Encoding Nondeterministic Fuzzy Tree Automata Into Recursive Neural Networks

Marco Gori, *Fellow, IEEE,* and Alfredo Petrosino, *Member, IEEE*

***Abstract*—Fuzzy neural systems have been a subject of great interest in the last few years, due to their abilities to facilitate the exchange of information between symbolic and subsymbolic domains. However, the models in the literature are not able to deal with structured organization of information, that is typically required by symbolic processing. In many application domains, the patterns are not only structured, but a fuzziness degree is attached to each subsymbolic pattern primitive. The purpose of this paper is to show how recursive neural networks, properly conceived for dealing with structured information, can represent nondeterministic fuzzy frontier-to-root tree automata. Whereas available prior knowledge expressed in terms of fuzzy state transition rules are injected into a recursive network, unknown rules are supposed to be filled in by data-driven learning. We also prove the stability of the encoding algorithm, extending previous results on the injection of fuzzy finite-state dynamics in high-order recurrent networks.**

***Index Terms*—Fuzzy neural networks, fuzzy systems, fuzzy tree automata, knowledge representation, recursive neural networks.**

I. Introduction

THE general tendency of merging fuzzy systems and neural networks appeared quite early in the development of fuzzy sets [22]. Bearing in mind an evident overlap in the architecture characteristic for fuzzy sets and neural networks, these two complementary technologies can produce an efficient synergy for the design of "intelligent" procedures. Several researchers have considered the possibility of integrating the advantages of fuzzy systems with those not less known of neural networks, giving rise to fuzzy neural networks (see [21] and [24] as representative sources to this regards). These kinds of studies are very interesting in all the application domains where the patterns are strongly correlated through structure and the processing is both numerical and symbolic, without discarding the component of structure which relates different portions of numerical data and the imprecise and incomplete nature of the data. First application perspectives are: automated reasoning and theorem proving [17], computational chemistry [2], [32], logo recognition [7], but other ones include medical and technical diagnoses, molecular biology, software engineering, geometrical and spatial reasoning, speech and natural processing and different applications of pattern recognition (see [10] for detailed considerations on these application domains).

For instance, let us consider problems of syntactic pattern recognition [11], [12]. Certain pattern classes contain objects, such as geometric figures, with an identifiable hierarchical structure that can be described by a formal grammar. A basic set of pattern primitives is selected and forms the set of grammar terminals. Grammar productions are a list of allowable relations among the primitives, while the pattern class is the set of strings generated by the grammar or the equivalent automaton. When the indeterminacy is due to inherent vagueness, the imprecise nature of patterns could be handled by fuzzy languages; the fuzziness may lie in the definition of primitives or in the physical relations among them. Thus, the primitives become labels of fuzzy sets and the production rules of the grammar are weighed. Fuzzy languages have shown some promise in dealing with patterns which possess ill-defined (fuzzy) boundaries [26], [31], [33], [37], [39]. The patterns are not only structured (usually tree structured), but each pattern primitive has a subsymbolic nature and lastly a fuzziness degree is attached to it. From the bottom to the top the pattern evolves in structure; this leads to consider more refined pattern representations and description. A pattern may be also represented at various resolution levels by a graph of primitives and their relations, and a grammar, whose production rules describe the evolution of the object primitives at increasing resolution levels, is introduced [3], [5].

In Figs. 1 and 2, two examples are sketched.

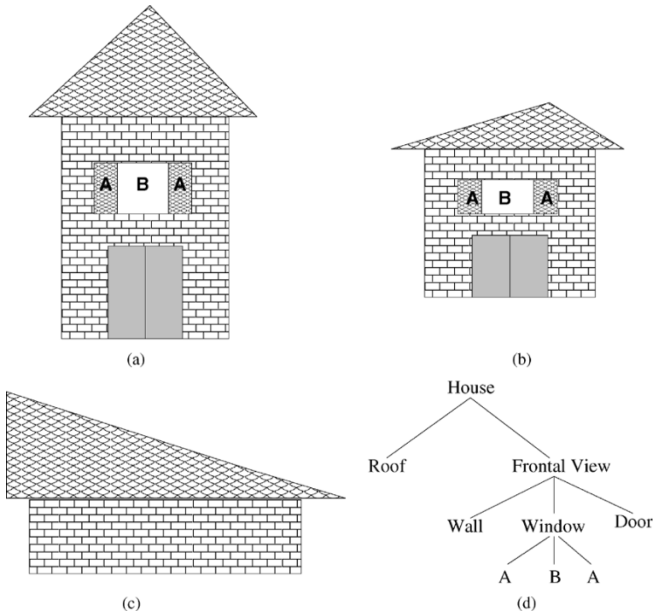

Fig. 1. Some examples of "houses" (a-c) and the tree structural representation (d). A real valued function μ is attached to each vertex to express the degree of membership of the associated concept. As example, denoting with $\mathrm{triangle}_a$, $\mathrm{triangle}_b$, and $\mathrm{triangle}_c$ the labels associated to the roof component of the trees, respectively, corresponding to the examples (a), (b), and (c), the membership values to the roof concept satisfy $\mu_{\mathrm{roof}}(\mathrm{triangle}_a) > \mu_{\mathrm{roof}}(\mathrm{triangle}_b) > \mu_{\mathrm{roof}}(\mathrm{triangle}_c)$.

Manuscript received April 2, 2002; revised September 1, 2003. This work was supported in part by Legge n. 5/2002 Regione Campania and in part by the National Research Council (CNR).

M. Gori is with Dipartimento di Ingegneria dell'Informazione, Università di Siena, 53100 Siena, Italy (e-mail: marco@dii.unisi.it).

A. Petrosino is with National Research Council, ICAR, Section of Naples, Complesso Università Monte Sant'Angelo, 80126 Napoli, Italy (e-mail: alfredo@ventotene.dma.unina.it).

Digital Object Identifier 10.1109/TNN.2004.837585

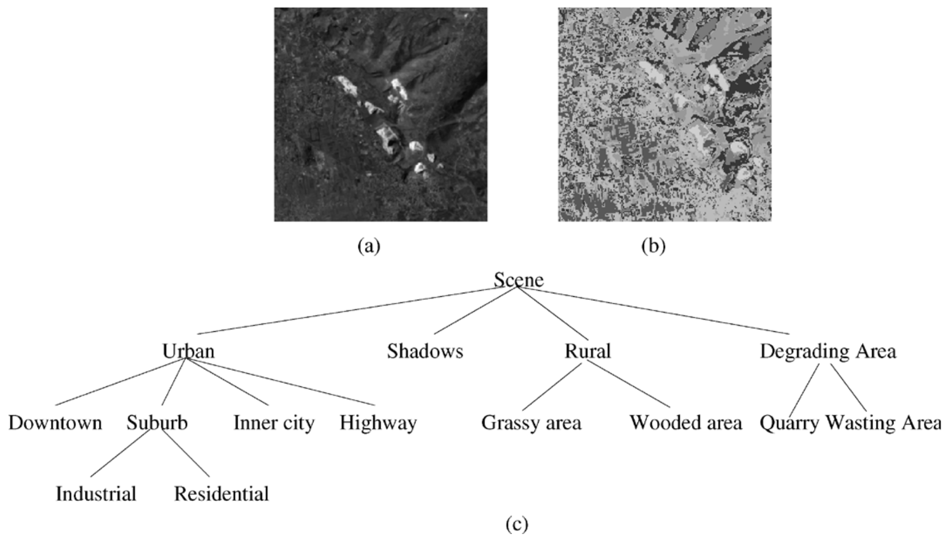

Fig. 2. Landsat TM image in band 5. (a) Clustering result of the bands 5-4-1 (b). Interpretation tree. (c) Fuzzy membership degree of each tree vertex to the associated class is attached, where each vertex corresponds to an ensemble of image pixels.

1) For illustration purposes, let us define a simplified version of a "house" to consist of an "isosceles triangle" and a "rectangle." Fig. 1 shows three "houses" and their corresponding tree structural representations based on the relation <consists of>, whose nodes (the "house" components) are described in terms of geometrical subsymbolic features like area, perimeter, but also in terms of a linguistic description such as a "small isosceles triangle," a "large rectangle," and so on. The linguistic description turns out to be a fuzzy membership degree of each "house" component to the concepts of "isosceles triangle" and "rectangle." Based on the fuzzy membership degrees attached to each tree vertex, the fuzzy membership degree of the entire tree to the class of "house" can be determined. For instance, the fuzzy membership degrees of the depicted figures may assume the values: 1) 0.8; 2) 0.6; and 3) 0.2, with the usual meaning that the higher the fuzzy value is, the greater the certainty of the belonging of the depicted figure to the class of "house" is.
2) Fig. 2 shows the application in remote sensed image interpretation. Multispectral Landsat TM images of an area of Southern Italy (near Naples) were analyzed using the Fuzzy C-Means clustering algorithm [1]. Eight clusters were found and the classification result of each pixel provides the basic pattern primitives and the related fuzzy membership degrees. A hierarchical tree model describing the spatial or contextual relationships between primitives can be constructed as shown in Fig. 2 (note that we drop details for sake of clarity). The spatial relations may be used to identify remote sensed images from, for instance, magnetic resonance images, to identify highway patterns or to improve the accuracy of the land-use classification in the Landsat image.

The embedding of the fuzzy-rule-based domain knowledge represented by a fuzzy tree automaton into connectionist models should provide two benefits: improving generalization to new instances and simplifying learning. The final goal is the tuning of network weights in order to revise and refine the prior (fuzzy) knowledge, both in terms of adaptation to newly presented patterns and of plasticity to better fit the network structure against the application domain structure. To highlight the inner structure in data, we focus on processing streams of structured data by recursive neural networks [10], where the temporal processing which takes place in recurrent neural networks is extended to the case of graphs by connectionist models [10]. Recursive neural networks can be initialized with prior knowledge about training structured data; this ability makes recursive neural networks useful tools for modeling fuzzy tree automata [38], where prior and fuzzy knowledge is available. The purpose of the present paper is to show how recursive neural networks can represent and model fuzzy tree automata either by using first-order connections or high-order ones. In particular, based on previous studies [28], [29], we prove that recursive neural networks with continuous sigmoidal functions can be constructed such that the encoded dynamics remains stable indefinitely. This study extends the results of [30], where a method for constructing fuzzy state automata in high-order recurrent networks is reported. The paper outline is as follows. We briefly review the principal results regarding fuzzy tree automata in Section II, giving specific importance to fuzzy frontier-to-root tree automata (FFRTA) which will be examined in the rest of the paper. Section III is dedicated to the formalism for recursive neural networks and the injection of FFRTA state transitions in first-order and high-order recursive networks which assign membership degrees to arbitrary trees accepted by an arbitrary FFRTA. In Section IV, we prove the stability of the encoding algorithm for the injection of FFRTA in high-order recursive networks. Some guidelines for further development are finally outlined in Section V.

II. Fuzzy Tree Automata

A. Basic Background

This section presents a brief introduction to tree grammars and automata. For a more detailed treatment of these topics we refer the reader to [13] and [19]. We choose, in the sequel, to indicate with the term vertex the node of both the input data structure and the state tree, while with the term node the neural network neuron.

Definition 1: A *ranked alphabet* is a pair (Σ, rank) where Σ is a finite set of symbols and rank the ranking function

$$\mathrm{rank}: \ \Sigma \to N^+ \cup \{0\} \tag{1}$$

where N^+ denotes the set of strictly positive integers. For all $a \in \Sigma$, $\mathrm{rank}(a)$ is called the rank of a.

Similarly, the rank can be defined as the relation $\mathrm{rank} \subseteq \Sigma \times N^+ \cup \{0\}$ such that $(a, \mathrm{rank}(a))$ is in $\Sigma \times N^+ \cup \{0\}$. In the sequel, T_Σ will denote the set of trees with vertices labeled by elements in Σ, also denoted as Σ-tree and Σ_i the set of symbols with rank i.

According to the previous definitions, any tree possesses vertices labeled with symbols in the ranked alphabet (Σ, rank), such that the ranking of a symbol a specifies the allowable number of children of a. If, as instance, $\mathrm{rank}(a)$ is in {2, 1, 0}, the symbol a may have no child or at most two children.

Definition 2: A *fuzzy tree grammar* $\mathcal{G}$ (FTG) is a 5-tuple

$$(V_N, V_T, \mathrm{rank}, P, S)$$

where: V_N, V_T are sets of nonterminals and terminals, respectively, rank is the ranking function; P is a finite set of fuzzy rewriting rules of the form $\mu(\gamma \to \psi) = \rho$ which are usually represented by $\gamma \xrightarrow{\rho} \psi$ with γ and ψ being trees with labels from the ranked alphabet $(V_N \cup V_T, \mathrm{rank})$ and μ a membership function with values in [0,1]; S is a finite subset of T_{V_N}.

A fuzzy tree grammar is in expansive form if its productions are of the form

$$\gamma \xrightarrow{\rho} a \qquad \text{or} \qquad \gamma \xrightarrow{\rho} \begin{array}{ccc} & A & \\ \swarrow & & \searrow \\ \xi_1 & \cdots & \xi_k \end{array}$$

where $\gamma \in V_N$, $\xi_i \in V_N \cup V_T$ $(i = 1, \ldots, k)$, $a, A \in V_T$. As shown in [20], for any regular FTG $\mathcal{G}_1$, i.e., a tree grammar where all that has been generated before the current symbol Φ is already known for a general production $\Phi \to \beta$, we can construct a FTG $\mathcal{G}_2$ with productions in expansive form; $\mathcal{G}_1$ and $\mathcal{G}_2$ are said to be equivalent. In the following, we mean with $\eta \overset{a}{\Rightarrow} \beta$, with β in $T_{V_N \cup V_T}$, the derivation of the tree β from the tree η by using productions $\Phi \xrightarrow{\rho} \psi$ in P such that Φ is a subtree of η rooted at a and β is obtained by replacing the occurrence of Φ at a by ψ with fuzziness degree ρ. In general, a derivation $\eta \overset{*}{\Rightarrow} \beta$ exists with β in $T_{V_N \cup V_T}$ if and only if there exist $\eta_0, \eta_1, \ldots, \eta_m$ $(m > 0)$ such that

$$\eta = \eta_0 \to \eta_1 \to \cdots \to \eta_m = \beta \tag{2}$$

and each production involved is in $\mathcal{G}$.

Definition 3: The *tree language* generated by $\mathcal{G}$ is defined as

$$L(\mathcal{G}) = \{\beta \in T_{V_T} \\ |\text{there exists } Y \in S \text{ such that } Y \overset{*}{\Rightarrow} \beta \text{ in } \mathcal{G}\}. \tag{3}$$

Since the tree symbols have graded memberships, it is useful to introduce the membership degree of an input tree:

Definition 4: Given a fuzzy tree grammar $\mathcal{G}$, the *membership degree of* a derived tree t in the language $L(\mathcal{G})$ is the maximum value of any derivation of t and the value of a specific derivation of t is equal to the minimum membership degree of the involved productions

$$\begin{aligned} &\mu_{\mathcal{G}}(t) \\ &= \mu_{\mathcal{G}}(S \overset{*}{\Rightarrow} t) \\ &= \max_{Y \overset{*}{\Rightarrow} t, Y \in S} \min[\mu_{\mathcal{G}}(Y \to \eta_1), \mu_{\mathcal{G}}(\eta_1 \to \eta_2), \ldots, \mu_{\mathcal{G}}(\eta_m \to t)]. \end{aligned} \tag{4}$$

The fuzzy tree automaton we refer to is the FFRTA, which processes input trees in a bottom-up manner (it reads trees from the leaves toward the root). The computation starting from the leaves can be inductively described as follows, by constructing at each step a "state tree" (a tree of the same shape of the input tree and labeled with states).

1) The starting state at a given leaf depends on the label γ $(\in \Sigma_0)$ of the leaf.
2) The state in which the automaton enters an inner vertex labeled with an $\mathrm{rank}(s)$-ary symbol s $(\mathrm{rank}(s) > 0)$, i.e., $s \in \Sigma_{\mathrm{rank}(s)}$, is determined by s and the states assigned at the $\mathrm{rank}(s)$ children vertices.

The input tree is accepted by the FFRTA A if the root of a Σ-tree t is reached in a final state.

A fuzzy tree automaton is just a fuzzy automaton [20] in which labeled trees replace strings as inputs and tree states replace usual states of a fuzzy finite-state automaton.

The previous description of the operation of a tree automaton suggests the following formal definition.

Definition 5: A *nondeterministic fuzzy frontier-to-root tree automaton* (NFFRTA) $\mathcal{A}$ over a ranked alphabet (Σ, rank) is a 5-tuple $(K, \Sigma, \mu, S_\epsilon, F)$, where

K	finite nonempty set of states;
(Σ, rank)	ranked alphabet of input symbols;
S_ϵ	initial fuzzy state;
$F \subseteq K$	set of final states which may be a fuzzy set over K.

The symbol μ denotes the fuzzy mapping $\mathcal{K} \times \Sigma \times K \to [0,1]$, $\mathcal{K}$ being a finite subset of K^+ (the set of subsets constructed over the set K, except for the empty set). $\mu(\{S_1, S_2, \ldots, S_k\}, a, S) = \rho$ means that when a vertex labeled with $a \in \Sigma_k$ has k children with associated states $S_1, \ldots, S_k$, the state S can be assigned to the vertex with membership degree ρ.

For sake of clearness, in the following we drop the parenthesis referring to as the first argument of the function $\mu()$. This is consistent with the definition mainly due to the use of a maximum rank tree automata as explained later. In addition, to clarify the exposition, in the following we shall use $\mu_a(S_1, S_2, \ldots, S_k, S)$ to denote $\mu(\{S_1, S_2, \ldots, S_k\}, a, S)$.

Definition 6: The *acceptance degree* $\mu_{\mathcal{A}}(t)$, of a Σ-tree t by a NFFRTA $\mathcal{A}$ is defined as

$$\begin{aligned} &\mu_{\mathcal{A}}(t) = \\ &\max_{S_j \in K} \{\min(\mu_{t_1}(S_{i_1}, \ldots, S_{i_r}, S_1), \ldots, \mu_{t_r}(S_{k_1}, \ldots, S_{k_r}, S_r)\} \end{aligned} \tag{5}$$

with $j = 1, \ldots, r$. If $t = a \in \Sigma_0$, then $\mu_t(\cdot, \ldots, \cdot) = \mu_a(\cdot, \ldots, \cdot)$.

According to the previous definition, the acceptance degree of Σ-tree t is equal to the maximum value of any derivation of t, where the value of a specific derivation of t is equal to the minimum weight of the productions used.

Without loss of generality, in the following we consider a restricted type of fuzzy tree automaton where the fuzziness degree is attached to all the automaton states (equivalently to the production rules of the tree language accepted by the automaton), except for the initial not fuzzy states; this automaton can be demonstrated to be equivalent to a fuzzy automaton [20].

Up to now, we have always mentioned NFFRTAs. At this point it is important to introduce the definition of deterministic frontier-to-root tree automata (DFFRTAs).

Definition 7: A DFFRTA $\tilde{\mathcal{A}}$ is a tree automaton with state transition function $\tilde{\mu}$ defined like that of NFFRTA with the condition that for each $S_i \in K$ and $a \in \Sigma_{\text{rank}(a)}$, if $\tilde{\mu}_a(S_{i_1}, S_{i_2}, \ldots, S_{i_{\text{rank}(a)}}, S_j) > 0$ and $\tilde{\mu}_a(S_{i_1}, S_{i_2}, \ldots, S_{i_{\text{rank}(a)}}, S_l) > 0, j \neq l$, then $S_j = S_l$.

The class of fuzzy languages accepted by DFFRTA is properly included in the family of fuzzy languages accepted by NFFRTA [25]. This consideration makes the use of DFFRTA unfavorable in practice and in the sequel we study NFFRTAs.

Usually, FFRTA are used primarily to characterize fuzzy tree languages in terms of their acceptance by such automata. As for state-finite automata, there exists a correspondence between NFFRTA and fuzzy tree grammars [6], [13]:

Proposition 8: For a given regular FTG $\mathcal{G}$, there exists a NFFRTA $\mathcal{A}$ (and vice versa) such that $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{A})$.

It is clear from the previous proposition that the grade of acceptance of a tree by the NFFRTA $\mathcal{A}$ gets the same value of the membership value $\mu_{\mathcal{G}}(t)$ (4) of the tree valued by the fuzzy grammar $\mathcal{G}$ which generates the language accepted by the NFFRTA (see [6] for more details).

Furthermore, let F be a designated set of final states, which may be a fuzzy subset of K. The language accepted by the NFFRTA can be thus defined as a fuzzy set of trees by

$$L(\mathcal{A}) = \{(t, \mu_{\mathcal{A}}(t)) \mid t \in T_\Sigma \text{ is accepted by } \mathcal{A} \text{ with acceptance degree } \mu_{\mathcal{A}}(t)\}. \quad (6)$$

In the following, an example about how a NFFRTA $\mathcal{A}$ is constructed from a fuzzy tree grammar $\mathcal{G}$ such that $L(\mathcal{A}) = L(\mathcal{G})$ is shown.

Example 1: Let $\mathcal{G} = (V_N, V_T, r, P, \lambda)$, where $V_N = \{\lambda, \xi\}$, $V_T = \{a, A\}$ and the set of fuzzy rewriting rules P is shown at the bottom of the page.

μ_a	S_λ	S_ξ	S_X
S_ϵ	0.7	0.6	1.0

μ_A	S_λ	S_ξ
S_X, S_λ	0.8	0
S_ξ, S_ξ	0.7	0
S_ξ, S_X	0	0.8

Fig. 3. Fuzzy rewriting rules for the NFFRTA A of Example 1.

$$\begin{tikzcd}
& & & A\,[S_\lambda, 0.7] & \\
& & A\,[S_\xi, 0.8] \arrow[ur] & & a\,[S_\xi, 0.6] \arrow[ul] \\
& A\,[S_\xi, 0.8] \arrow[ur] & & a\,[S_X, 1.0] \arrow[ul] & \\
a\,[S_\xi, 0.6] \arrow[ur] & & a\,[S_X, 1.0] \arrow[ul] & &
\end{tikzcd}$$

Fig. 4. Example of input tree t accepted by the NFFRTA $\mathcal{A}$ of Example 1 with acceptance degree $\mu_{\mathcal{A}}(t) = 0.6$, where the assigned states and the fuzzy degrees coming by the use of the state transitions are denoted as couples between square brackets.

Each fuzzy rewriting rule of the form

$$\begin{tikzcd}
& & A \arrow[dl] \arrow[dr] & \\
\gamma \arrow[r, "\rho"] & \xi_1 & \cdots & \xi_r
\end{tikzcd} \quad (7)$$

results in a state transition $\mu_A(S_{j_1}, \ldots, S_{j_r}, S_\gamma) = \rho$ where $S_{j_1}, \ldots, S_{j_r}$ correspond to symbols $\xi_1, \ldots, \xi_r$ and S_γ corresponds to γ, while the rules of the form $\phi \xrightarrow{\rho} a$, with $\phi = \lambda, \xi$ produce state transitions $\mu_a(S_\epsilon, S_\phi) = \rho$, with S_ϵ the initial state and S_ϕ corresponding to the symbol ϕ. The NFFRTA $\mathcal{A}$ derived by $\mathcal{G}$ is given by $(K, \Sigma, \mu, S_\epsilon, F)$, with $K = \{S_\lambda, S_\xi, S_\epsilon, S_X\}$, $\Sigma = \{a, A\}$, $F = \{S_\lambda\}$, and μ, as reported in Fig. 3. S_X is a state added to reflect the situation when in (7) one of ξ_i, $i = 1, \ldots, r$ is a terminal symbol; this corresponds to the introduction of the fuzzy transition $\mu_a(S_\epsilon, S_X) = \max(\rho)$, where $\max(\rho)$ is the maximum value for ρ. Without loss of generality, we consider that $\max(\rho) = 1.0$ (see Fig. 3). An input tree processed by the NFFRTA $\mathcal{A}$ and accepted with acceptance degree 0.6 is depicted in Fig. 4.

$$\begin{tikzcd}
& & A \arrow[dl] \arrow[dr] & \\
\lambda \arrow[r, "0.8"] & a & & \lambda
\end{tikzcd}
\qquad
\lambda \xrightarrow{0.7} a
\qquad
\begin{tikzcd}
& & A \arrow[dl] \arrow[dr] & \\
\lambda \arrow[r, "0.7"] & \xi & & \xi
\end{tikzcd}$$

$$\begin{tikzcd}
& & A \arrow[dl] \arrow[dr] & \\
\xi \arrow[r, "0.8"] & \xi & & a
\end{tikzcd}
\qquad
\xi \xrightarrow{0.6} a.$$

Input: NFFRTA A.

Output: DFRTA $\tilde{A}$ which accepts tree with the same acceptance degree than A.

Algorithm:

1. Derive a fuzzy tree grammar G which generates the trees accepted by the NFFRTA A using standard techniques [13].
2. Set $MD = \{\rho | \rho > 0 \text{ is a membership degree in } \mathcal{A}\}$
3. For each distinct $\rho \in MD$ compute the set of trees:

$$\mathcal{L}(\mathcal{G}, \rho) = \{t | t \in T_\Sigma, \mu_\mathcal{G}(t) = \rho\}$$

4. For each distinct $\mathcal{L}(\mathcal{G}, \rho)$ derive a DFRTA. The union defines a nondeterministic DFRTA.
5. Tranform the nondeterministic DFRTA into an equivalent DFRTA using standard algorithms [19].

Fig. 5. Algorithm for the transformation of a NFFRTA into an equivalent DFRTA.

For the sake of clearness, we consider maximum rank NFFRTA. Let $\text{rank}(s)$ be the rank of the symbols $s \in V_T \cup V_N$ and $r = \max_{s \in V_T \cup V_N} \text{rank}(s)$. Any NFFRTA $\mathcal{A} = (K, \Sigma, \mu, S_\epsilon, F)$ can be transformed in polynomial time to a maximum rank r NFFRTA $\mathcal{A}' = (K', \Sigma, \mu', S_\epsilon, F)$, where $K' = K \cup \{S_0\}$, S_0 being the nil state. Each transition in $\mathcal{A}$ denoted by $\mu : \mathcal{K} \times \Sigma \times K \rightarrow [0,1]$ is extended to $\mu' : (K')^r \times \Sigma \times K \rightarrow [0,1]$, such that for $s \in \Sigma$ and $S_{j_p} \in K$, $p = 1, \ldots, \text{rank}(s)$, $\mu_s(S_{j_1}, \ldots, S_{j_{\text{rank}(s)}}, S_i) = \rho$ becomes $\mu'_s(S_{j_1}, \ldots, S_{j_{\text{rank}(s)}}, \overbrace{S_0, \ldots, S_0}^{r-\text{rank}(s)}, S_i) = \rho$ with $r - \text{rank}(s)$ nil states S_0.

The following section outlines how an NFFRTA can be transformed into a deterministic frontier-to-root automaton (DFRTA) [13] which computes the same grade acceptance. The DFRTA is defined equivalently to the NFFRTA except that the state transition has membership degree $\rho = 1.0$. To highlight this difference, in the following the DFRTA state transitions will be denoted with function δ and the function $\mu_a()$ will be replaced by $\delta_a()$.

B. NFFRTA-to-DFRTA Transformation

Based on the work of Mizumoto *et al.* [27], a fuzzy language $\mathcal{L}(\mathcal{G})$ can be characterized in terms of nonfuzzy languages, i.e.,

$$\mathcal{L}(\mathcal{G}, \rho) = \{t \in T_{V_T} | \mu_\mathcal{G}(t) = \rho\}.$$

This procedure yields disjoint languages $\mathcal{L}(\mathcal{G}, \rho_l)$, each one being the equivalence class of trees whose membership acceptance grade is ρ_l. Each language $\mathcal{L}(\mathcal{G}, \rho_l)$ is accepted by a nondeterministic finite tree automaton, which can be converted into a deterministic finite tree automaton DFRTA which computes the grade of acceptance ρ at the end of all the state transitions leading to an accepting state [19]. This state is characterized by the membership value ρ which is the grade of acceptance of the equivalence class of trees in $\mathcal{L}(\mathcal{G}, \rho)$. The algorithm for the transformation of a NFFRTA into DFRTA is shown in Fig. 5.

Example 2: Let $\mathcal{G}$ and $\mathcal{A}$ be, respectively, the fuzzy tree grammar and the NFFRTA reported in Example 1. The language generated by $\mathcal{G}$ (and accepted by $\mathcal{A}$) can be described in terms of cut-point languages $\mathcal{L}(\mathcal{G}, \rho)$. In particular

$$L(\mathcal{G}) = \mathcal{L}(\mathcal{G}, 0.6) \cup \mathcal{L}(\mathcal{G}, 0.7) \tag{8}$$

where (9)–(10) are shown at the bottom of the page.

Here a tree has been described in terms of the pseudo-term representation as mentioned in the previous section and n stands for the number of the repetitions of the symbols within the parentheses. Following the above described method for constructing a DFRTA from a fuzzy language, the DFRTA which accepts the language $\mathcal{L}(\mathcal{G}, 0.6)$ is $\mathcal{A}_{0.6} = (\{S_1, S_2, S_3, S_4\}, \Sigma, \delta, S_1, \{S_4\})$, while the DFRTA accepting the language $\mathcal{L}(\mathcal{G}, 0.7)$ is $\mathcal{A}_{0.7} =$

$$\begin{aligned}\mathcal{L}(\mathcal{G}, 0.6) = &\{(t, 0.6) | t = \overbrace{A(a,}^{k} \overbrace{A(\cdots A(}^{n} A(a,a), \overbrace{a) \cdots a)}^{n} \overbrace{\ )\ }^{k}, n \geq 1, k \geq 0\} \cup \\ &\{(t, 0.6) | t = \overbrace{A(a,}^{k} A(\overbrace{A(\cdots A(}^{n} A(aa) \overbrace{a) \cdots a)}^{n} \overbrace{A(\cdots A(}^{m} A(aa) \overbrace{a) \cdots a)}^{m}) \overbrace{\ )\ }^{k}, n, m \geq 0, k \geq 0\}\end{aligned} \tag{9}$$

$$\mathcal{L}(\mathcal{G}, 0.7) = \{(t, 0.7) | t = \overbrace{A(a, \ldots, A(a,}^{n} a \overbrace{) \cdots)}^{n}, n \geq 0\}. \tag{10}$$

DFRTA $\mathcal{A}_{0.6}$:

δ_a	S_1	S_2	S_3	S_4
S_1	0	1	0	0

δ_A	S_1	S_2	S_3	S_4
S_2, S_2	0	0	1	0
S_3, S_2	0	0	0	1
S_4, S_2	0	0	0	1
S_3, S_3	0	0	0	1
S_3, S_4	0	0	0	1
S_4, S_3	0	0	0	1
S_4, S_4	0	0	0	1
S_2, S_4	0	0	0	1

(a)

DFRTA $\mathcal{A}_{0.7}$:

δ_a	S_1	S_2	S_3
S_1	0	1	0

δ_A	S_1	S_2	S_3
S_2, S_2	0	0	1
S_2, S_3	0	0	1

(b)

Fig. 6. Transition functions of the DFRTAs $\mathcal{A}_{0.6}$ and $\mathcal{A}_{0.7}$, which accept the languages $\mathcal{L}(\mathcal{G}, 0.6)$ and $\mathcal{L}(\mathcal{G}, 0.7)$ of Example 2.

$(\{S_1, S_2, S_3\}, \Sigma, \delta, S_1, \{S_2, S_3\})$. The corresponding state transitions δ are reported in Fig. 6(a) and (b).

Finally, the DFRTA which emulates the behavior of the NF-FRTA with the same acceptance degree is

$$\mathcal{A} = (\{S_1, S_2, S_3, S_4\}, \Sigma, \delta, S_1, \{S_2, S_3, S_4\})$$

where the final states S_2, S_3 correspond to the acceptance degree $\mu_{S_2} = \mu_{S_3} = 0.7$, while the final state S_4 corresponds to the acceptance degree $\mu_{S_4} = 0.6$ (see Fig. 7 for the values of the transition function).

III. Encoding of Fuzzy Tree Automata in First and High Order Recursive Neural Networks

A. *Recursive Neural Networks: Background Topics*

Recursive neural networks are characterized not to possess explicit feedback connections; the recurrent processing is then driven by the inherent recursive structure of the patterns in the training domain. Consider that instances in the training domain are structured patterns of information described by ordered r-ary trees. According to the mathematical induction, an r-ary tree is either 1) empty, or 2) an ordered $r+1$ tuple $t(t_1, \ldots, t_r)$ where the root t is a vertex and t_i are r-ary trees, $i = 1, \ldots, r$. Here by an ordered r-ary tree we mean an r-ary tree where for each vertex a total order on the edges leaving from it is defined. A total order can be induced on the tree nodes by topologically sorting the nodes with a linear order $<$, such that $v < w$ for any existing edge (v, w). As shown in Fig. 8, two different such orderings for the depicted tree are shown. The nodes are processed following their topological order, which

δ_a	S_1	S_2	S_3	S_4
S_1	0	1	0	0

δ_A	S_1	S_2	S_3	S_4
S_2, S_2	0	0	1	0
S_3, S_2	0	0	0	1
S_4, S_2	0	0	0	1
S_3, S_3	0	0	0	1
S_3, S_4	0	0	0	1
S_4, S_3	0	0	0	1
S_4, S_4	0	0	0	1
S_2, S_3	0	0	1	0
S_2, S_4	0	0	0	1

Fig. 7. Transition function of the DFRTA after the transformation of the NFFRTA of Example 1.

also guarantees a cycle-free graph. This guarantees that the state of each child of the node is already available when the state transition function is computed. Informally speaking, the encoding of a given tree into a distributed network is achieved by recursively setting the previously computed representation for the direct subtrees together with the representation of the label attached to the root. By starting this process from the leaves to the tree root and following their topological order, a representation of the whole tree is achieved and can be mapped to a particular user-specified output (structured or not).

To define the dynamics of the network, let us introduce the generalized shift operator as defined in [10]. The usual shift operator q^{-l} applied to $x(t)$ returns the variable x at time $t-l$, i.e., $q^{-l}x(t) = x(t-l)$, with $l \geq 1$. These operators are characterized by a composition rule so that $q^{-l} = q^{-1}q^{1-l}$, where $q^0 = 1$ is the neutral operator defined as $q^0 x(t) = x(t)$. More generally, an ordered set of generalized shift operators can be defined when a high-dimensional data structure (for instance r-ary trees) is considered. For $k = 1, \ldots, r$, the generalized shift operator is denoted by q_k^{-1} and is associated to the kth child of a given vertex. In such a case, the composition rule is also valid, but the commutative rule is no longer valid, whereas an inverse shift operator q_k for r-ary trees can be defined as $q_k q_k^{-1} x_v = x_v$, $k = 1, \ldots, r$, with x_v denoting the state variable associated to vertex v.

From now on, uniformly labeled r-ary trees will be denoted by the boldface uppercase letters corresponding to the label space of the tree; for instance, $\mathbf{Y}$ denotes a tree with labels in $\mathcal{Y}$. Labels are accessed by vertex subscripts, i.e., $\mathbf{Y}_v$ denotes the label attached to vertex v. Given a tree structure $\mathbf{Y}$, the tree obtained by ignoring all node labels will be referred to as the skeleton of $\mathbf{Y}$, denoted by $\mathrm{skel}(\mathbf{Y})$. Two structures $\mathbf{Y}$ and $\mathbf{Z}$ can be distinguished because they have different skeletons, i.e., $\mathrm{skel}(\mathbf{Y}) \neq \mathrm{skel}(\mathbf{Z})$, or they have the same skeleton but different vertex labels. The class of trees defined over the local universe domain $\mathcal{Y}$ and skeleton in $\#^{(1,o)}$, i.e., the set of graphs whose vertices have in-degree 1 and maximum out-degree o, will be denoted as $\mathcal{Y}^{\#^{(1,o)}}$. In the following, the training set

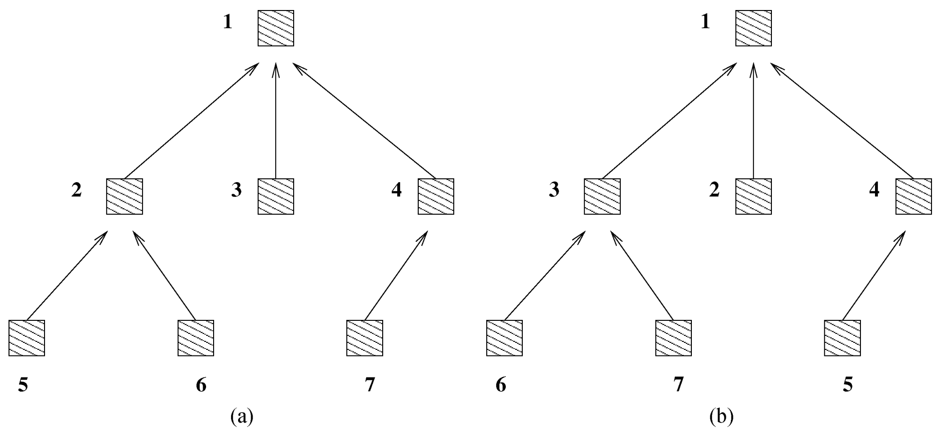

Fig. 8. Two different total orderings by topological sorting the tree nodes.

comprises couples $(\mathbf{U}, \mathbf{Y})$, where $\mathbf{U} \in \mathcal{U}^{\#^{(1,o)}}$ are input trees with symbols s_k in Σ, each converted into a vector $\mathbf{U}_k$ in $\mathcal{R}^n$, attached to nodes; the same is true for $\mathbf{Y} = \tau(\mathbf{U}) \in \mathcal{Y}^{\#^{(1,o)}}$ which represent output trees. Since $\tau(\cdot)$ is a binary transduction between input and output trees, $\tau \subseteq \mathcal{U}^{\#^{(1,o)}} \times \mathcal{Y}^{\#^{(1,o)}}$ the aim of any supervised learning algorithm is to estimate the transduction $\tau(\cdot)$. In particular, we shall confine ourselves to transductions from $\mathbf{U} \in \mathcal{U}^{\#^{(1,o)}}$ to $\mathbf{Y} \in \mathcal{Y}^{\#^{(1,o)}}$, which are:

1) IO-isomorph, i.e., $\text{skel}(\tau(\mathbf{U})) = \text{skel}(\mathbf{U})$;
2) causal, i.e., $\tau(\mathbf{U})_v$ only depends on the subtree of $\mathbf{U}$ induced by v and its descendants, $\forall v$.

Such an IO-isomorph transduction $\tau(\cdot)$ admits a recursive state representation if there exists a structure space $\mathcal{X}^{\#^{(1,o)}}$ such that for each $\mathbf{U}, \mathbf{Y}$ there exists $\mathbf{X} \in \mathcal{X}^{\#^{(1,o)}}$ with $\text{skel}(\mathbf{X}) = \text{skel}(\mathbf{U}) = \text{skel}(\mathbf{Y})$ and for each vertex $v \in \text{skel}(\mathbf{U})$

$$\mathbf{X}_v = f(\mathbf{X}_{\text{ch}[v]}, \mathbf{U}_v) \tag{11}$$

$$\mathbf{Y}_v = g(\mathbf{X}_v, \mathbf{U}_v) \tag{12}$$

where $\mathbf{X}_{\text{ch}[v]}$ is a fixed size array of labels attached to the (ordered) children of v and

$$f : \mathcal{X}^o \times \mathcal{U} \to \mathcal{X} \tag{13}$$

$$g : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}. \tag{14}$$

According to the above definition, f will be referred to as the *state transition function* and g is referred to as the *output function*. We shall confine ourselves to stationary trasductions, i.e., $f(\cdot)$ and $g(\cdot)$ do not depend on v. Furthermore, another restriction comes from the computations involved. Specifically, any automaton constructed to accept o-ary trees proceeds its computation from the external vertices of the o-ary tree toward the root, and a tree is accepted if an accepting state is reached, by realizing the deterministic causal IO-isomorph transduction $\tau(\cdot)$. In particular, for maximum rank r NFFRTA transformed into DFRTA, the state transition function and the output function look like

$$\mathbf{X}_v = \delta(q_1^{-1}\mathbf{X}_v, \ldots, q_r^{-1}\mathbf{X}_v, \mathbf{U}_v) \tag{15}$$

$$\mathbf{Y}_t = \begin{cases} \mu_{\mathbf{X}_t}, & \text{if } \mathbf{X}_t \in F \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

where $\mathbf{X}_t$ is the state reached at the input tree root and F the set of accepting final states. Note that the state transition explicated in (15) corresponds to the deterministic state transitions $\delta_{\mathbf{U}_v}(q_1^{-1}\mathbf{X}_v, \ldots, q_r^{-1}\mathbf{X}_v)$.

According to the total order induced on the tree nodes, the states are updated following a recursive message passing scheme such that a state label $\mathbf{X}_v$ is updated after the state labels corresponding to the children of v in the order defined, as instance, by any reversed topological sort of the tree nodes.

B. Encoding in First-Order Recursive Networks

The implementation of DFRTAs in recursive neural networks follows the construction lemmas reported in [23] and [36]. Let $\mathcal{A} = (K, \Sigma, \delta, S_\epsilon, F)$ be an arbitrary DFRTA with an r-ranked symbol alphabet Σ, m states, n input symbols and t be a given tree and v a vertex in t whose childrens have already been assigned the states $(S_1, \ldots, S_r)$. $\mathcal{A}$ assigns the state S to v iff

$$\bigvee_{\substack{\delta_a(St_1, \ldots, St_r, S) \\ a \in \Sigma}} ((a = \text{label}(v)) \wedge (\wedge_{i=1}^{r}(St_i = S_i)))$$

where $\vee$ and $\wedge$, respectively, denote the OR and AND operations, $\text{label}(v)$ indicates the label of the node v and St_i the generic ith state.

The recursive network is characterized by the dependence of a vertex v from its r children $\text{ch}[v]$, which can be expressed by means of the pointer matrices $W_k \in \mathcal{R}^{m,m}$, $k = 1, \ldots, r$. Likewise, the information represented by the symbols in Σ, converted into vectors $\mathbf{U}_v \in \mathcal{R}^n$ and attached to the network nodes, can be propagated by the weight matrix $\widetilde{W} \in \mathcal{R}^{m,n}$. Hence, each neuron in the first-order network computes the sigmoidal transfer function, which is demostrated to not going to cumulate approximation errors (see for instance [8] and [15])

$$\mathbf{X}_v = \vec{\sigma}\left(\sum_{k=1}^{o} W_k \cdot q_k^{-1}\mathbf{X}_v + \widetilde{W} \cdot \mathbf{U}_v + \boldsymbol{\theta}\right) \tag{17}$$

where $\vec{\sigma}(\cdot)$ means a vector of sigmoidal functions, i.e., $\vec{\sigma}(x_1, \ldots, x_n) = (\sigma(x_1), \ldots, \sigma(x_n))$. The weight matrices W_k and $\widetilde{W}$ are the same for every node v, i.e., the transduction $\tau(\cdot)$ is stationary, and $\boldsymbol{\theta} \in \mathcal{R}^m$ is the bias term.

These recursive state neurons are used to construct the DFRTA state transitions by implementing boolean-like AND and OR functions by constraining the incoming weights. In [36],

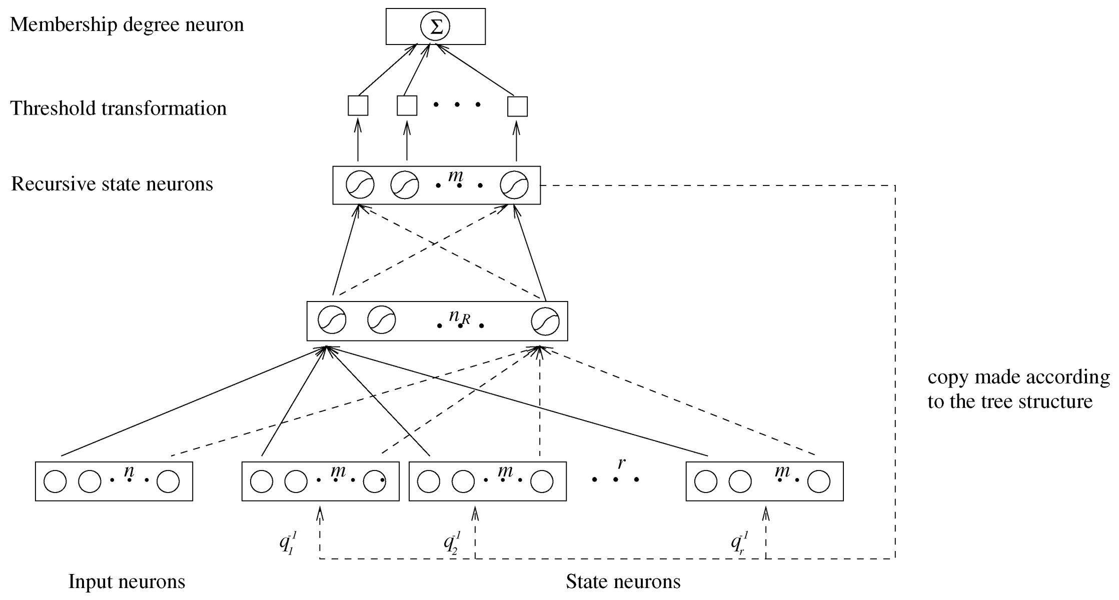

Fig. 9. First-order recursive neural network associated with an NFFRTA.

it is proved that one layer of first-order recursive neurons and threshold transfer functions has the power to simulate any deterministic bottom-up tree automaton. The construction scheme follows the idea explained above even if it does not explicitly identify neurons as Boolean gates. There, the case of sigmoid functions is afforded by using the approximation of threshold functions by sigmoid ones. In [23], it is proposed a neural model for processing structured data, i.e., the neural folding architecture (FA), and shown that the FA with first-order connections and the general transfer function σ_g (i.e., $\sigma_g : \mathcal{R} \rightarrow]\mu^-, \mu^+[$ and σ_g continuous) can simulate any DFRTA using two layers in the folding part and only one unit in the transformation part. The classical sigmoid function is the case when $\mu^- = 0$ and $\mu^+ = 1$. With the meaning that the folding part corresponds to hidden layers and the transformation part the output layer, the same results reported in [23] can be applied to our case. This proposed network construction algorithm can implement any DFRTA with m states and n input symbols using a network with n_R units which act as logical $\wedge$ on the first layer, n_R being the number of explicitly specified transitions, m units which act as logical $\vee$ on the second layer, each ith unit with arity $n_i = \#\{\delta_a(S_1, \ldots, S_r, S) \in R\}$ where R is the set of rules, for a total of no more than $(r+1)m + n + nm^r$ continuous neurons and no more than $O(nm^r(r+m+1)+m)$ weights. Similar results can be obtained for sigmoid units also following the study introduced in [34]. According to Sima's theorem, for any tolerance δ such that $0 < \delta < \delta_{\max}$ and for any output tolerance ϵ such that $0 < \epsilon \leq \delta$, there exists an analog neuron with activation function σ_g and weights $w_0, \ldots, w_n \in \mathcal{R}$ such that the threshold transfer function σ_H with weights $W_0, \ldots, W_n$ can be realized for all inputs $x = (x_1, \ldots, x_n)$ as

$$\sigma_H\left(W_0 + \sum_{j=1}^{n} W_j x_j\right) = \tau_\epsilon\left(\sigma_g\left(w_0 + \sum_{j=1}^{n} w_j r_\delta(x_j)\right)\right)$$

with $\tau_\epsilon(x) = 0$ if $x \in (\mu^- - \epsilon, \mu^- + \epsilon)$, $\tau_\epsilon(x) = 1$ if $x \in (\mu^+ - \epsilon, \mu^+ + \epsilon)$ and undefined otherwise, and r_δ the inverse mapping $r_\delta(k) = \{x \in \mathcal{R} | \tau_\delta(x) = k\} \in \{0,1\}$. Based on Sima's theorem, the conditions for a stable simulation when the weight values are chosen as $w_i = HW_i$ are shown to be [4]

$$H > \frac{\max\{-\sigma_g^{-1}(\mu^- + \epsilon), \sigma_g^{-1}(\mu^+ - \epsilon)\}}{\min_{i=1,\ldots,n} |W_0 + W_i| - n\delta W}$$
$$\delta < \min_{i=1,\ldots,n} \frac{|W_0 + W_i|}{nW}$$

with $W = \max_{j=1,\ldots,n}\{|W_j|\}$. Specifically, in [4], it is shown that by adopting the encoding of a deterministic tree automaton in a first-order recursive network (see [36]), the conditions hold and in particular

$$H > \frac{2\sigma_g^{-1}(1-\epsilon)}{1 - 2rm\epsilon}$$

that together with $\delta < 1/2rm$ leads to a number of weights which grows slower than $\log(rm)$.

The addition of a third layer consisting of a single linear neuron allows the computation of the fuzzy membership of any given string. The weights attached to each connection between the second layer and the output neuron are just the membership functions associated to the final states of the DFRTA after the transformation of the NFFRTA. Let μ_v denote this fuzzy membership of state S_v, if S_v is a final state, 0 otherwise, then the output neuron computes the following function:

$$\mu(s) = \sum_v \mu_v \sigma_{\text{th}}(X_v) \tag{18}$$

where the sum is over final state variables X_v and σth is the threshold function with a suitable (usually > 0.5) threshold th, i.e., $\sigma_{\text{th}}(x) = 1$ if $x \geq th$ and 0 otherwise. Fig. 9 is a pictorial representation of the computation involved in the first-order recursive neural network. The following example shows how a first-order recursive neural network for the NFFRTA of Example 1 is constructed.

Example 3: The recursive neural network for the NNFRTA, shown in Example 1 with fuzzy rewriting rules reported in Fig. 3

and emulated by the DFFRTA with deterministic transition rules reported in Fig. 7, possesses the structure depicted in Fig. 9.

1) Since the adopted symbols are two, i.e., a and A, the number of inputs is $n \geq 1$ if the logaritmic representation of the inputs is adopted, otherwise $n = 2$ if the 1-of-2 coding is used.
2) The states are $\{S_1, S_2, S_3, S_4\}$ and the value of m is 4 (1-of-4 coding).
3) The arity of each state is $r = 2$.
4) The number of transition rules is $n_R = 9$, corresponding to $\delta_a(S_1, S_1)$, $\delta_A(S_2, S_2)$, $\delta_A(S_3, S_2)$, $\delta_A(S_4, S_2)$, $\delta_A(S_3, S_3)$, $\delta_A(S_3, S_4)$, $\delta_A(S_4, S_4)$, $\delta_A(S_2, S_3)$, and $\delta_A(S_2, S_4)$.
5) The weights between the layer of recursive state neurons and the threshold transformation layer have values 0, except for those related to the states S_3 with value 0.7 and the state S_4 with value 0.6.

Based on the constructive procedure reported before, the following theorem holds

Theorem 1: The recursive neural network with first-order connections and stable dynamics can simulate any NFFRTA using two layers of neurons with sigmoid transfer function, a binary thresholded function based transformation part and one layer of linear neurons.

C. Encoding in $(r+1)$-Order Recursive Networks

High-order neural networks, proposed by Giles *et al.* for both static networks [16] and recurrent networks [14], are very interesting models for dealing with symbolic tasks.

In this section, the extension of recursive neurons with high-order connections is reported. We consider that each neuron possesses r recursive connections (the valence of the domain is r), even if not all of them will be used for computing the output of a node x with out-degree less than r. The output of the recursive neuron is defined as

$$\begin{aligned} X_{i,v} &= \sigma(\alpha_{i,v}) = \frac{1}{1+e^{-\alpha_{i,v}}} \\ \alpha_{i,v} &= \theta_i + \sum_{j_1} \cdots \sum_{j_r} \sum_{k} W_{i,j_1,\ldots,j_r,k} \prod_{p=1}^{r} \left(q_p^{-1}\mathbf{X}_v\right)_{j_p} U_{k,v} \end{aligned} \tag{19}$$

where the notation is the same adopted in the previous section, except for the weights $W_{i,j_1,\ldots,j_r,k}$ of the recursive neurons. The product $\prod_{p=1}^{r}\left(q_p^{-1}\mathbf{X}_v\right)_{j_p} U_{k,v}$ in the equation directly corresponds to the state transition $\delta_s(S_{j_1}, \ldots, S_{j_r}) = S_i$, where the state S_{j_k} has been encoded as $q_k^{-1}\mathbf{X}_v$, the state S_i as $\mathbf{X}_v$ and the symbol s as $\mathbf{U}_v$. The goal is to achieve a nearly orthonormal internal representation of the DFRTA states with the desired network dynamics. Some output neurons S_f are demanded to accept or reject the input tree.

Due to the recursive nature of the network dynamics, the stability is guaranteed only if the function $\sigma(\cdot)$ asimptotically converges toward its fixed points in the saturation regions. Let us consider that the input is $U_{s,v} = \delta_{s,v}$ where δ denotes here the Kronecker delta function, i.e., each input symbol is assigned to an input neuron. According to the encoding algorithm, the state vector $[X_{1,v}, X_{2,v}, \ldots, X_{m,v}]$, is characterized by a one-hot encoding, i.e., the current state $X_{j,v}$ produced by a DFRTA transition sends a high signal to its connected neurons, while the remaining states send low signals.

Before proceeding, in the sequel for sake of clearness we shall adopt the following notation:

$$X^p_{j_p,v} \equiv \left(q_p^{-1}\mathbf{X}_v\right)_{j_p}$$

and, where clear from the context

$$X^p_{j_p,v} \equiv X_{jp,v}.$$

Given a DFRTA with m states and n input symbols, a network with m $(r+1)$-order recursive neurons and n input neurons is built. The algorithm 1) programs the weights of a network in order to reflect the state transitions of the DFRTA and 2) programs the output of the response neuron for each DFRTA state. By considering that the symbol s_k is coded as $\mathbf{U}_k$, the network weights and biases are programmed with strength H as follows:

$$W_{0,j_1,\ldots,j_r,k} = \begin{cases} H, & \text{if } \delta_{s_k}(S_{j_1}, \ldots, S_{j_r}) \in F \\ -H, & \text{otherwise} \end{cases} \tag{20}$$

$$W_{z,j_1,\ldots,j_r,k} = \begin{cases} H, & \text{if } \delta_{s_k}(S_{j_1}, \ldots, S_{j_r}) = S_z \\ -H, & \text{otherwise} \end{cases} \quad z \in \{j_1, \ldots, j_r\} \tag{21}$$

$$W_{i,j_1,\ldots,j_r,k} = \begin{cases} H, & \text{if } \delta_{s_k}(S_{j_1}, \ldots, S_{j_r}) = S_i \\ 0, & \text{otherwise} \end{cases} \quad i \notin \{j_1, \ldots, j_r\} \tag{22}$$

$\theta_{i,v} = -H/2$ for all neurons $X_{i,v}$. The weights have the following meaning.

1) The weigh,t expressed in (20), $W_{0,j_1,\ldots,j_r,k}$ corresponds to the weight of the link between the neuron computing the product between $X_{j_{1,v}}, \ldots, X_{j_{r,v}}$ and the input $U_{k,v}$ and the neuron related to a final state S_0. It is responsable of the encoding of the transition $\delta_{s_k}(S_{j_1}, \ldots, S_{j_r}) \in F$, i.e., it operates as "classifier" by recognizing whether the new state is a final state or not.
2) The weight, expressed in (21), $W_{z,j_1,\ldots,j_r,k}$ corresponds to the weight of the link between the neuron computing the product between $X_{j_{1,v}}, \ldots, X_{j_{r,v}}$ and the input $U_{k,v}$ and the neuron related to a state among $\{S_{j_1}, \ldots, S_{j_r}\}$. It is responsable of the encoding of the transition $\delta_{s_k}(S_{j_1}, \ldots, S_{j_r}) = S_z$, with $z \in \{j_1, \ldots, j_r\}$.
3) The weight, expressed in (22), $W_{i,j_1,\ldots,j_r,k}$ corresponds to the weight of the link between the neuron computing the product between $X_{j_{1,v}}, \ldots, X_{j_{r,v}}$ and the input $U_{k,v}$ and the neuron related to the state S_i $(i \neq j_1, \ldots, j_r)$. It is responsable of the encoding of the transition $\delta_{s_k}(S_{j_1}, \ldots, S_{j_r}) = S_i$ $i \notin \{j_1, \ldots, j_r\}$.

As shown in Fig. 10, the encoding of the transition

$$\begin{tikzcd} & s_k & & & \\ S_{j_1} \arrow[ur] & \cdots & S_{j_r} \arrow[ul] & \arrow[r, "\rho"] & S_i \end{tikzcd} \tag{23}$$

is depicted. The network state vector

$$[X_{1,v}, X_{2,v}, \ldots, X_{m,v}]$$

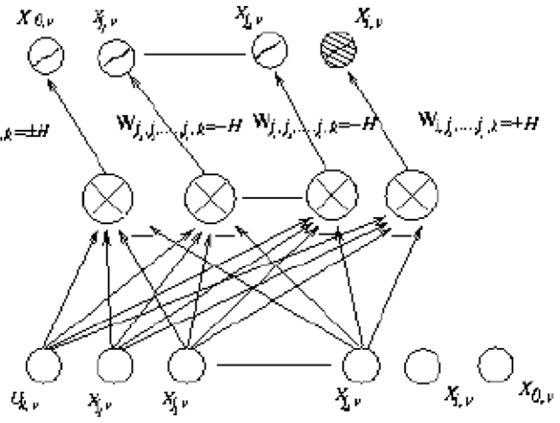
$X_{0,v}$
$X_{j_1,v}$
$X_{j_r,v}$
$X_{i,v}$
$W_{0,j_1,\ldots,j_r,k}=\pm H$
$W_{j_1,j_1,\ldots,j_r,k}=-H$
$W_{j_r,j_1,\ldots,j_r,k}=-H$
$W_{i,j_1,\ldots,j_r,k}=+H$
$U_{k,v}$
$X_{j_1,v}$
$X_{j_2,v}$
$X_{j_r,v}$
$X_{i,v}$
$X_{0,v}$

Fig. 10. Encoding of the transition state rule $\delta_{s_k}(S_{j_1},\ldots,S_{j_r}) = S_i$ into an r-ary recursive network. $\times$ means the operation $\prod_{p=1}^{r} X_{j_p,v} \cdot U_{k,v}$. The value of the weight $W_{0,j_1,\ldots,j_r,k}$ depends if the state S_i is an accepting state or not.

is initialized such that $X_{i,v}$, $1 \le i \le m$, is 1 if $X_{i,v}$ is an initial state, 0 otherwise. Given an input tree t, the network accepts t if the value of one final state output neuron is greater than 0.5, in any other case the network rejects the input tree. Fig. 11 is a pictorial representation of the computation involved in an high-order recursive neural network.

Example 4: Following Example 3, the high-order recursive neural network for the NNFRTA shown in Example 1, emulated by the DFFRTA reported in Fig. 7 possesses the structure depicted in Fig. 11, where the values of m, n_R are the same of those reported in Example 3, while the number of inputs is $n = 2$ (1-of-2 coding). Recall from Example 3, that the $n_R = 8$ transition rules are $\delta_a(S_1, S_1)$, $\delta_A(S_2, S_2)$, $\delta_A(S_3, S_2)$, $\delta_A(S_4, S_2)$, $\delta_A(S_3, S_3)$, $\delta_A(S_3, S_4)$, $\delta_A(S_4, S_4)$, $\delta_A(S_2, S_3)$, and $\delta_A(S_2, S_4)$.

Each transition is programmed according to [(20)–(22)] by weights $W_{i,j_1,\ldots,j_r,k}$ feeding from the state neurons $X_{j_{1,v}}, \ldots, X_{j_{r,v}}$ and input neuron $U_{k,v}$ to state neuron $X_{i,v}$. Let us indicate the symbols a and A, respectively, with input neuron 1 and 2. Neurons $0, 1, \ldots, 4$ represent the network output, state S_1, so on , up to state S_4.

As instance, the transition $\delta_A(S_2, S_2)$ is programmed with weight values (shown in the table at the bottom of the page) while the transition $\delta_A(S_4, S_2)$ is programmed considering that S_4 is an accepting state, with weight values (shown in the table at the bottom of the page).

Following the same considerations made for first-order networks, the following theorem holds.

Theorem 2: The recursive neural network with high-order connections programmed as in (20)–(22) and stable dynamics can simulate any NFFRTA using one layer of production neurons, one layer of neurons with a sigmoid transfer function, a binary thresholded function based transformation part and one layer with one linear neuron as in (18).

The following section is part of the proof of the previous theorem, showing the conditions under which a recursive neural network with high-order connections programmed as in (20)–(22) possesses stable dynamics.

IV. High-Order Network Stability Analysis

To demonstrate the stability of encoding in high-order networks, we follow and generalize the arguments treated in [28]. Keeping in mind that for each deterministic automaton state transition one programmed weight $W_{i,j_1,j_2,\ldots,j_r,k}$ is involved, the dynamics of the network can be described as

$$X_{i,v} = \frac{1}{1 + e^{H/2 - H \cdot \prod_{p=1}^{r} X_{jp,v}^{p}}} \tag{24}$$

or, equivalently, $X_{i,v} = \sigma(\alpha_{i,v})$ with appropriate weight values.

If we suppose that the output sigmoidal function produces values in]0, 1[, and expanding the term $\alpha_{i,v}$ to account all the possible combinations of signals (corresponding to different kinds of transitions), we can make out different cases. In the following, we distinct between active state transition, which is the state transition currently analyzed according to the total order, while for inactive state transition, we mean the transition which is present in the set of transition of the automaton, but it is not that is currently analyzed. The contributions of the neurons involved in the encoding of the inactive transitions will be referred to as residual inputs. Before proceeding to discuss about the possible cases, let us state what stable means in our concern.

The encoding is said to be *stable* if there exists a tight upper bound for low signals and a lower bounds for high signals. To characterize low and high signals and derive the conditions of stability of the encoding algorithm reported in Section III-C, the definition of two new functions is needed.

First, let us also assume that all low and high signals have the same value and define a new function $l(x, H, n)$ which takes the residual inputs into consideration, i.e.,

$$l(x, H, n) = \frac{1}{1 + e^{H/2 - nHx}} \tag{25}$$

$W_{0,2,2,2}$	$W_{1,2,2,2}$	$W_{2,2,2,2}$	$W_{3,2,2,2}$	$W_{4,2,2,2}$
−H	0	−H	+H	0

$W_{0,4,2,2}$	$W_{1,4,2,2}$	$W_{2,4,2,2}$	$W_{3,4,2,2}$	$W_{4,4,2,2}$
+H	0	−H	0	+H

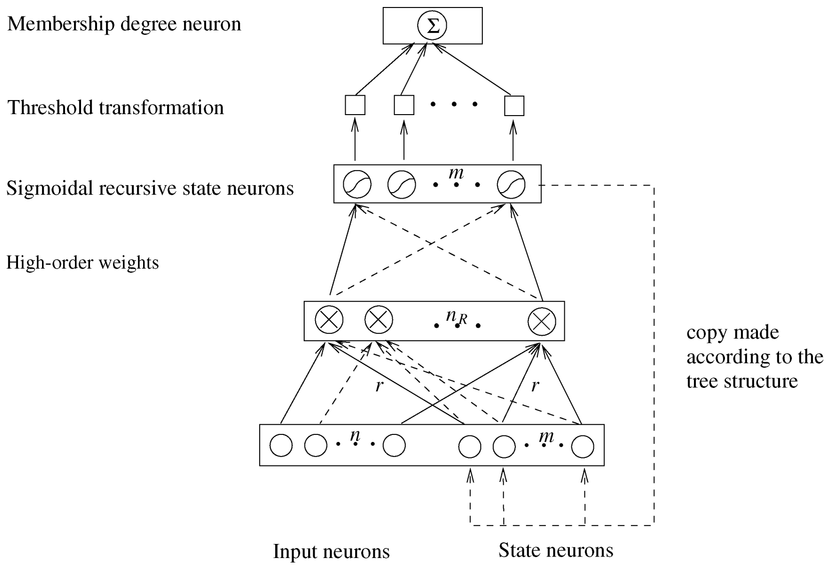

Fig. 11. High-order recursive neural network associated with an NFFRTA.

where n represents the maximum number of residual inputs to a neuron's input. Some results adapted from [28] are useful for the following treatment.

Lemma 1: The function $l(x, H, n)$ has two stable fixed points $0 < \phi_l^-(n,H) < \phi_l^+(n,H) < 1$ and one unstable fixed point $\phi_l^0(n,H)$ if H is chosen such that

$$H > H_0^-(n) = \frac{2(1+(1-x)\log(\frac{1-x}{x}))}{1-x} \tag{26}$$

where $\phi_l^-(n,H)$ denotes the low fixed point and $\phi_l^+(n,H)$ the high fixed point as mentioned before. Again, from [28] it can be demonstrated that: low signals in the transition equations are signals bounded from above by the fixed point $\phi_l^-(n,H)$. To quantify high signals we refer to the following result.

Lemma 2: The function

$$h(x,H,n) = \frac{1}{1+e^{H/2-H(x-\phi_l^-)}} \tag{27}$$

has three fixed points $0 < \phi_h^-(n,H) < \phi_h^0(n,H) < \phi_h^+(n,H) < 1$ *if* H is chosen such that

$$H > H_0^+(n) = \frac{2(1+(1-x)\log(\frac{1-x}{x}))}{(1+2\phi_l^-)(1-x)}. \tag{28}$$

It can be demonstrated that: high signals are signals bounded from below by the fixed point $\phi_h^0(n,H)$ of the function $h(x,H,n)$.

Based on these considerations, let us study the possible cases that can arise. The cases are characterized by the type of input and output states in the transitions.

- *Case 1:* The active state transition is $\delta_{s_k}(S_{j_1}, S_{j_2}, \ldots, S_{j_r}) = S_i$, and possibly there are other state transitions which go into S_i starting from tuples of states not involving $S_{j_1}, \ldots, S_{j_r}$.
- *Case 2:* The active state transition is $\delta_{s_k}(S_{j_1}, S_{j_2}, \ldots, S_{j_r}) = S_i$, and at least one of the indeces $j_1, \ldots, j_r$ is equal to i. This situation will be referred to as a self-loop state transition. Possibly there are other self-loop state transistions which go into S_{j_l}, $l = 1, \ldots, r$, starting from $S_{j_1}, \ldots, S_{j_r}$.

To make the analysis in more details, we shall denote in the sequel with

$$C'_{i,k} = \{(l'_1, l'_2, \ldots, l'_r) | \exists l'_p = i\}$$

the number of r-tuple for which $W_{i,l'_1,l'_2,\ldots,l'_r,k} = H$ and there exists at least an index l'_p equal to i, and with

$$C_{i,k} = \{(l_1, l_2, \ldots, l_r) | l_p \neq i, j_p \; p = 1, \ldots, r\}$$

the number of r-tuples for which $W_{i,l_1,\ldots,l_r,k} = H$ with $l_p \neq i, j_p$, $p = 1, \ldots, r$.

Let us analyze the cases in detail.

Case 1: The current state transition is $\delta_{s_k}(S_{j_1}, S_{j_2}, \ldots, S_{j_r}) = S_i$. Let $X^1_{j_1,v}, X^2_{j_2,v}, \ldots, X^r_{j_r,v}$ be the state neurons corresponding to $S_{j_1}, S_{j_2}, \ldots, S_{j_r}$. Recall that according to our encoding algorithm S_{j_p} may also be void, i.e., $S_{j_p} = \emptyset$. Let us also assume that there are other states $S_{l_1}, S_{l_2}, \ldots, S_{l_r}$ from which one comes into the state S_i, i.e., $\delta_{s_k}(S_{l_1}, S_{l_2}, \ldots, S_{l_r}) = S_i$, such that l_p is different from i, j_p with $p = 1, \ldots, r$. In this case, since the activation $\alpha_{i,v}$ of the current state neuron $X_{i,v}$ is computed over all the neurons incoming into it, low values from neurons $X_{l_p,v}$ contribute to the activation of $X_{i,v}$ with weights $W_{i,l_1,l_2,\ldots,l_r,k} = H$. Two subscases should also be investigated: the case where there is a self-loop transition, referred to in the sequel as subcase 1a) and the case where there is a no-loop transition, referred to as subcase 1b).

Subcase 1a): There exists at least one self-loop transition $\delta_{s_k}(S_{l'_1}, S_{l'_2}, \ldots, S_{l'_r}) = S_i$, $(l'_1, \ldots, l'_r) \in C'_{i,k}$, which is inactive [see Fig. 12(a)]. In

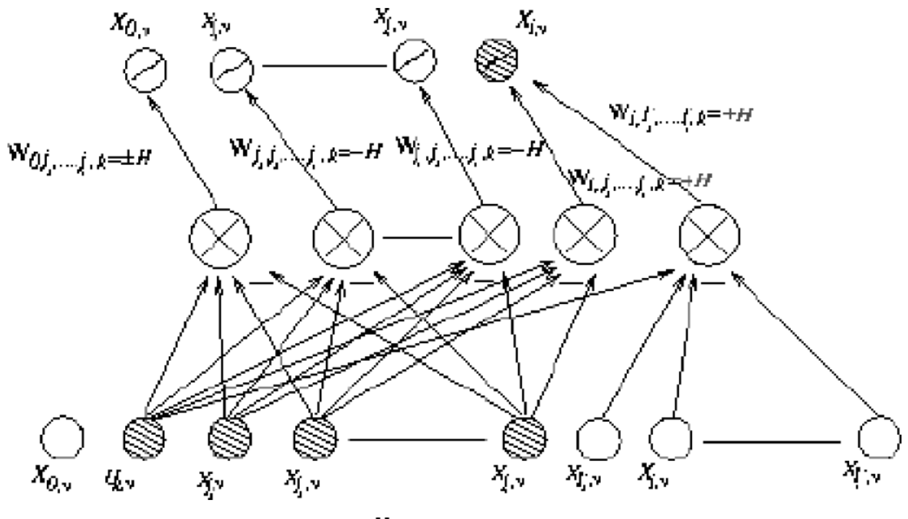
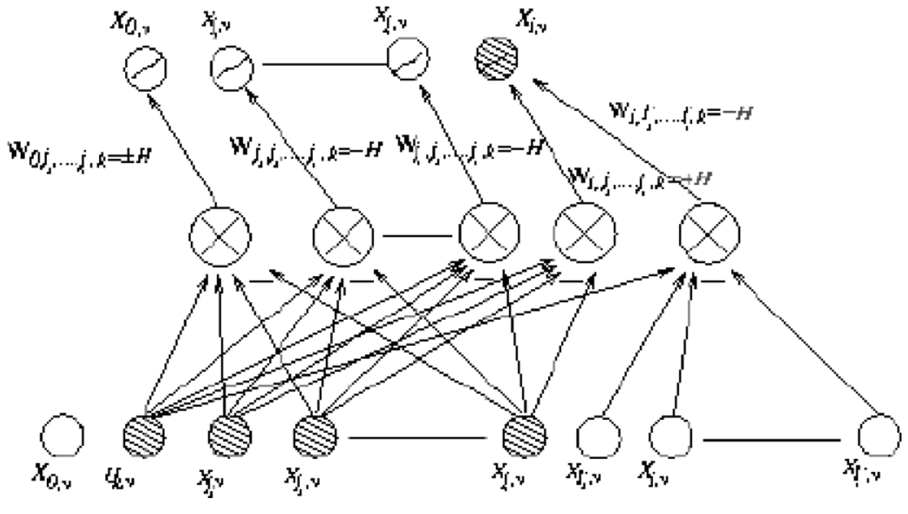

Fig. 12. Figures illustrate the encoding of the state transitions described as subcases 1a) (without self-loop on $X_{i,v}$) and 1b) (with self-loop on $X_{i,v}$). The states partecipating in the active transition are marked with patterned neuron for input states and filled neuron for the output state. To avoid confusion the residual inputs are not shown and just one self-loop has been considered.

this case, it holds $W_{i,l'_1,\ldots,l'_r,k} = H$ and the neuron dynamics is governed by

$$X_{i,v} = \sigma\left(-\frac{H}{2} + H\prod_{p=1}^{r} X^p_{j_p,v} + \sum_{(l'_1,\ldots,l'_r)\in C'_{i,k}} H\prod_{p=1}^{r} X^p_{l'_p,v} + \sum_{(l_1,\ldots,l_r)\in C_{i,k}} H\prod_{p=1}^{r} X^p_{l_p,v}\right). \quad (29)$$

Subcase 1b): There does not exist any self-loop transition $\delta_{s_k}(S_{l'_1}, S_{l'_2}, \ldots, S_{l'_r}) \neq S_i$, $(l'_1,\ldots,l'_r) \in C'_{i,k}$ [see Fig. 12(b)]. Here the activation of neuron i updates as

$$X_{i,v} = \sigma\left(-\frac{H}{2} + H\prod_{p=1}^{r} X^p_{j_p,v} - \sum_{(l'_1,\ldots,l'_r)\in C'_{i,k}} H\prod_{p=1}^{r} X^p_{l'_p,v} \sum_{(l_1,\ldots,l_r)\in C_{i,k}} H\prod_{p=1}^{r} X^p_{l_p,v} + \right). \quad (30)$$

From the analysis of the previous equations, it follows that low values $X_{i,v}$ will be changed into high values (low $X_{i,v} \rightarrow$ *high* $X_{i,v}$) while the values of the state neurons $j_1, j_2, \ldots, j_r$ which contribute to the state transition change their state from high value to low one (high $X_{j_p,v} \rightarrow$ low $X_{j_p,v}$), by

$$X_{j_p,v} = \sigma\left(-\frac{H}{2} - H\prod_{p=1}^{r} X^p_{j_p,v} + \sum_{(l_1,\ldots,l_r)\in C_{i,k}} H\prod_{p=1}^{r} X^p_{l_p,v}\right). \quad (31)$$

Case 2: The current active state transition is a self-loop transition $\delta_{s_k}(S_{l'_1}, S_{l'_2}, \ldots, S_{l'_r}) = S_i$. For this case the output of neuron i remains high, while the neurons $l'_p \neq i$ will change from high values to low ones. The equations governing the neuron state changes high $X_{i,v} \rightarrow$ high $X_{i,v}$ and high $X_{l'_p,v} \rightarrow$ low $X_{l'_p,v}$ become

$$X_{i,v} = \sigma\left(-\frac{H}{2} + \sum_{(l_1,\ldots,l_r)\in C_{i,k}} H\prod_{p=1}^{r} X^p_{l_p,v} + \sum_{(l'_1,\ldots,l'_r)\in C'_{i,k}} H\prod_{p=1}^{r} X^p_{l'_p,v}\right) \quad (32)$$

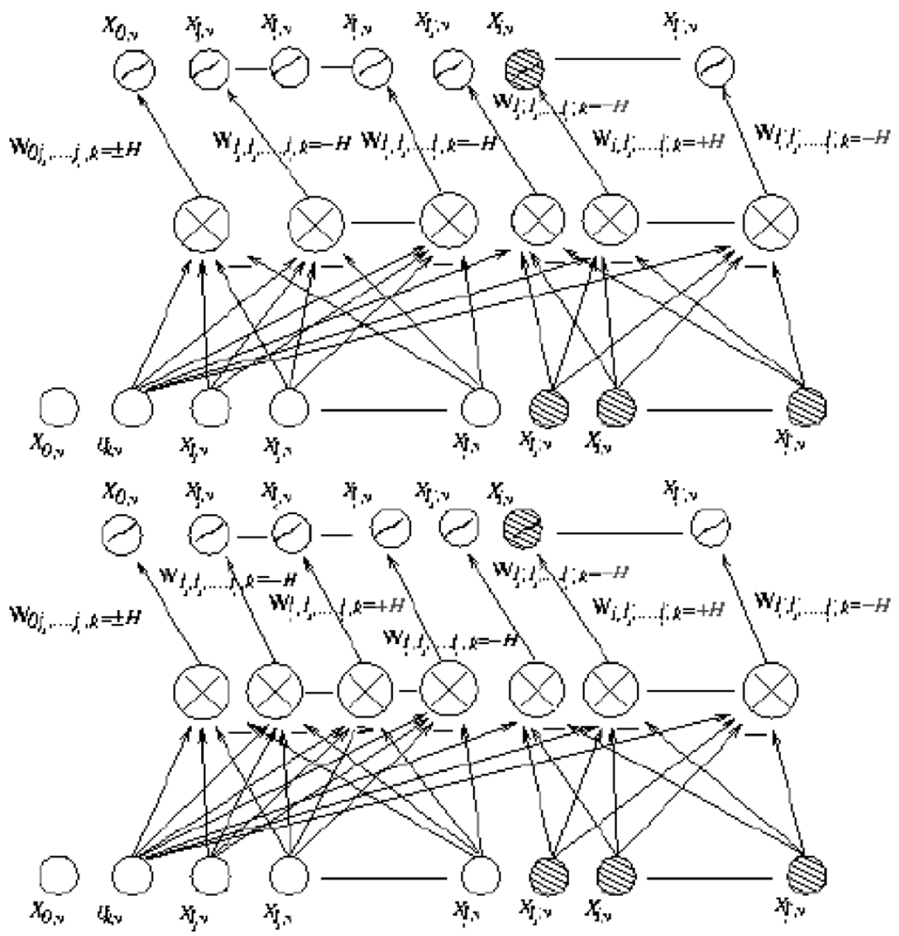

Fig. 13. Figures illustrate the encoding of the state transitions described as subcases 2a) (without self-loop on $X_{l_p,v}$) and 2b) (with self-loop on $X_{l_p,v}$). The states partecipating in the active transition are marked with patterned neuron for input states and filled neuron for the output state. The meaningful involved weight values have been shown in red. To avoid confusion the residual inputs are not shown and just one self-loop has been considered.

$$X_{l'_p,v} = \sigma\left(-\frac{H}{2} - H\prod_{p=1}^{r} X^p_{l'_p,v} + \sum_{(l_1,\ldots,l_r)\in C_{i,k}} H\prod_{p=1}^{r} X^p_{l_p,v}\right). \tag{33}$$

For all cases, the low signals $X_{l_p,v}$, $l_p \neq i, j_1, \ldots, j_r$, will remain low after updating. Also here two subcases can be considered.

Subcase 2a): There exists self-loop state transitions $\delta_{s_k}(S_{l_1}, S_{l_2}, \ldots, S_{l_r}) = S_{l_p}$ [see Fig. 13(a)]. In this case, the equation governing the state neuron change is

$$X_{l_p,v} = \sigma\left(-\frac{H}{2} - H\prod_{p=1}^{r} X^p_{l_p,v} + \sum_{(l'_1,\ldots,l'_r)\in C'_{i,k}} H\prod_{p=1}^{r} X^p_{l'_p,v}\right) \quad l'_p \neq l_p. \tag{34}$$

Subcase 2b): There does not exist self-loop state transitions $\delta_{s_k}(S_{l_1}, S_{l_2}, \ldots, S_{l_r}) = S_{l_p}$ [see Fig. 13(b)]. The equation governing the state neuron change is

$$X_{l_p,v} = \sigma\left(-\frac{H}{2} + H\prod_{p=1}^{r} X^p_{l_p,v} + \sum_{(l'_1,\ldots,l'_r)\in C'_{i,k}} H\prod_{p=1}^{r} X^p_{l'_p,v}\right) \quad l'_p \neq l_p. \tag{35}$$

For the worst case analysis, we investigate only two cases.

1) The case of state changes from the low value of $X_{i,v}$ to a low value when a minimum number of input signals comes into neuron i. This case corresponds to (30).
2) The case of state changes from the low value of $X_{i,v}$ to a high value when a maximum number of input signals comes into neuron i. This case corresponds to the (32).

It is straightforward to note that the stability for these state changes implies the stability for all the remaining weaker state changes. Based on the these considerations, let us consider the state changes governed by the (29) and (32). In the sequel, for sake of clearness, we avoid to indicate explicitely the dependence of the fixed points from n and H.

Consider (29). Since the stability for low to high values implies that the value of $X_{i,v}$ should be lower bounded by ϕ_l^0, the argument of $l()$ must be greater than ϕ_h^0

$$-\frac{H}{2} + H\phi_h^+ - \sum_{p=1}^{r} \binom{r}{p} H\phi_l^- > \phi_h^0. \tag{36}$$

Similarly, for (32), the following inequality must be satisfied:

$$-\frac{H}{2} + m^r H\phi_l^- < \phi_l^0. \tag{37}$$

From conditions (37) and (38), the necessary conditions for the stable encoding of a DFRTA in a $(r+1)$th recursive network may be expressed by the following theorem.

Theorem 3: For a DFRTA with m states, n input symbols, i.e., $n = \#(V \cup \Sigma)$, and r the in-degree of each state, a recursive neural network with m sigmoidal neurons, n input neurons, m biases of values $-H/2$ and n_w $(r+1)$-order weights, $n_w \leq 3 * n * (mr)$, with values in $\{-H, 0, +H\}$ can be built from the DFRTA such that the network is stable if

$$\phi_h^+ > (2^r - 1)\phi_l^- + \frac{1}{2} + \frac{\phi_h^0}{H} \tag{38}$$

$$\phi_l^- < \frac{1}{m^r}\left(\frac{1}{2} + \frac{\phi_l^0}{H}\right). \tag{39}$$

Corollary 1: For a DFRTA with m states, n input symbols, i.e., $n = \#(V \cup \Sigma)$, and r the in-degree of each state, the recursive neural network encoding the DFRTA recognizes the tree accepted by the DFRTA if

$$\phi_l^- < \frac{2}{m^r} \quad ; \quad \phi_h^+ > \frac{1}{2}\left(1 + \frac{1}{m^r} + \frac{2\phi_h^0}{H}\right). \tag{40}$$

Proof: The proof easily derives from the consideration that in the case of a tree t which is not in $\mathcal{L}(\mathrm{DFRTA})$ the neuron output S_0 must be low also in the case of one reached rejecting state, i.e.,

$$-\frac{H}{2} - H\phi_h^+ + (m^r - 1)H\phi_l^- < 0.5 \tag{41}$$

and the same output neuron S_0 must be high with just one reached final accepting state, i.e.,

$$-\frac{H}{2} + H\phi_h^+ - (m^r - 1)H\phi_l^- > 0.5 \tag{42}$$

and from the fact that $\phi_l^- + \phi_h^+ < 2$ [28].

Since the neural network is sparse, i.e., m state neurons are used to encode a DFRTA with m states through the one-hot encoding, and not a dense binary encoding where $\log m$ state neurons should be sufficient, the worst case analysis made for the maximum number of input contributions to the actual state $\delta_{s_k}(S_{j_1}, \ldots, S_{j_r}) = S_i$, has to equivalently be made in the case of the number of nonzero connections to a state neuron is λm^r, where λ is the connectivity factor defined here as the ratio of the maximum number of transitions to any state to m^r. The following corollary holds:

Corollary 2: Given the $(r+1)$-order recursive neural network constructed for a DFRTA with m states, n input symbols, and r the in-degree of each state, let $0 < \lambda \leq 1$ be the connectivity factor of the network. The network stability is guaranteed if

$$\phi_h^+ > (2^r - 1)\phi_l^- + \frac{1}{2} + \frac{\phi_h^0}{H} \tag{43}$$

$$\phi_l^- < \frac{1}{\lambda m^r}\left(\frac{1}{2} + \frac{\phi_l^0}{H}\right). \tag{44}$$

Proof: The proof is straightforward if we substitute the maximum number of state transitions m^r with the restricted number λm^r.

V. Conclusion

In this paper, we have reported a method for injecting nondeterministic fuzzy frontier-to-root tree automata in recursive neural networks by using continuous discriminant functions. Specifically, based on previous results about the encoding of stable representations of finite-state dynamics in recurrent networks, we have proposed a method for encoding fuzzy tree automata in recursive networks which compute the acceptance degree of the automaton to an input string. The algorithm transforms the fuzzy tree automaton into an equivalent deterministic one, which computes the same acceptance degree. A fuzzy membership degree related to each accepting state is associated to the deterministic accepting states, while unaccepting deterministic states have labels equal to zero. The acceptance degree is equal to the membership degree of the last visited accepting state.

We also report a theoretical analysis about the accuracy of the computation in terms of stability of the encoding network. The results we obtain extends those already reported in literature concerning the dynamics of recurrent networks which encode finite state automata. It seems quite natural to expect that the theoretical analysis we have reported can be extended to web automata where relations are expressed in the form of directed acyclic graphs (DAG). In particular, an interesting issue is to extend our results to models capable of dealing with partial orders among the variable described by direct ordered acyclic graphs (DOAGs). This should open the way to the use of fuzzy neural networks in several domains like chemistry, software engineering, etc. where the data to be handled are structured in the form of graphs.

The authors would like to remark that the study we have reported is in the framework of the capability of neural networks to represent automata by their production rules. This can be viewed as a foundation to the problem of learning fuzzy rules from example strings, leading to the situation in which available prior knowledge expressed in terms of fuzzy state transition rules of a tree automata (in our case) are injected into a recursive network, while unknown rules are supposed to be filled in by data-driven learning. This with the aim to reduce the learning complexity, improve generalization and extract knowledge by rules which is in the inner structure of the available data.

References

[1] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[2] A. M. Bianucci, A. Micheli, A. Sperduti, and A. Starita, "Quantitative structure activity relationaships of benzodiazepines by recursive cascade correlation," in *Proc. 1998 IEEE Int. Joint Conf. Neural Networks*, 1998, pp. 117–122.

[3] P. Bottoni, L. Cinque, S. Levialdi, and P. Mussio, "Matching the resolution level to salient image features," *Pattern Recognit.*, vol. 31, no. 1, pp. 89–104, 1998.

[4] R. C. Carrasco and M. L. Forcada, "Simple strategies to encode tree automata in sigmoid recursive neural networks," *IEEE Trans. Knowledge Data Eng.*, vol. 13, pp. 148–156, Mar.–Apr. 2001.

[5] L. Cinque and L. Lombardi, "Shape description by a syntactic pyramidal approach," *Image Vis. Comput.*, vol. 13, no. 8, pp. 599–607, 1995.

[6] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*. New York: Academic, 1980, vol. 144, Mathematics in Science and Engineering, pp. 220–226.

[7] E. Francesconi, P. Frasconi, M. Gori, S. Marinai, J. Q. Sheng, G. Soda, and A. Sperduti, "Logo recognition by recursive neural networks," in *Proc. 2nd IAPR Workshop Graphics Recognition*, 1997, pp. 144–151.

[8] P. Frasconi, M. Gori, and G. Soda, "Recurrent neural networks and prior knowledge for sequence processing: A constrained nondeterministic approach," *Knowl.-Based Syst.*, vol. 8, no. 6, pp. 313–332, 1995.

[9] ——, "Representation of finite state automata in recurrent radial basis function networks," *Mach. Learn.*, vol. 23, pp. 5–32, 1996.

[10] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Trans. Neural Networks*, vol. 9, pp. 768–786, Sept. 1998.

[11] K. S. Fu, *Syntactic Methods in Pattern Recognition*. New York: Academic, 1974.

[12] *Syntactic Pattern Recognition. Applications*, K. S. Fu, Ed., Springer-Verlag, New York, 1976.

[13] F. Gèsceg and M. Steinby, *Tree Automata*. Budapest, Hungary: Akadèmiai Kiadò, 1984.

[14] C. L. Giles, D. Chen, C. Miller, H. Chen, G. Sun, and Y. Lee, "Second-order recurrent neural networks for grammatical inference," in *Proc. Int. Joint Conf. Neural Networks*, vol. II, July 1991, pp. 273–281.

[15] C. L. Giles, G. Kuhn, and R. Williams, "Dynamic recurrent neural networks: Theory and applications," *IEEE Trans. Neural Networks*, vol. 5, pp. 153–156, Jan. 1994.

[16] C. L. Giles and T. Maxwell, "Learning, invariance and generalization in high-order neural networks," *Appl. Opt.*, vol. 26, no. 23, pp. 49–72, 1987.

[17] C. Goller, "A connectionist approach for learning search-control heuristics for automated deduction systems," Ph.D. dissertation, Faculty Comp. Sci., Tech. Univ. Munich, Munich, Germany, 1997.

[18] M. Gori, A. Kuchler, and A. Sperduti, "On the implementation of frontier-to-root tree automata in recursive neural networks," *IEEE Trans. Neural Networks*, vol. 10, pp. 1305–1314, Nov. 1999.

[19] J. Hopcroft and J. Ulmann, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley, 1979.

[20] Y. Inagaki and T. Fukumura, "On the description of fuzzy meaning of context-free languages," in *Fuzzy Sets and their Applications to Cognitive and Decision Processes*, L. A. Zadeh, K. S. Fu, K. Tanaka, and M. Shimura, Eds. New York: Academic, 1975, pp. 301–328.

[21] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[22] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI–7, pp. 693–699, 1985.

[23] A. Kuchler, "On the correspondence between neural folding architectures and tree automata," Faculty Comput. Sci, Univ. Ulm, Ulmer Informatik-Berichte, no. 98-06, 1998.

[24] C.-T. Lin and C. S. G. Lee, *Neural Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[25] A. Matescu, A. Salomaa, K. Salomaa, and S. Yu, "Lexical analysis with a simple finite-fuzzy-automaton model," *J. Univ. Comput. Sci.*, vol. 1, no. 5, pp. 292–311, 1995.

[26] S. Mensch and H. Lipp, "Fuzzy specification of finite state machines," in *Proc. Eur. Design Automation Conf.*, 1990, pp. 622–626.

[27] M. Mizumoto, J. Toyoda, and K. Tanaka, "Fuzzy languages," *Syst. Comput. Controls*, vol. 1, pp. 87–100, 1970.

[28] C. W. Omlin and C. L. Giles, "Constructing deterministic finite-state automata in recurrent neural networks," *J. Assoc. Comput. Mach.*, vol. 43, no. 2, pp. 937–972, 1996.

[29] ——, "Stable encoding of large finite-state automata in recurrent neural networks with sigmoid discriminants," *Neural Computat.*, vol. 8, no. 4, pp. 675–696, 1996.

[30] C. W. Omlin, K. K. Thornber, and C. L. Giles, "Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 6, Feb. 1998.

[31] A. Pathak and S. Pal, "Fuzzy grammars in syntactic recognition of skeletal maturity from X-rays," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, pp. 657–667, Sept.-Oct. 1986.

[32] T. Schmitt and C. Goller, "Relating chemical structure to activity: An application of the neural folding architecture," in *Proc. 5th Int. Workshop Fuzzy-Neuro Systems '98*.

[33] B. J. Sheu, *Neural Information Processing and VLSI*. Norwell, MA: Kluwer, 1995.

[34] J. Sima, "Analog stable simulation of discrete neural networks," *Neural Netw. World*, vol. 7, pp. 679–686, 1997.

[35] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Trans. Neural Networks*, vol. 8, pp. 714–735, May 1997.

[36] A. Sperduti, "On the computational power of recurrent neural networks for structures," *Neural Netw.*, vol. 10, no. 3, pp. 395–400, 1997.

[37] S. Tamura and K. Tanaka, "Learning of fuzzy formal languages," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 98–102, 1973.

[38] J. Thacher, "Tree automata: An informal survey," in *Currents in the Theory of Computing*, A. Aho, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1973, pp. 143–172.

[39] M. G. Thomason, "Finite fuzzy automata, regular fuzzy languages and pattern recognition," *Pattern Recognit.*, vol. 5, pp. 383–390, 1973.

Marco Gori (S'91–M'91–SM'97–F'01) received the Laurea degree in electronic engineering from Università di Firenze, Italy, in 1984, and the Ph.D. degree from the Università di Bologna, Italy, in 1990.

From 1988 to 1989, he was a Visiting Student with the School of Computer Science, McGill University, Montreal, QC, Canada. In 1992, he became an Associate Professor of computer science at Università di Firenze and, in 1995, he joined the University of Siena, where he is currently a Full Professor. His main research interests are in neural networks, pattern recognition, and applications of machine learning to information retrieval on the Internet. He has lead a number of research projects on these themes with either national or international partners, and has been involved in the organization of many scientific events, including the IEEE-INNS International Joint Conference on Neural Networks, for which he acted as the program chair.

Dr. Gori serves as an Associate Editor of a number of technical journals related to his areas of expertise, including Pattern Recognition, the IEEE TRANSACTIONS ON NEURAL NETWORKS, *Neurocomputing*, and the *International Journal on Pattern Recognition and Artificial Intelligence*. He is the Italian chairman of the IEEE Neural Network Council (R.I.G.), is acting as the co-chair of the TC3 technical committee of the International Association for Pattern Recognition (IAPR) on Neural Networks, and is the President of the Italian Association for Artificial Intelligence.

Alfredo Petrosino (M'04) received the Laurea degree (*cum laude*) in computer science from the Universita' di Salerno, Salerno, Italy, in 1989.

He was Fellow Researcher at the Italian National Research Council (CNR) from 1989 to 1994 and of the International Institute for Advanced Scientific Studies (IIASS) in 1995. Since 1996, he has been a Researcher at the National Institute of Physics of Matter (INFM) and from 2000 at CNR, where he became Senior Researcher in 2002. In 2004, he became an Associate Professor of computer science, Universita' Parthenope di Napoli. His main research interests include image processing, pattern recognition, neural networks, multimedia systems, and parallel computing. He is the author of more than 50 referred papers, is co-editor of four international published volumes, and has been involved in the organization of workshops and conferences, including the series of WILF workshops over fuzzy logic theory and applications.

Dr. Petrosino serves as Associate Editor of the *Pattern Recognition* journal and is a member of the International Association for Pattern Recognition (IAPR).