a), (q_2, r_1)) = A_1(q_1, f_1(r_2), q_2) \wedge A_2(r_2, a, r_1) = 0.1,$$
$$A^0((q_2, r_2), (f_1, a), (q_2, r_1)) = A_1(q_2, f_1(r_2), q_2) \wedge A_2(r_2, a, r_1) = 0.3,$$
$$A^0((q_1, r_1), (f_2, a), (q_2, r_2)) = A_1(q_1, f_2(r_1), q_2) \wedge A_2(r_1, a, r_2) = 0.1,$$
$$A^0((q_2, r_1), (f_2, a), (q_2, r_2)) = A_1(q_2, f_2(r_1), q_2) \wedge A_2(r_1, a, r_2) = 0.3,$$
$$A^0((q_1, r_2), (f_2, a), (q_1, r_1)) = A_1(q_1, f_2(r_2), q_1) \wedge A_2(r_2, a, r_1) = 0.2,$$
$$A^0((q_2, r_2), (f_2, a), (q_1, r_1)) = A_1(q_2, f_2(r_2), q_1) \wedge A_2(r_2, a, r_1) = 0.4,$$
$$A^0((q_1, r_1), (f_3, a), (q_1, r_2)) = A_1(q_1, f_3(r_1), q_1) \wedge A_2(r_1, a, r_2) = 0.2,$$
$$A^0((q_2, r_1), (f_3, a), (q_1, r_2)) = A_1(q_2, f_3(r_1), q_1) \wedge A_2(r_1, a, r_2) = 0.4,$$
$$A^0((q_1, r_2), (f_3, a), (q_2, r_1)) = A_1(q_1, f_3(r_2), q_2) \wedge A_2(r_2, a, r_1) = 0.1,$$
$$A^0((q_2, r_2), (f_3, a), (q_2, r_1)) = A_1(q_2, f_3(r_2), q_2) \wedge A_2(r_2, a, r_1) = 0.3,$$
$$A^0((q_1, r_1), (f_4, a), (q_1, r_2)) = A_1(q_1, f_4(r_1), q_1) \wedge A_2(r_1, a, r_2) = 0.2,$$
$$A^0((q_2, r_1), (f_4, a), (q_1, r_2)) = A_1(q_2, f_4(r_1), q_1) \wedge A_2(r_1, a, r_2) = 0.4$$
$$A^0((q_1, r_2), (f_4, a), (q_1, r_1)) = A_1(q_1, f_4(r_2), q_1) \wedge A_2(r_2, a, r_1) = 0.2,$$
$$A^0((q_2, r_2), (f_4, a), (q_1, r_1)) = A_1(q_2, f_4(r_2), q_1) \wedge A_2(r_2, a, r_1) = 0.4.$$

Definition 8.3.3 Assume that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton and that $q, r \in Q$. Then, r is called an *immediate successor* of q if there is an $a \in \Sigma$ such that $A(q, a, r) > 0$. Also, r is called a *successor* of q if there is a $\sigma \in \Sigma^*$ such that $A^*(q, \sigma, r) > 0$.

Proposition 8.3.1 *Assume that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton and that $q, r \in Q$. Then, the following are true:*

(i) q is a successor of q; and
(ii) if r is a successor of q and s is a successor of r, then s is a successor of q.

Definition 8.3.4 Assume that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton and that $q \in Q$. Then, the set of all successors of q are denoted by $S(q)$.

Definition 8.3.5 Assume that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton and that $T \subseteq Q$. Then, the set of all successors of T, which is denoted by $S_Q(T)$, is the set

$$S_Q(T) = \bigcup \{S(q) | q \in T\}.$$

Definition 8.3.6 Assume that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton and that $T \subseteq Q$. Also, suppose that ν is a fuzzy subset of $T \times \Sigma \times T$ and $N = (T, \Sigma, \nu)$. Then, the fuzzy finite state automaton N is called a *subautomaton* of M if

(i) $A|_{T \times \Sigma \times T} = \nu$ (i.e. ν is a restriction of A), and
(ii) $S_Q(T) \subseteq T$.

Subsystems and strong subsystems are special cases of subautomata:

Definition 8.3.7 Assume that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton and that δ is a fuzzy subset of Q. Then, (Q, δ, Σ, A) is called a *subsystem* of M if for all $q, r \in Q$ and for all $a \in \Sigma$,

$$\delta(q) \geq \delta(r) \geq A(r, a, q).$$

Theorem 8.3.1 *Assume that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton and that δ is a fuzzy subset of Q. Then, (Q, δ, Σ, A) is a subsystem of M if and only if for all $r, q \in Q$ and for all $\sigma \in \Sigma^*$,*

$$\delta(q) \geq \delta(r) \wedge A^*(r, \sigma, q).$$

Proof: Let (Q, δ, Σ, A) be a subsystem. Also, assume that $q, r \in Q$ and $\sigma \in \Sigma^*$. The result is proved by induction on the length of σ, that is, $|\sigma| = n$. If $n = 0$, then $\sigma = \epsilon$. If $r = q$, then $\delta(q) \wedge A^*(q, \epsilon, q) = \delta(q)$. If $q \neq r$, then $\delta(r) \wedge A^*(r, \epsilon, q) = 0 \leq \delta(q)$. Thus, the result is true when $n = 0$. Assume that the result is true for all

$\tau \in \Sigma^*$, such that $|\tau| = n-1$ and $n > 0$. Suppose that $\sigma = \tau a$, $|\tau| = n-1$, $\tau \in \Sigma^*$, and $a \in \Sigma$. Then,

$$\begin{aligned}
\delta(r) \wedge A^*(r, \sigma, q) &= \delta(r) \wedge A^*(r, \tau a, q) \\
&= \delta(r) \wedge \left(\bigvee \{A^*(r, \tau, s) \wedge A(s, a, q) | s \in Q\} \right) \\
&= \bigvee \{\delta(r) \wedge A^*(r, \tau, s) \wedge A(s, a, q)| \\
&\leq \bigvee \{\delta(r) \wedge A(s, a, q) | s \in Q\} \\
&\leq \delta(q).
\end{aligned}$$

This means that $\delta(q) \geq \delta(r) \wedge A^*(r, \sigma, q)$. The proof of the converse is trivial. □

Definition 8.3.8 Assume that $M = (Q, \Sigma, A)$ is a fuzzy finite state automaton and δ is a fuzzy subset of Q. Then, (Q, δ, Σ, A) is a *strong subsystem* of M if and only if for all $r, q \in Q$ and if there exists an $a \in \Sigma$ such that $A(r, a, q) > 0$, then $\delta(q) \geq \delta(r)$.

We can also compose automata.

Definition 8.3.9 Assume that $M_1 = (Q_1, \Sigma_1, A_1)$ and $M_2 = (Q_2, \Sigma_2, A_2)$ are two fuzzy finite state automata. Also, assume that $\Sigma_1 \cap \Sigma_2 = \emptyset$. Let

$$M_1 \cdot M_2 = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, A_1 \cdot A_2),$$

where

$$(A_1 \cdot A_2)((r_1, r_2), a, (q_1, q_2)) = \begin{cases} A_1(r_1, a, q_1), & \text{if } a \in \Sigma_1 \text{ and } r_2 = q_2, \\ A_2(r_2, a, q_2), & \text{if } a \in \Sigma_2 \text{ and } r_1 = q_1, \\ 0, & \text{otherwise.} \end{cases}$$

for all $(r_1, r_2), (q_1, q_2) \in Q_1 \times Q_2$, $a \in \Sigma_1 \cup \Sigma_2$. Then, $M_1 \cdot M_2$ is a fuzzy finite state automaton and it is called the *Cartesian composition* of M_1 and M_2.

Definition 8.3.10 Suppose that $M_1 = (Q_1, \Sigma_1, A_1)$ and $M_2 = (Q_2, \Sigma_2, A_2)$ are two fuzzy finite state automata such that $Q_1 \cap Q_2 = \emptyset$ and $\Sigma_1 \cap \Sigma_2 = \emptyset$. Then, the fuzzy finite state automaton

$$M_1 + M_2 = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, A_1 + A_2),$$

where

$$(A_1 + A_2)(r, x, q) = \begin{cases} A_1(r, a, q), & \text{if } a \in \Sigma_1 \text{ and } r, q \in Q_1, \\ A_2(r, a, q), & \text{if } a \in \Sigma_2 \text{ and } r, q \in Q_2, \\ 0, & \text{otherwise.} \end{cases}$$

is called the *sum* of M_1 and M_2.

Mansoor Doostfatemeh and Stefan C. Kremer [104] had presented a new definition for fuzzy automata. A number of reasons (i.e. membership assignment, output mapping, multi-membership resolution, and the concept of acceptance for fuzzy automata) necessitated the introduction of this new definition:

Definition 8.3.11 An octuple $(Q, \Sigma, \tilde{R}, Z, \omega, \tilde{\delta}, F_1, F_2)$ is a *general fuzzy automaton* (GFA) $\tilde{F}$, where

- Q is a finite set of states, $Q = \{q_1, q_2, \dots, q_n\}$;
- Σ is a finite crisp set of input symbols, $\Sigma = \{a_1, a_2, \dots, a_m\}$;
- $\tilde{R} \subseteq \mathscr{F}(Q)$ is the set of fuzzy start states;
- Z is a finite crisp set of output symbols, $Z = \{b_1, b_2, \dots, b_k\}$;
- $\omega : Q \to Z$ is the output function that maps a (fuzzy) state to the output set;
- $F_1 : [0,1] \times [0,1] \to [0,1]$ is a fuzzy binary relation in μ, the set of membership values of a predecessor, and δ, the active states (it is called *membership assignment function*);
- $\tilde{\delta} : (Q \times [0,1]) \times \Sigma \times Q \xrightarrow{F_1(\mu,\delta)} [0,1]$ is the augment transition function; and
- $F_2 : [0,1]^* \to [0,1]$ is the multi-membership resolution function. If there are several simultaneous transitions to the active state q_m at time $t+1$, then F_2 is a function that specifies the *strategy* that resolves the multi-membership active states and assigns a single membership value to them.

The authors have presented a multi-membership resolution algorithm, but we are not going to present it here. But let us see how some simple fuzzy automata can be described as general fuzzy automata.

Example 8.3.4 A fuzzy finite automaton with final states is a GFA with $Z = \{\text{accept}, \text{reject}\}$ such that $\omega : Q \to Z$ is defined as follows:

$$\omega(q_i) = \begin{cases} \text{accept}, & \text{if } q_i \in Q_f, \\ \text{reject}, & \text{if } q_i \notin Q_f. \end{cases}$$

Here Q_f is the set of terminal states.

8.4 Fuzzy Turing Machines

The first general description of a fuzzy Turing machine was given by Zadeh [306]. He presumed that a *fuzzy algorithm* should contain vague commands (Zadeh used the term "fuzzy commands," but in order to be consistent with the terminology

used in this book, we will call them "vague commands"). According to Zadeh, the following are examples of simple vague commands:

(i) Set y *approximately equal to* 10, if x is *approximately equal to* 5.
(ii) If x is *large*, increase y by *several* units.
(iii) If x is *large*, increase y by *several* units; if x is *small*, decrease y by *several* units; otherwise keep y unchanged.

This command is vague, because the text that appears slanted corresponds to fuzzy sets. For example, the numbers that are approximately equal to 10 or 5 are two different fuzzy sets. Based on this, Zadeh vaguely described a fuzzy Turing machine as one where instructions are performed to a degree. A number of concrete proposals followed (see Ref. [275] for a detailed presentation of all these proposals), however, we will only present the one by Jiří Wiedermann [298] since it is the most complete presentation and the most interesting of all previous attempts. Wiedermann's machine is a nondeterministic fuzzy Turing machine with hypercomputational capabilities. In a nutshell, Wiedermann has shown that his machine can solve problems no ordinary Turing machine can solve.

Definition 8.4.1 A nondeterministic fuzzy Turing machine with a unidirectional tape is a nonuple

$$\mathscr{F} = (Q, T, I, \Delta, \_, q_0, q_f, \mu, *),$$

where:

- Q is a finite set of states;
- T is a finite set of tape symbols;
- I is a set of input symbols, where $I \subseteq T$;
- Δ is a transition relation and it is a subset of $Q \times T \times Q \times T \times \{L, N, R\}$. Each action that the machine takes is associated with an element $\delta \in \Delta$. In particular, for $\delta = (q_i, t_i, q_{i+1}, t_{i+1}, d)$ this means that, when the machine is in state q_i and the current symbol that has been read is t_i, then the machine will enter state q_{i+1}, the symbol t_{i+1} will be printed on the current cell and the scanning head will move according to the value of d, that is, if d is L, N, or R, then the head will move one cell to the left, will not move, or it will move one cell to the right, respectively.
- $\_ \in T \setminus I$ is the blank symbol;
- q_0 and q_f are the initial and the final state, respectively;
- $\mu : \Delta \to [0, 1]$ is a fuzzy relation on Δ; and
- $*$ is a t-norm.

Definition 8.4.2 When μ is a partial function from $Q \times T$ to $Q \times T \times \{L, N, R\}$ and T is a fuzzy subset of Q, then the resulting machine is called a deterministic fuzzy Turing machine.

A configuration gives

(i) the position of the scanning head,
(ii) what is printed on the tape, and
(iii) the current state of the machine.

If S_i and S_{i+1} are two configurations, then $S_i \vdash^{\alpha} S_{i+1}$ means that S_{i+1} is reachable in one step from S_i with a plausibility degree that is equal to α if and only if there is a $\delta \in \Delta$ such that $\mu(\delta) = \alpha$, and by which the machine goes from S_i to S_{i+1}. When a machine starts with input some string w, the characters of the string are printed on the tape starting from the leftmost cell; the scanning head is placed atop the leftmost cell, and the machine enters state q_0. If

$$S_0 \vdash^{\alpha_0} S_1 \vdash^{\alpha_1} S_2 \vdash^{\alpha_2} \cdots \vdash^{\alpha_{n-1}} S_n,$$

then S_n is reachable from S_0 in n steps. Assume that S_n is reachable from S_0 in n steps, then the plausibility degree of this *computational path* is

$$D((S_0, S_1, \ldots, S_n)) = \begin{cases} 1, & n = 0, \\ D((S_0, S_1, \ldots, S_{n-1})) * \alpha_{n-1}, & n > 0. \end{cases}$$

Obviously, the value that is computed with this formula depends on the specific path that is chosen. Since the machine is nondeterministic, it is quite possible that some configuration S_n can be reached via different computational paths. Therefore, when a machine starts from S_0 and finishes at S_n in n steps, the plausibility degree of this computational path, which is called a *computation*, should be equal to the maximum of all possible computation paths:

$$d(S_n) = \max[D((S_0, S_1, \ldots, S_n))].$$

In other words, the plausibility degree of the computation is equal to the plausibility degree of the computational path that is most likely to happen.

Assume that a machine starts from configuration S_0 with w as input. Then, a computational path $S_0, S_1, \ldots, S_m$ is an accepting path of configurations if the state of S_m is q_f. In addition, the string w is accepted with degree equal to $d(S_m)$.

Definition 8.4.3 Assume that $\mathscr{F}$ is a fuzzy nondeterministic Turing machine. Then, an input string w is accepted with plausibility degree $e(w)$ by $\mathscr{F}$ if and only if:

- there is an accepting configuration from the initial configuration S_0 on input w;
- $e(w) = \max_S\{d(S) | S \text{ is an accepting configuration reachable from } S_0\}$.

Also,

Definition 8.4.4 The fuzzy language accepted by some machine $\mathscr{F}$ is the fuzzy set that is defined as follows:

$$L(\mathscr{F}) = \{(w, e(w)) | w \text{ is accepted by } \mathscr{F} \text{ with plausibility degree } e(w)\}.$$

The class of all fuzzy languages accepted by a fuzzy Turing machine, in the sense just explained, with (classically) computable t-norms is denoted by Φ.

Theorem 8.4.1 $\Phi = \Sigma_1^0 \cup \Pi_1^0$, *that is,* Φ *is the union of the recursively enumerable and the co-recursively enumerable languages.*

Proof: (Sketch) In order to show that $\Phi = \Sigma_1^0 \cup \Pi_1^0$, one has to prove that $\Sigma_1^0 \cup \Pi_1^0 \subseteq \Phi$ and, at the same time, that $\Phi \subseteq \Sigma_1^0 \cup \Pi_1^0$. Assume that L is a language such that $L \in \Sigma_1^0 \cup \Pi_1^0$. Then either $L \in \Sigma_1^0$ or $L \in \Pi_1^0$. When L is recursively enumerable, there is a nonfuzzy machine $\mathscr{M}$ that is able to semidecide L. In what follows, it will be shown that for each constant $0 \leq c < 1$ there is a fuzzy Turing machine $\mathscr{F}$, whose acceptance criterion is given in Definition 8.4.4, such that $w \in L$ if and only if w is accepted by $\mathscr{F}$ with plausibility degree equal to 1, that is, $(w, 1) \in L(\mathscr{F})$, and $w \notin L$ if and only if $(w, c) \in L(\mathscr{F})$. Given any t-norm and $0 \leq c < 1$, one can specify $\mathscr{F}$ as follows: unless it is explicitly stated, by default all commands will have plausibility degree that is equal to 1. Suppose that w is the input to $\mathscr{F}$. Then, this machine will make a nondeterministic branch and one path will lead to the simulation of $\mathscr{M}$. In addition, when $\mathscr{M}$ enters an accepting state, $\mathscr{F}$ will enter an accepting state q with plausibility degree equal to one. Also, the other path leads to an accepting state q' with plausibility degree equal to c. Now, if $w \in L$, then both q and q' are reached. In addition, w will be accepted in q with plausibility degree equal to one (i.e. $(w, 1) \in L(\mathscr{F})$). If $w \notin L(\mathscr{F})$, then the machine will not enter any accepting state but r, and so $(w, c) \in L(\mathscr{F})$. Assume now that L is co-recursively enumerable and so $\overline{L}$ is recursively enumerable. This implies that there is a machine $\mathscr{M}'$ that recognizes $\overline{L}$. Following an argument similar to the one presented so far, one can show that for each constant $0 \leq c < 1$ there is a fuzzy Turing machine $\mathscr{G}$ such that $w \in \overline{L}$ if and only if w is accepted by $\mathscr{G}$ with plausibility degree equal to one (i.e. $(w, 1) \in L(\mathscr{G})$) and $w \notin \overline{L}$ if and only if $(w, c) \in L(\mathscr{G})$. In other words, $w \notin L$ if and only $(w, 1) \in L(\mathscr{G})$, and $w \in L$ if and only if $(w, c) \in L(\mathscr{G})$. And this concludes the first part of the proof. Let us now proceed with the second part.

Now it will be shown that for any fuzzy Turing machine $\mathscr{F} = (Q, T, I, \Delta, \_, q_0, q_f, \mu, *)$ with a computable t-norm, there are crisp languages $L_1 \in \Sigma_1^0$ and $L_2 \in \Pi_1^0$ such that $(w, d) \in L(\mathscr{F})$ if and only if $w\#d \in L_1 \cap \in L_2 \Sigma_1^0 \cup \Pi_1^0$ for any $w \in (I \setminus \{\#\})^*$ and $\# \in T$. Assume that F is a fuzzy Turing machine, where all commands have plausibility degrees that are equal to one (essentially, this machine is a nondeterministic Turing machine). Also, assume that $ACF_{\mathscr{F}}(w)$ is a set that contains all accepting configurations of $\mathscr{F}$ on input w, and that e is the evaluation function that assigns to each accepting configuration its plausibility degree. In addition, consider the commutative ordered semigroup $G = ([0, 1], *, \leq)$, where $*$ is a t-norm and $\leq$ is the usual ordering relation. Furthermore, let

$\mathbb{Q} \supset J = \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset [0,1]$, where $\alpha_1 < \alpha_2 < \cdots < \alpha_k$, be the range of μ and $G(J)$ be a subsemigroup of G generated by replacing $[0,1]$ with a set that has as elements all elements of J and all elements produced by the t-norm, that is, when α_m and α_n are elements of this set, then $\alpha_m * \alpha_n$ also belongs to this set. It is relatively easy to see that

$$\Sigma_1^0 \ni L_1 = \{w\#d | \exists a \in ACF_{\mathscr{F}}(w) : e(a) = d \in G(J)\},$$
$$\Pi_1^0 \ni L_2 = \{w\#d | \forall a \in ACF_{\mathscr{F}}(w) : e(a) \leq d \in G(J)\}.$$

In particular, in order to show that $L_1 \in \Sigma_1^0$, one should consider an ordinary nondeterministic machine $\mathscr{M}_1$, which on input $w\#d$, where $d = \alpha_n * \alpha_m$, first guesses the numbers α_n and α_m, then computes $\alpha_n * \alpha_m$, and checks whether $d \in G(J)$. Next, $\mathscr{M}_1$ guesses a and simulates F on w to see whether $a \in ACF_{\mathscr{F}}(w)$ and $e(a) = d$. In order to show that $L_2 \in \Pi_1^0$, one should consider an ordinary nondeterministic machine M_2 that accepts the language

$$\overline{L}_2 = \{w\#d \mid \exists a \in ACF_{\mathscr{F}}(w) : e(a) > d\}.$$

By a similar argument, one concludes that $\overline{L}_2 \in \Sigma_1^0$ and so $L_2 \in \Pi_1^0$. Furthermore, it is clear that $(w,d) \in L(\mathscr{F})$ if and only if $w\#d \in L_1 \cap L_2$, because the second condition is just a reformulation of conditions for $\mathscr{F}$ to accept w with plausibility degree equal to d according to 8.4.4. □

There have been some arguments against the validity of this proof, but we will not discuss it further (see Ref. [275] for details). In addition, there are some results (e.g. the existence of a universal fuzzy Turing machine) that are discussed in [275, 278].

8.5 Other Fuzzy Models of Computation

Fuzzy Turing machines and fuzzy automata are not the only fuzzy models of computation, however, they are the ones that keep busy most researchers. Indeed, there are models of computation that are based on *fuzzy multisets*.

Definition 8.5.1 Let X be a universe. A *multiset* A of the set X, is characterized by a function $A : X \to \mathbb{N}$. For every $x \in X$, the value $A(x)$ denotes the number of times x belongs to A.

Definition 8.5.2 Assume X is a set of elements. Then, a fuzzy multiset A drawn from X is characterized by a function $A : X \to \mathbb{N}^{[0,1]}$. The expression $\mathbb{N}^{[0,1]}$ is the set of all multisets of $[0,1]$.

It is not difficult to see that any fuzzy multiset A is actually characterized by a function

$$A : X \times [0,1] \to \mathbb{N},$$

which is obtained from the former function by *uncurrying* it. However, one can demand that for each element x, there is only one membership degree and one multiplicity. In other words, a "fuzzy multiset" A should be characterized by a function $X \to [0,1] \times \mathbb{N}$. To distinguish these structures from fuzzy multisets, we will call them *multi-fuzzy* sets [271]. Given a multi-fuzzy set A, the expression $A(x) = (i, n)$ denotes that there are n copies of x that belong to A with a degree that is equal to i. A generalization of this definition was presented in [273]:

Definition 8.5.3 An L-fuzzy hybrid set $\mathscr{A}$ is a mathematical structure that is characterized by a function $\mathscr{A} : X \to L \times \mathbb{Z}$, where L is a frame, and it is associated with an L-fuzzy set $A : X \to L$. More specifically, the equality $\mathscr{A}(x) = (\ell, n)$ means that $\mathscr{A}$ contains exactly n copies of x, where $A(x) = \ell$.

By substituting $\mathbb{Z}$ with $\mathbb{N}$ in the previous definition, the resulting structures will be called L-multi-fuzzy sets.

Assuming that $\mathscr{A}$ is an L-fuzzy hybrid set, then we can define the following two functions: the *multiplicity function* $\mathscr{A}_m : X \to \mathbb{Z}$ and the *degree function* $\mathscr{A}_\mu : X \to L$. Clearly, if $\mathscr{A}(x) = (\ell, n)$, then $\mathscr{A}_m(x) = n$ and $\mathscr{A}_\mu(x) = \ell$. Note that it is equally easy to define the corresponding functions for an L-multi-fuzzy set.

Definition 8.5.4 Assume that $\mathscr{A}$ is an L-fuzzy hybrid set that draws elements from a universe X. Then, its cardinality is defined as follows:

$$\operatorname{card} \mathscr{A} = \sum_{x \in X} \mathscr{A}_\mu(x) \otimes \mathscr{A}_m(x),$$

where $\otimes : L \times \mathbb{Z} \to \mathbb{R}$ is a binary multiplication operator that is used to compute the product of $\ell \in L$ "times" $n \in \mathbb{Z}$.

Example 8.5.1 If $L = [0,1] \times [0,1]$ (i.e, when extending "intuitionistic" fuzzy sets), then $(i, j) \times n = in - jn$.

Remark 8.5.1 When L is the interval $[0,1]$, then $\otimes$ is the usual multiplication operator.

In order to give the basic properties of fuzzy hybrid sets, it is necessary to define the notion of subsethood. Before, going on with this definition, we will introduce

the partial order $\ll$ over $\mathbb{Z}$, which is defined as follows, for all $n, m \in \mathbb{Z}$:

$$\begin{aligned} n \ll m \equiv\ & (n = 0) \vee \\ & ((n > 0) \wedge (m > 0) \wedge (n \leq m)) \vee \\ & ((n < 0) \wedge (m > 0)) \vee \\ & (|n| \leq |m|). \end{aligned}$$

Here $\wedge$ and $\vee$ denote the classical logical conjunction and disjunction operators, respectively. In addition, the symbols $\leq$ and $<$ are the well-known ordering operators, and $|n|$ is the absolute value of n.

Example 8.5.2 From Definition 8.5.4 it should be obvious that $0 \ll n$, for all $n \in \mathbb{Z}$. Also, $3 \ll 4$, $-3 \ll 4$, and $-4 \ll -3$.

Proposition 8.5.1 *The relation $\ll$ is a partial order.*

Proof: We have to prove that the relation $\ll$ is reflexive, antisymmetric, and transitive:

Reflexivity. Assume that $a \in \mathbb{Z}$. If $a = 0$, then $a \ll a$ from the first part of the disjunction. If $a < 0$, then $a \ll a$ from the fourth part of the disjunction, and if $a > 0$, then $a \ll a$ from the second part of the disjunction.

Antisymmetry. Assume that $a, b \in \mathbb{Z}$, $a \ll b$, and $b \ll a$. Then if $a = 0$, this implies that $b = 0$ and so $a = b$. If $a < 0$, then it follows that $b < 0$, $|a| \leq |b|$, and $|b| \leq |a|$, which implies that $a = b$. Similarly, if $a > 0$, then it follows that $b > 0$, $a \leq b$, and $b \leq a$, which implies that $a = b$.

Transitivity. Assume that $a, b, c \in \mathbb{Z}$, $a \ll b$, and $b \ll c$. If $a = 0$, then clearly $a \ll c$. If $a < 0$ and $b < 0$, then either $c < 0$ or $c > 0$, but since $|b| \leq |c|$, this implies that $a \ll c$. If $a > 0$ and $b > 0$, then $c > 0$ and since $b \leq c$ this implies that $a \ll c$. If $a < 0$ and $b > 0$, then since $b \ll c$, this implies that $c > 0$, which means that $a \ll c$. □

Note that $a \gg b$ is an alternative form of $b \ll a$, which will be used in the rest of this section. Let us now proceed with the definition of the notion of subsethood for L-fuzzy hybrid sets:

Definition 8.5.5 Assume that $\mathscr{A}, \mathscr{B} : X \to L \times \mathbb{Z}$ are two L-fuzzy hybrid sets. Then, $\mathscr{A} \subseteq \mathscr{B}$ if and only if $\mathscr{A}_\mu(x) \sqsubseteq \mathscr{B}_\mu(x)$ and $\mathscr{A}_m(x) \ll \mathscr{B}_m(x)$ for all $x \in X$.

Note that for all $\ell_1, \ell_2 \in L$, $\ell_1 \sqsubseteq \ell_2$ if ℓ_1 is "less than or equal" to ℓ_2 in the sense of the partial order defined over L. The definition of subsethood for L-multi-fuzzy sets is more straightforward:

Definition 8.5.6 Assume that $\mathscr{A}, \mathscr{B} : X \to L \times \mathbb{N}$ are two L-multi-fuzzy sets. Then $\mathscr{A} \subseteq \mathscr{B}$ if and only if $\mathscr{A}_\mu(x) \sqsubseteq \mathscr{B}_\mu(x)$ and $\mathscr{A}_m(x) \leq \mathscr{B}_m(x)$ for all $x \in X$.

Let us now present the definitions of union and sum of L-multi-fuzzy sets:

Definition 8.5.7 Assuming that $\mathscr{A}, \mathscr{B} : X \to L \times \mathbb{N}$ are two L-multi-fuzzy sets, then their union, denoted $\mathscr{A} \cup \mathscr{B}$, is defined as follows:

$$(\mathscr{A} \cup \mathscr{B})(x) = (\mathscr{A}_\mu(x) \sqcup \mathscr{B}_\mu(x), \max\{\mathscr{A}_m(x), \mathscr{B}_m(x)\}),$$

where $a \sqcup b$ is the join of $a, b \in L$.

Definition 8.5.8 Suppose that $\mathscr{A}, \mathscr{B} : X \to L \times \mathbb{N}$ are two L-multi-fuzzy sets. Then their sum, denoted $\mathscr{A} \uplus \mathscr{B}$, is defined as follows:

$$(\mathscr{A} \uplus \mathscr{B})(x) = (\mathscr{A}_\mu(x) \sqcup \mathscr{B}_\mu(x), \mathscr{A}_m(x) + \mathscr{B}_m(x)).$$

P system is a model of computation, inspired by the way cells live and function. The model is built around the notion of nested compartments surrounded by porous *membranes* (hence the term *membrane computing*). It is quite instructive to think of the membrane structure as a bubbles-inside-bubbles structure, where we have a bubble that contains bubbles, which, in turn, contain other bubbles, etc. Initially, each compartment contains a number of possible repeated objects (i.e. a multiset of objects). Once "computation" commences, the compartments exchange objects according to a number of multiset processing rules that are associated with each compartment; in the simplest case, these processing rules are just multiset rewriting rules. The activity stops when no rule can be applied anymore. The result of the computation is equal to the number of objects that reside within a designated compartment called the *output membrane*.

In [271], a fuzzified version of P systems was presented. The basic idea behind this particular attempt to fuzzify P systems is the substitution of one or all ingredients of a P system with their fuzzy counterparts. From a purely computational point of view, it turns out that only P systems that process multi-fuzzy sets are interesting, the reason being the fact that these systems are capable of computing (positive) real numbers. By replacing the multi-fuzzy sets employed in the first author's previous work with L-multi-fuzzy sets, the computational power of the resulting P systems will not be any "greater," nevertheless, these systems may be quite useful in modeling living organisms. But, things may get really interesting if

we consider P systems with L-fuzzy hybrid sets, in general. Here we are going to give only the definition of these systems. For a full exposition see Ref. [275].

Definition 8.5.9 A general fuzzy P system is a construction

$$\Pi_{\mathrm{FD}} = (O, \mu, w^{(1)}, \dots, w^{(m)}, R_1, \dots, R_m, i_0),$$

where:

(i) O is an alphabet (i.e. a set of distinct entities) whose elements are called *objects*;
(ii) μ is the membrane structure of degree $m \geq 1$; membranes are injectively labeled with succeeding natural numbers starting with one;
(iii) $w^{(i)} : O \to L \times \mathbb{Z}, 1 \leq i \leq m$, are L-fuzzy hybrid sets over O that are associated with each region i;
(iv) $R_i, 1 \leq i \leq m$, are finite sets of multiset rewriting rules (called *evolution rules*) over O. An evolution rule is of the form $u \to v$, $u \in O^*$ and $v \in O^*_{\mathrm{TAR}}$, where $O_{\mathrm{TAR}} = O \times \mathrm{TAR}$, $\mathrm{TAR} = \{\mathrm{here}, \mathrm{out}\} \cup \{\mathrm{in}_j | 1 \leq j \leq m\}$. The effect of each rule is the removal of the elements of the left-hand side of each rule from the "current" compartment and the introduction of the elements of right-hand side to the designated compartments;
(v) $i_0 \in \{1, 2, \dots, m\}$ is the label of an elementary membrane (i.e. a membrane that does not contain any other membrane), called the *output* membrane.

Fuzzy multisets have been also used to define *fuzzy multiset grammars* (see Ref. [259] and references therein). These grammars are recognized by special kinds of automata. In what follows, $\Sigma^{\uplus}$ will denote the set of all multisets whose universe is the set Σ. Also, given two multisets $A, B : X \to \mathbb{N}$, then $(A \uplus B)(x) = A(x) + B(x)$ and $(A \ominus B)(x) = \max(0, A(x) - B(x))$.

Definition 8.5.10 A *fuzzy multiset finite automaton* (FMFA) is a quintuple $M = (Q, \Sigma, \delta, \sigma, \tau)$, where

- Q and Σ are nonempty finite sets called the *state-set* and *input-set*, respectively;
- $\delta : Q \times \Sigma^{\uplus} \times Q \to [0, 1]$ is a map called *transition map*;
- $\sigma : Q \to [0, 1]$ is a map called the *fuzzy set of initial states*; and
- $\tau : Q \to [0, 1]$ is a map called the *fuzzy set of final states*.

A *configuration* of a fuzzy multiset finite automaton M is a pair (p, α), where p and α denote the current state and the current multiset, respectively. Transitions in a fuzzy multiset finite automaton are described with the help of configurations. The transition from configuration (p, α) leads to configuration (q, β)

with membership degree $k \in [0,1]$ if there exists a multiset $\gamma \in \Sigma^{\Psi}$ with $\gamma \subseteq \alpha$, $\delta(p,\gamma,q) = k$ and $\beta = \alpha \ominus \gamma$. This transition is written as $(p,\alpha) \xrightarrow{k} (q,\beta)$.

Definition 8.5.11 For a given set Σ, a *fuzzy multiset language* is a map $L : \Sigma^{\Psi} \to [0,1]$. Let $M = (Q, \Sigma, \delta, \sigma, \tau)$ be a fuzzy multiset (finite) automaton. Then, the set

$$L(M) = \{\alpha \in \Sigma^{\Psi} | \alpha \text{ is accepted by } M\}$$

is called the fuzzy multiset language of M. A fuzzy multiset language $L : \Sigma^{\Psi} \to [0,1]$ is *accepted* by an FMFA $M = (Q, \Sigma, \delta, \sigma, \tau)$ if

$$L(\alpha) = \bigvee \{\sigma(q) \wedge \mu_M((q,\alpha) \to * (p, 0_{\Sigma})) \wedge \tau(p) : q, p \in Q\},$$

for all $\alpha \in \Sigma^{\Psi}$. Furthermore, a fuzzy multiset language $L : \Sigma^{\Psi} \to [0,1]$ is called *regular* if there exists a FMFA M such that $L = L(M)$.