Stable Encoding of Large Finite-State Automata in Recurrent Neural Networks with Sigmoid Discriminants

Christian W. Omlin [a,b], C. Lee Giles [a,c]

[a] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540

[b] CS Department, Rensselaer Polytechnic Institute, Troy, NY 12180

[c] UMIACS, U. of Maryland, College Park, MD 20742

University of Maryland Technical Report CS-TR-3337 & UMIACS-TR-94-101

Abstract

We propose an algorithm for encoding deterministic finite-state automata (DFAs) in second-order recurrent neural networks with sigmoidal discriminant function and we prove that the languages accepted by the constructed network and the DFA are identical. The desired finite-state network dynamics is achieved by programming a small subset of all weights. A worst case analysis reveals a relationship between the weight strength and the maximum allowed network size which guarantees finite-state behavior of the constructed network. We illustrate the method by encoding random DFAs with 10, 100, and 1,000 states. While the theory predicts that the weight strength scales with the DFA size, we find the weight strength to be almost constant for all the experiments. These results can be explained by noting that the generated DFAs represent average cases. We empirically demonstrate the existence of extreme DFAs for which the weight strength scales with DFA size.

1 INTRODUCTION

It is possible to train recurrent neural networks to behave like deterministic finite-state automata [Elman, 1990, Frasconi et al., 1991, Giles et al., 1992, Pollack, 1991, Servan-Schreiber et al., 1991, Watrous and Kuhn, 1992]. The internal representation of learned DFA states can deteriorate due to the dynamical nature of recurrent networks making predictions about the generalization performance of trained recurrent networks difficult [Zeng et al., 1993]. Methods for constructing DFAs in recurrent networks with *hard-limiting* neurons discriminant functions have been proposed [Alon et al., 1991, Horne and Hush, 1994, Minsky, 1967];

methods for constructing networks with sigmoidal and radial-basis discriminant functions are discussed in [Frasconi et al., 1993, Gori et al., 1994, Giles and Omlin, 1993]. We prove that recurrent networks with *continuous sigmoidal* discriminant functions can be constructed such that the encoded finite-state dynamics remains stable indefinitely. Notice that we do not claim that such a stable representation can be *learned.* The stability of the internal DFA representation is based on on the premise that the discriminant functions have sufficiently high gain such that the neurons always operate in a saturated mode.

We show empirically that the gain does not scale with the size of networks that implement *average* DFAs; however, there exist extreme cases of DFAs where we observe a scaling of the neurons' gain with network size.

2 ENCODING DFA DYNAMICS

2.1 Finite State Automata

A deterministic finite-state automaton (DFA) is an acceptor for a regular language $L(M)$. Formally, a DFA M is a 5-tuple $M = < \Sigma, Q, R, F, \delta >$ where $\Sigma = \{a_1, \ldots, a_m\}$ is the alphabet of the language L, $Q = \{q_1, \ldots, q_n\}$ is a set of states, $R \in Q$ is the start state, $F \subseteq Q$ is a set of accepting states and $\delta : Q \times \Sigma \rightarrow Q$ defines state transitions in M. A string is accepted by the DFA M if an accepting state is reached; otherwise, the string is rejected.

2.2 Recurrent Network

We implement DFAs in discrete-time, recurrent networks with second-order weights W_{ijk}. The continuous network dynamics are described by the following equations:

$$S_i^{(t+1)} = h(a_i(t)) = \frac{1}{1 + e^{-a_i(t)}}, \qquad a_i(t) = b_i + \sum_{j,k} W_{ijk} S_j^{(t)} I_k^{(t)}, \tag{1}$$

where b_i is the bias associated with hidden recurrent state neurons S_i; I_k denotes the input neuron for symbol a_k. The product $S_j^{(t)} I_k^{(t)}$ directly corresponds to the state transition $\delta(q_j, a_k) = q_i$. After a string has been processed, the output of a designated neuron S_0 decides whether the network accepts or rejects a string. The network accepts a given string if the value of the output neuron S_0^t at the end of the string is greater than 0.5; otherwise, the network rejects the string.

2.3 Encoding Algorithm

The encoding algorithm achieves a nearly orthonormal internal representation of the desired DFA dynamics; it constructs a network with $n+1$ recurrent state neurons (including the output neuron) and m input neurons from a DFA with n states and m input symbols. There is a one-to-one correspondence between state neurons S_i and DFA states q_i. Consider a DFA state transition $\delta(q_j, a_k) = q_i$. Setting W_{ijk} to a large positive value $+H$ will ensure that $S_i^{(t+1)}$ will be high and setting W_{jjk} to a large negative value $-H$ will guarantee that the output $S_j^{(t+1)}$ will be low. Furthermore, if state q_i is an accepting state, then we program the weight W_{0jk} to $+H$; otherwise, we set W_{0jk} to $-H$. We set the bias terms b_i of all state neurons S_i to $-H/2$. For each DFA state transition, at most three weights of the network have to be programmed. The initial state $\mathbf{S}^0$ of the network is $\mathbf{S}^0 = (S_0^0, 1, 0, 0, \ldots, 0)$. The initial value of the response neuron S_0^0 is 0 if the DFA's initial state q_0 is a rejecting state and 1 otherwise.

3 ANALYSIS

We prove the stability of DFA encodings in recurrent neural networks using fixed point analysis. We only give the proofs of the theorems which establish our results; for proofs of auxiliary lemmas see [Omlin, 1995].

3.1 Fixed Point Analysis

Recall that the recurrent network changes its state according to equation (1). Our DFA encoding algorithm yields a special form of that equation describing the dynamics of a constructed network:

$$S_i^{(t+1)} = g(H(2x_i - 1)/2) = h(x, H) = \frac{1}{1 + e^{H/2(1-2x)}} \tag{2}$$

The bias term $-H/2$ is common to all state neurons. Hx is the weighted sum feeding into neuron $S_i^{(t+1)}$. Under certain conditions, the discriminant function $h(.)$ has fixed points which allow a stable internal representation of DFA states. We state here without proof the following facts about fixed points of the function $h(.)$; the proofs can be found in [Omlin, 1995].

Lemma 3.1.1 *For $0 < H < 4$, $h(x, H)$ has the following fixed point:*

$$\phi^0 = 0.5$$

Furthermore, $h(x, H)$ converge to ϕ^0 for any choice of a start value x_0.

Lemma 3.1.2 *For $H \geq 4$, $h(x, H)$ has three fixed points $\phi^0 = 0.5$, ϕ^- and ϕ^+.*

Lemma 3.1.3 *for $x < \phi^0$ and $\phi^0 < x$, $h^t(x, H)$ (with $H > 4$) converges to ϕ^- and ϕ^+, respectively.*

Lemma 3.1.4 *For arbitrary $H > 0$, the two fixed points ϕ^- and ϕ^+ are related as follows:*

$$\phi^- + \phi^+ = 1$$

3.2 Worst Case Analysis

We will now investigate the conditions under which a constructed network implements a given DFA. The main result shows how large a network may be for fixed H such that the languages of the DFA and the constructed network are identical:

Theorem 3.2.1 *Let ρ denote the maximum fraction of DFA states q_j from which there are state transitions $\delta(q_j, a_k) = q_i$ for a fixed choice of a_k and any q_i. Then, a sparse recurrent neural network RNN with $n{+}1$ sigmoidal state neurons, m input neurons, at most $3mn$ second-order weights with alphabet $\Sigma_w = \{-H, 0, +H\}$ $(4 < H_{min} < H < H_{pred})$, $n+1$ biases with alphabet $\Sigma_b = \{-H/2\}$, and maximum fan-out $3m$ can be constructed from a DFA M with n states and m input symbols of states such that the internal state representation remains stable, i.e. $S_i > 0.5$ when q_i is the current DFA state and $S_i < 0.5$ otherwise if*

$$n < \frac{1}{2\ \rho\ \phi_\Delta^-(H)}(1 + \frac{1}{H}) \quad with \quad \phi_\Delta^+(H) > \frac{1}{4}(3 + \frac{1}{H})$$

for a proper choice of H.

Proof: ¿From the DFA encoding algorithm, we can derive four different types of neuron state changes:

low $\rightarrow$ high:

$$S_i^{t+1} = h(S_j^t * H + \sum_{S_l \epsilon C_{i,l}} S_l^t * H - S_i^t * H) \quad (S_j^t : high, S_l^t, S_i^t : low) \tag{3}$$

$$S_i^{t+1} = h(S_j^t * H + \sum_{S_l \epsilon C_{i,l}} S_l^t * H + S_i^t * H) \quad (S_j^t : high, S_l^t, S_i^t : low) \tag{4}$$

high $\rightarrow$ high:

$$S_i^{t+1} = h(S_i^t * H + \sum_{S_l \epsilon C_{i,l}} S_l^t * H) \quad (S_i^t : high, S_l^t : low) \tag{5}$$

high $\rightarrow$ low:

$$S_i^{t+1} = h(S_i^t * H + \sum_{S_l \epsilon C_{i,l}} S_l^t * H) \quad (S_i^t : high, S_l^t : low) \tag{6}$$

low $\rightarrow$ low:

$$S_i^{t+1} = h(S_i^t * H + \sum_{S_l \epsilon C_{i,l}} S_l^t * H) \quad (S_i^t, S_l^t : low) \tag{7}$$

$$S_i^{t+1} = h(S_i^t * H + \sum_{S_l \epsilon C_{i,l}} S_l^t * H) \quad (S_i^t, S_l^t : low) \tag{8}$$

where

$$C_{i,l} = \{S_j \ \mid \ W_{ijl} = H\} \tag{9}$$

The inputs I_k^t are not shown explicitly since we assume that each input symbol is assigned a separate input neuron in a one-hot encoding. The DFA state transitions corresponding to the these types of neuron state changes are shown in figure 1.

Note that the term $-H/2$ is implicit; they cause neuron outputs which do not correspond to the current DFA state to be reset to a small value. The term $S_j^t * H$ where S_j^t is a high signal represents the *principal contribution* to the neuron S_i^{t+1} which is responsible for driving the output of neuron S_i^{t+1} high when the network executes a DFA state transition $\delta(q_j, a_k) = q_i$. All other terms are the *residual contributions* to the input of neuron S_i^{t+1} where the signal S_l^t is low. The term $\sum S_l^t * H$ contributes to the total input of state neuron S_i^{t+1} if there are other transitions $\delta(q_l, a_k) = q_i$ in the DFA from which the recurrent network is constructed. Since there is a one-to-one correspondence between state neurons and DFA states, there will always be a negative contribution $-S_i^t * H$ for the current DFA state transition $\delta(q_j, a_k) = q_i$, i.e. only S_i^t can drive the signal S_i^{t+1} low. Equations (3) and (4) only differ with respect to the sign of the residual input $S_i^t * H$. If there is a state transition $\delta(q_i, a_k) = q_i$, then equation (4) applies; otherwise, there is a residual low signal S_i^t trying to drive S_i^{t+1} low and equation (3) applies. Similarly, either equation (7) or (8) is chosen for state transitions of the type $low \rightarrow low$. The above equations account for all possible contributions to the net input of all state neurons because the encoding algorithm constructs a *sparse* recurrent network.

Before we proceed, we make some observations about the equations (3)-(9) which will simplify the analysis.

Observation 3.2.1 *An analysis for state transitions of type* $low \rightarrow high$ *and* $low \rightarrow low$ *also covers state transitions of type* $high \rightarrow high$ *and* $high \rightarrow low$*, respectively.*

The state transition types $high \rightarrow high$ and $high \rightarrow low$ cause stronger high and low signals, respectively, than the state transitions types $low \rightarrow high$ and $low \rightarrow low$. Thus, an analysis that shows stability of high and low signals for these types of state transitions implies the stability of high and low signals for state transitions of type $high \rightarrow high$ and $high \rightarrow low$, respectively.

Observation 3.2.2 *Of the two possible equations (3) and (4) for state transitions* $low \rightarrow high$*, the former equation also covers the latter equation.*

If high signals can be kept stable for equation (3), then they certainly can also be kept stable for equation (4). No separate analysis is necessary for thetwo equations.

Observation 3.2.3 *Of the two possible equations (7) and (8) for state transitions* $low \rightarrow low$*, the latter equation also covers the former equation.*

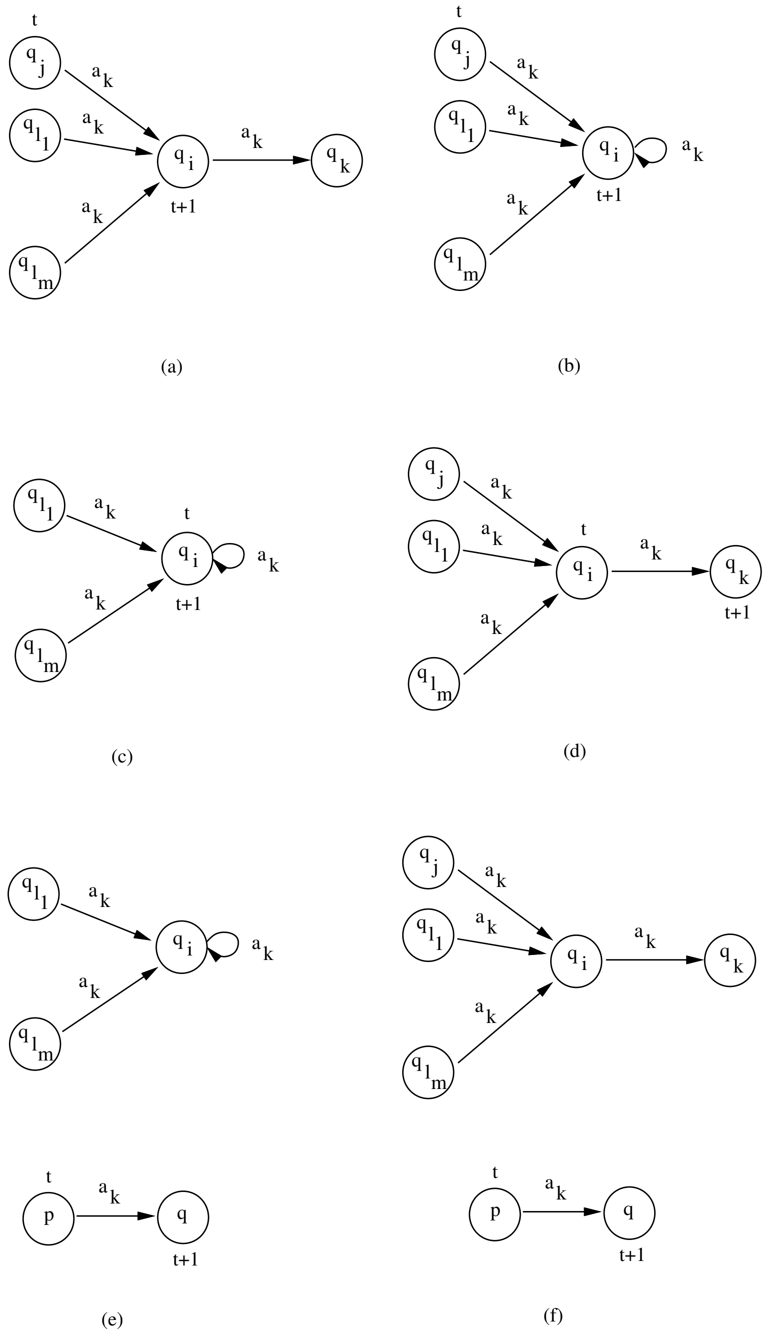


Figure 1: **Neuron State Changes and Corresponding DFA State Transitions**: The figure (a)-(f) illustrate the DFA state transitions corresponding to all possible state changes of neuron S_i; the DFA state(s) participating in the current transitions are marked with t and $t+1$. (a) $low \rightarrow high$ (no self-loop on q_i) (b) $low \rightarrow high$ (with self-loop on q_i) (c) $high \rightarrow high$ (necessarily a self-loop on q_i) (d) $high \rightarrow low$ (necessarily no self-loop on q_i) (e) $low \rightarrow low$ (no self-loop on q_i) (f) $low \rightarrow low$ (with self-loop on q_i). Notice that, even though state q_i is neither the source nor the target of the current state transition in cases (e) and (f), the corresponding state neuron S_i still receives residual inputs from state neurons $S_{l_1}, \ldots, S_{l_m}$.

An argument similar to that given for validity of observation 3.2.2 can be given.

Thus, we are left with only the two following types of state transitions which represent the worst cases:

low $\rightarrow$ *low*:

$$S_i^{t+1} = h(S_i^t * H + \sum_{S_l \epsilon C_{i,l}} S_l^t * H) \quad (S_i^t, S_l^t : low) \tag{10}$$

low $\rightarrow$ *high*:

$$S_i^{t+1} = h(S_j^t * H + \sum_{S_l \epsilon C_{i,l}} S_l^t * H - S_i^t * H) \quad (S_j^t : high, S_l^t, S_i^t : low) \tag{11}$$

Thus, the network equation (1) takes on the special form

$$a_i \equiv -H/2 + Hx_i \;\; with \;\; x_i = \sum_j S_j^t \tag{12}$$

since all but one input neuron have value 0 at any given time t. Notice that the number of terms in the sum $\sum_j S_j^t$ is equal to the number of DFA states q_j for which there are transitions $\delta(q_j, a_k) = q_i$ for the current input symbol a_k.

The ideal case where neurons do not receive residual inputs from other neurons, successive network state changes can be expressed as the iteration of the function $h^t(x, H)$

$$h^t(x, H) \;\; = \;\; \begin{cases} h(x, H) & t = 1 \\ h(h^{t-1}(x, H), H) & t > 1 \end{cases} \tag{13}$$

with $x = 0$ and $x = 1$ for low and high signals, respectively

We can now define a new function $h_\Delta^t(x, H)$ which takes the residual inputs into consideration.

Since the initial output value of all state neurons except the neuron assigned to a DFAs start state are zero, the residual inputs under this worst case assumption are identical for all neurons; let Δx denote the residual neuron inputs. Then, the function $h_\Delta^t(x, H)$ is defined as

$$h_\Delta^t(x, H) \;\; = \;\; \begin{cases} h(x, H) & t = 1 \\ h(h_\Delta^{t-1}(x + \Delta x, H) + \Delta x, H) & t > 1 \end{cases} \tag{14}$$

The initial values for low and high signals are $x = 0$ and $x = 1$, respectively.

Consider a state q_i and let D_{ik} denote the number of states q_j such that $\delta(q_j, a_k) = q_i$ for each symbol a_k. Setting $D = \max\{D_{ik}\}$, each recurrent state neuron receives residual input from at most $\rho = \frac{D}{n}$ recurrent neurons for a chosen input symbol a_k.

We make the following simplifying assumption:

Assumption 3.2.1 *Each state neuron receives residual inputs from exactly ρ other neurons for all input symbols.*

Although this is generally not the case, this worst case covers all possible DFAs and simplifies the analysis.

Then, we can quantify Δx for the case of low signals:

Lemma 3.2.1 *The low signals are bounded from above by the fixed point ϕ_Δ^- of the function*

$$h^t_{\Delta^-}(x, H) = \begin{cases} h(0, H) & t = 1 \\ h(\rho \cdot n \cdot h^{t-1}_{\Delta^-}(x, H), H) & t > 1 \end{cases} \tag{15}$$

Similarly, we can quantify high signals:

Lemma 3.2.2 *The high signals are bounded from below by the fixed point ϕ_Δ^+ of the function*

$$h^t_{\Delta^+}(x, H) = \begin{cases} h(1, H) & t = 1 \\ h(h^{t-1}_{\Delta^+}(x, H) - h^{t-1}_{\Delta^-}(x, H), H) & t > 1 \end{cases} \tag{16}$$

The derivation of these iterated functions can be found in [Omlin, 1995]; the functions (16) and (17) converge toward their fixed points ϕ_Δ^- and ϕ_Δ^+ according to lemma 3.1.3.

In practice, only few neurons ever exceed or fall below the fixed points ϕ^- and ϕ^+, respectively. Furthermore, the network has a built-in reset mechanism which allows low and high signals to be strengthened. Low signals S_j^t are strengthened to $g(-H/2)$ when there exists no state transition $\delta(., a_k) = q_j$. In that case, the neuron S_j^t receives no inputs from any of the other neurons; its output becomes less than ϕ^- since $g(-H/2) = h(0, H) < \phi^-$ for $H > 4$. Similarly, high signals S_i^t get strengthened if either low signals feeding into neuron S_i on a current state transition $\delta(\{q_j\}, a_k) = q_i$ have been strengthened during the previous time step or when the number of positive residual inputs to neuron S_i compensates for a weak high signal from neurons $\{q_j\}$. Thus only a small number of neurons will have $S_j^t > \phi^-$ or $S_j^t < \phi^+$ and only for a finite amount of time. For the majority of neurons we have $S_j^t \leq \phi^-$ and $S_j^t \geq \phi^+$. Since constructed recurrrent networks are able to regenerate their internal signals and since typical DFAs do not have the worst case properties assumed in this analysis, the conditions guaranteeing stable low and high signals are generally much too strong for some given DFA. Scaling issues are discussed elsewhere [Omlin and Giles, 1994].

In order for the internal DFA state representation to remain stable, the low and high signals must remain sufficiently distinguishable:

Definition 3.2.1 *An encoding of DFA states in a SORNN is called* <u>*stable*</u> *if all the low and high signals are less and larger than 0.5, respectively.*

We now return to the worst case state equations (10) and (11).

In order for the function (16) to converge toward the low fixed point ϕ_Δ^-, the argument of equation (10) must satisfy the following invariant property:

$$-\frac{H}{2} + H * n * \rho * \phi_\Delta^- < \frac{1}{2} \tag{17}$$

Similarly, the argument of equation (11) must satisfy the following invariant property in order for function (17) to converge toward the high fixed point ϕ_Δ^+:

$$-\frac{H}{2} + H * \phi_\Delta^+ - H(1 - \phi^+)\,\frac{1}{2}. \tag{18}$$

Solving inequalities (18) and (19) for n and ϕ_Δ^+, respectively, we obtain the conditions for stable low and high signals of the theorem.

Although we have stated the conditions for stable low and high signals separately, the condition for stable signals can be simplified as follows:

Corollary 3.2.1 *If all neurons (not including the output neuron) have at least two weights $W_{ixk} = W_{iyk} = H$ for all input symbols a_k, then the internal representation of a DFA remains stable if*

$$\phi_\Delta^-(H) < \frac{1}{2n}(1 + \frac{1}{H})$$

Proof: Substituting $1 - \phi_\Delta^+$ for ϕ_Δ^- in theorem 3.2.1 we get

$$1 - \phi_\Delta^+ < \frac{1}{2n}(1 + \frac{1}{H}) \tag{19}$$

$$1 - \frac{1}{2n}(1 + \frac{1}{H}) < \phi_\Delta^+ \tag{20}$$

$$\frac{2n-1}{2n} + \frac{2n-1}{2Hn} < \phi_\Delta^+ \tag{21}$$

But stable high signals require

$$\frac{3}{4} + \frac{1}{4H} < \phi_{\Delta}^{+} \tag{22}$$

Comparing inequalities (22) and (23), we conclude that the former implies the latter for $n \geq 2$.

We can now state the following theorem for the construction of recurrent networks for specific DFAs:

Corollary 3.2.2 *Let $L(M_{DFA})$ denote the language accepted by a DFA M with n states and let $L(M_{RNN})$ be the language accepted by the sparse RNN constructed from M; then, we have $L(M_{RNN}) = L(M_{DFA})$ if*

$$\phi_{\Delta}^{-}(H) < \frac{1}{2n} \quad \textit{and} \quad \phi_{\Delta}^{+}(H) > \frac{1}{4}\,(3 + \frac{1}{H})$$

Proof: We just need to establish a condition for correct string classification. As we will see, the condition for stable dynamics in partially recurrent networks is not sufficient to guarantee correct string classification.

For the case of an ungrammatical strings, the following condition must be satisfied:

$$-\frac{H}{2} - H * \phi_{\Delta}^{+} + (n-1) * H * \phi_{\Delta}^{-} < \frac{1}{2} \tag{23}$$

where we have made the usual simplification about the convergence of the outputs to the fixed points ϕ_{Δ}^{-} and ϕ_{Δ}^{+}; furthermore, we assume that only one DFA state is a rejecting state; then the output neuron's residual inputs from all other state neurons is positive, weakening the intended high signal for the network's output neuron. Notice that the output neuron is the only neuron which can be forced toward a low signal by neurons other than itself.

A similar condition can be formulated for grammatical strings:

$$-\frac{H}{2} + H * \phi_{\Delta}^{+} - (n-1) * H * \phi_{\Delta}^{-} > \frac{1}{2} \tag{24}$$

The above two inequalities can be simplified into a single inequality:

$$-2 * H * \phi_{\Delta}^{+} + 2 * (n-1) * H * \phi_{\Delta}^{-} < 0 \tag{25}$$

Solving for ϕ_{Δ}^{-}, we get the following condition for the correct output of a network:

$$\phi_{\Delta}^{-} < \frac{1}{n} \tag{26}$$

Thus we have the following conditions for stable low signals and correct string classification: for $(H > 1)$.

$$\phi_{\Delta}^{-}(H) \;<\; \begin{cases} \frac{1}{2\rho n}\,(1 + \frac{1}{H}) < \frac{1}{\rho n} \quad \textit{for} \;\; H > 1 & \text{(dynamics)} \\ \frac{1}{n} & \text{(classification)} \end{cases} \tag{27}$$

Comparing these two conditions, it follows that the condition for correct string classification implies the condition for stable low signals for $\lambda rho \leq 1$.

Other scenarios of worst cases can be considered: Instead of partial interconnectivity, a worst case analysis can be carried out where it is assumed that each neuron receives inputs from all other neurons. In that case, the conditions for stable finite-state dynamics dominate the condition for correct string classification. Obviously, those conditions imply the conditions for partially recurrent networks. It is also possible to consider a fully recurrent networks where all those weights that are not programmed to $+H$ or $-H$ are assigned random values from an interval $[-W, W]$. That scenario applies when a recurrent network initialized with the knowledge of some DFA is further trained on data for knowledge refinement of revision; in that case, the weight strength H and the value W can be chosen such that the weights initialized to random values do not destroy the encoded knowledge.

4 EXPERIMENTS

In order to empirically validate our analysis, we constructed networks from randomly generated DFAs with 10, 100 and 1,000 states. For each of the three DFAs, we randomly generated different test sets each consisting of 1,000 strings of length 10, 100, and 1,000, respectively. The randomly generated, minimized 100-state DFA with alphabet $\Sigma = \{0, 1\}$ we encoded into a recurrent network with 101 state neurons is shown in figure 2. The networks' generalization performance on these test sets for rule strength $H = \{0.0, 0.1, 0.2, \ldots, 7.0\}$ are shown in figures 3-5. A misclassification of these long strings for arbitrary large values of H would indicate a network's failure to maintain the stable finite-state dynamics that was encoded. However, we observe that the networks can implement stable DFAs as indicated by the perfect generalization performance for some choice of the rule strength H and chosen test set. Thus, we have empirical evidence which supports our analysis.

All three networks achieve perfect generalization for all three test sets for approximately the same value of H. Apparently, the network size plays an insignificant role in determining for which value of H stability of the internal DFA representation is reached, at least across the considered 3 orders of magnitude of network sizes.

For the 100-state DFA in figure 2, we computed the value $H_{pred} = 13.8$ which guarantees stable finite-state dynamics of the constructed network. H_{pred} was computed from a worst case analysis; it should thus come as no surprise that $4 < H_{exp} < H_{pred}$.

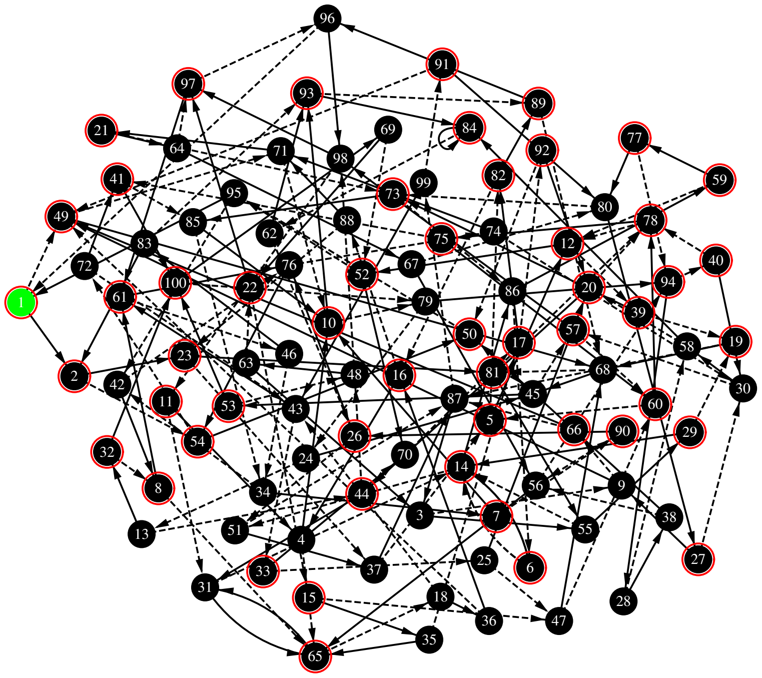


Figure 2: **Randomly generated 100-state DFA:** The minimal DFA has 100 states and alphabet $\Sigma = \{0, 1\}$. State 1 is the start state. States with and without double circles are accepting and rejecting states, respectively.

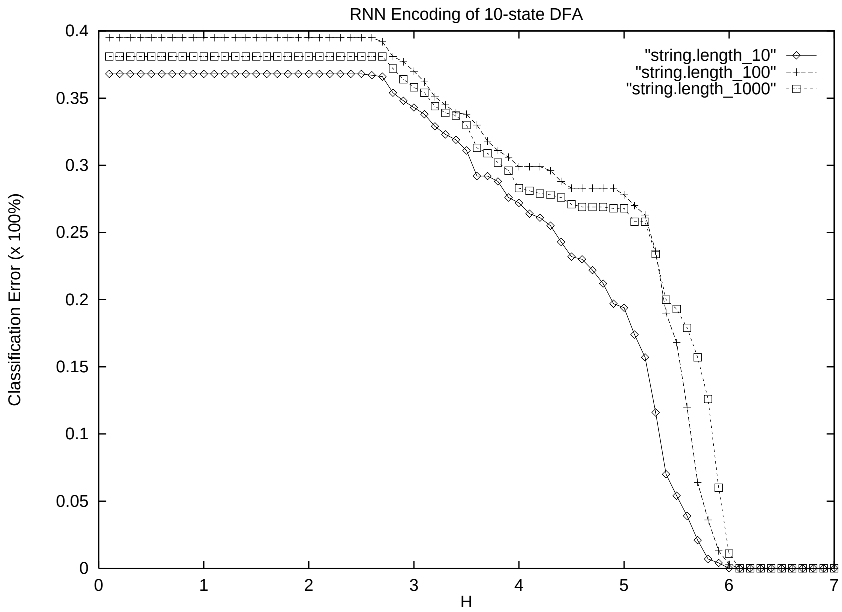


Figure 3: **Performance of 10-state DFA:** The network classification performance on three randomly-generated data sets consisting of 1,000 strings of length 10 ($\Diamond$), 100 ($+$), and 1,000 ($\Box$), respectively, as a function of the rule strength H (in 0.1 increments) is shown. The network achieves perfect classification on the strings of length 1,000 for $H > 6.0$.

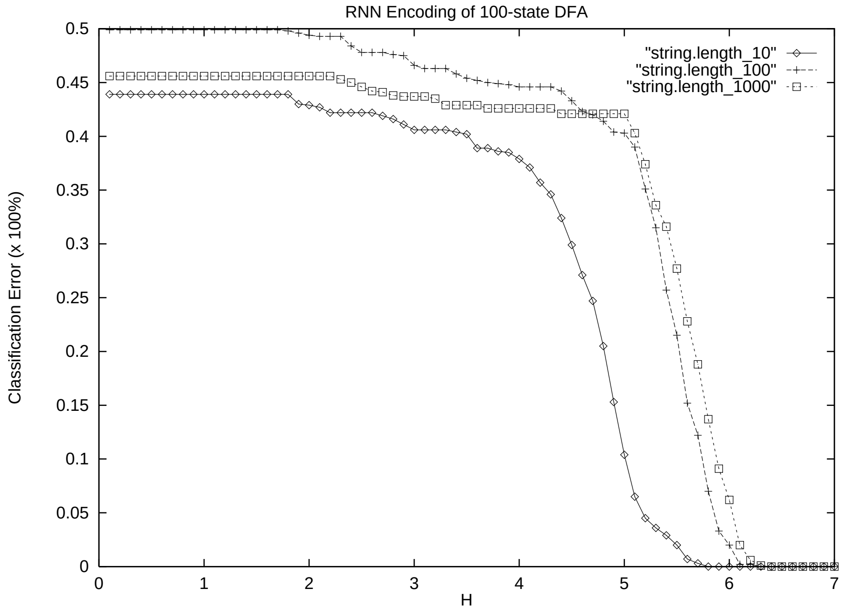


Figure 4: **Performance of 100-state DFA:** The network classification performance on three randomly-generated data sets consisting of 1,000 strings of length 10 ($\diamond$), 100 ($+$), and 1,000 ($\square$), respectively, as a function of the rule strength H (in 0.1 increments) is shown. The network achieves perfect classification on the strings of length 1,000 for $H > 6.2$.

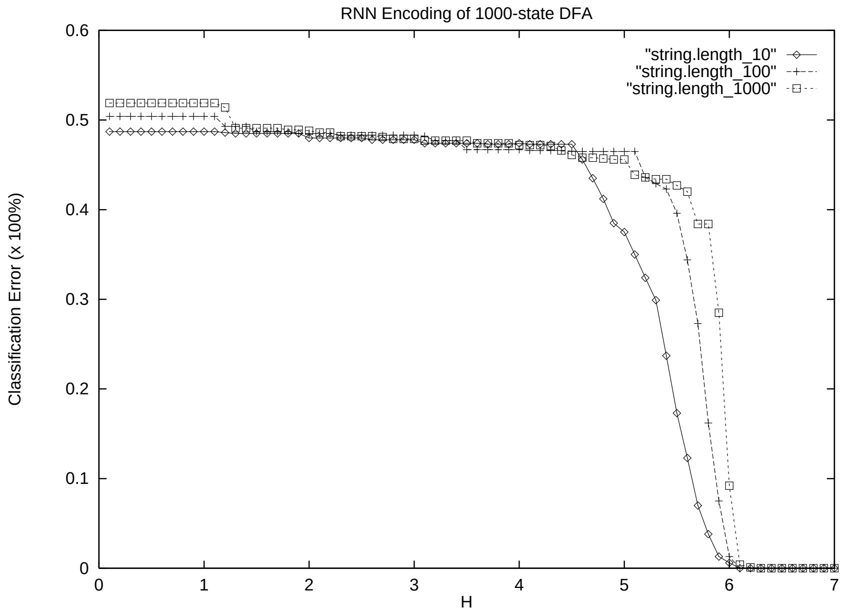


Figure 5: **Performance of 1000-state DFA:** The network classification performances on three randomly-generated data sets consisting of 1,000 strings of length 10 ($\diamond$), 100 ($+$), and 1,000 ($\square$), respectively, as a function of the rule strength H (in 0.1 increments). The network achieves perfect classification on the strings of length 1,000 for $H > 6.1$.

5 SCALING ISSUES

5.1 Preliminaries

The worst case analysis of section makes the following predictions about the implementation of arbitrary DFAs:

(1) neural DFAs can be constructed that are stable for arbitrary string length for finite value of the weight strength H,

(2) for most neural DFA implementations, $H_{emp} < H_{pred}$, and

(3) the value of H scales with the DFA size, i.e. the larger the DFA and thus the network, the larger H will be for guaranteed stability.

Predictions (1) and (2) are supported by our experiments. However, when we compare the values H_{emp} in the above experiments for DFAs of different sizes, we find that $H_{emp} \approx 6$ for all three DFAs. This observation seems inconsistent with the theory. The reason for this inconsistency lies in the assumption of a *worst case* for the analysis, whereas the DFAs we implemented represent *average cases.* For the construction of the randomly generated 100-state DFA we found correct classification of strings of length 1,000 for $H_{emp} = 6.3$. This value corresponds to a DFA whose states have 'average' indegree $n = 1.5$. [The magic value 6 also seems to occur for networks which are trained. Consider a neuron S_i; then, the weight which causes transitions between dynamical attractors often has a value ≈ 6 [Tino, 1994].]

However, there exist DFAs which exhibit the scaling behavior that is predicted by the theory. We will briefly discuss such DFAs. That discussion will be followed by an analysis of the condition for stable DFA encodings for asymptotically large DFAs.

5.2 DFA States with Large Indegree

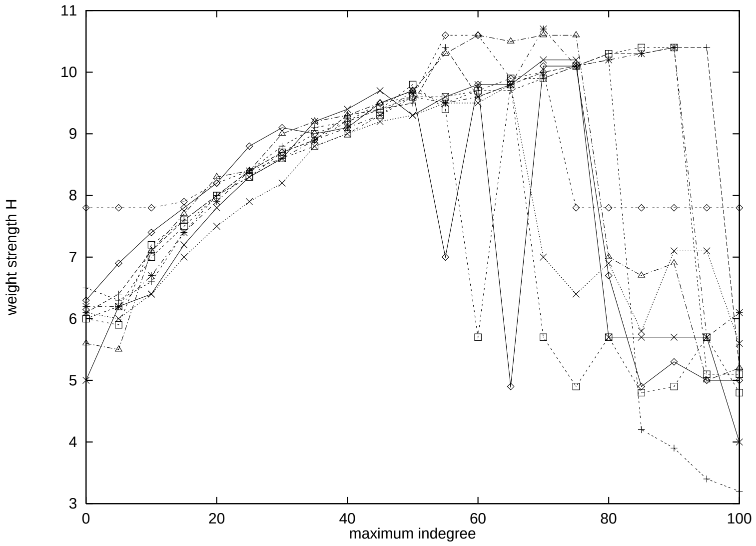
We can approximate the worst case analysis by considering an extreme case of a DFA:

(1) Select an arbitrary DFA state q_ρ;

(2) select a fraction ρ of states q_j and set $\delta(q_j, a_k) = q_\rho$.

(3) For low values of ρ, a constructed network behaves similar to a randomly generated DFA.

(4) As the number of states q_j for which $\delta(q_j, a_k) = q_\rho$ increases, the behavior gradually moves toward the worst case analysis where one neuron. receives a large number of residual inputs with for a designated input symbol a_k.



captionScaling Weight Strength: An accepting state q_ρ in 10 randomly generated 100-state DFAs was selected. The number of states q_j for which $\delta(q_j, 0) = q_\rho$ was gradually increased in increments of 5% of all DFA states. The graph shows the minimum value of H_{emp} for correct classification of 100 strings of length 100. H_{emp} increases up to $\rho = 75\%$; for $\rho > 75\%$, the DFA becomes degenerated causing H_{emp} to decrease again.

We constructed a network from a randomly generated DFA M_0 with 100 states and two input symbols. We derived DFAs $M_{\rho_1}, M_{\rho_2}, \ldots, M_{\rho_R}$ where thefraction of DFA states q_j from M_{ρ_i} to $M_{\rho_{i+1}}$ with $\delta(q_j, a_k) = q_\rho$ increased by $\Delta\rho$; for our experiments, we chose $\Delta\rho = 0.05$. Obviously, the languages $L(M_{\rho_i})$ change for different values of ρ_i. The graph in figure 6 shows for 10 randomly generated DFAs with 100 state the minimum weight strength H necessary to correctly classify 100 strings of length 100 - a new data set was randomly generated for each DFA - as a function of ρ in 5% increments. We observe that H_{emp} generally increases with increasing values of ρ; in all cases, the hint strength H_{emp} sharply decline for some percentage value ρ. As the number of connections $+H$ to a single state neuron S_i increases, the number of residual inputs which can cause unstable internal DFA representation and incorrect classification decreases. Let us assume that the extreme DFA state q_ρ is an accepting state. Then, the input to output neuron S_0^{t+1} is

$$-\frac{H}{2} + H * S_\rho^t + \sum_{q_l \in F} S_l^t * H - \sum_{q_l \notin F} S_l^t * H \tag{28}$$

For correct classification, the net input must be larger than 0.5. As the value of ρ increases, the number of terms in the first and second sum increase and decrease, respectively. Thus, smaller values of H lead to correct string classification. A similar argument can be made if q_ρ is a rejecting state.

We observe that there are two runs where outliers occur, i.e. $H_{\rho_i} > H_{\rho_{i+1}}$ even though we have $\rho_i < \rho_{i+1}$. Since the value H_ρ depends on the randomly generated DFA, the choice for q_ρ and the test set, we can expect such an uncharacteristic behavior to occur in some cases.

5.3 Asymptotic Case Analysis

We are interested in finding an expression for the average number of residual inputs to a neuron in *large* DFAs. Since we are dealing with a second-order network architecture, disjoint parts of the network participate in the computation of the next state for any given input symbol. Thus, we can limit our analysis to DFAs with a single input symbol.

Consider a DFA M and its underlying graph $G(V, E)$ whose vertices V and directed edges E are the DFA states Q and state transitions δ, respectively. We assume that $G(V, E)$ is randomly generated: For any given vertex v_j, a directed edge e_{ij} is drawn to another vertex v_i with equal probability $1/n$ for all vertices of G. The number of directed edges entering any given vertex v_i from other vertices v_m is the number of residual inputs state neuron S_i receives from other state neurons S_m. Thus we only need to compute the expected number of incoming edges ("in-degree") for a DFA generated according to the above probability distribution.

The probability $p(d = k)$ for a vertex to have in-degree k follows a binomial distribution; thus, the average in-degree is given by the expected value of k which can be written as:

$$E\{d = k\} = \sum_{k=1}^{n} k \begin{pmatrix} n \\ k \end{pmatrix} \frac{1}{n^k} (1 - \frac{1}{n})^{n-k}$$

For $n \to \infty$ and $\lambda = np \approx 1$ where p is the probability that an event occurs (in our case we have $p = 1/n$ and thus $\lambda = 1$) and $p \to 0$ the binomial distribution asymptotically converges toward the Poisson distribution:

$$E\{d = k\} = \sum_{k=1}^{\infty} k \ e^{-\lambda} \ \frac{\lambda^k}{k!}$$

With $\lambda = 1$, we conclude

$$E\{d = k\} = e^{-1} \sum_{k=1}^{\infty} \frac{1}{(k-1)!} = e^{-1} \sum_{k=0}^{\infty} \frac{1}{k!} = e^{-1} \ e = 1$$

We can now state the following asymptotic result for the construction of large DFAs:

Theorem 5.3.1 *Let n denote the number of states in a DFA. For $n \to \infty$, the languages accepted by the DFA M and the constructed neural RNN are identical only for $H \to \infty$.*

Proof: Recall the worst case equations (10) and (11) for state transitions of type *low* $\rightarrow$ *low* and *low* $\rightarrow$ *high*. For the asymptotic case $n \rightarrow \infty$, these equations simplify.

The worst case equations of section 3.2.1 apply here also; however, the residual inputs are zero. Thus the following two conditions for stable low and high signals, respectively, must be satisfied:

$$-\frac{H}{2} + H * \phi_{\Delta}^{-} < \frac{1}{2} \tag{29}$$

and

$$-\frac{H}{2} + H * \phi_{\Delta}^{+} - H * \phi_{\Delta}^{-} < \frac{1}{2} \tag{30}$$

Combining these two inequalities and solving for ϕ_{Δ}^{-} leads to the condition $\phi_{\Delta}^{-}(H) < \frac{1}{3}$. Thus, we have the following conditions for stability of the finite-state dynamics and correct string classification in asymptotically large DFAs:

$$\textit{stability of signals}: \; \phi_{\Delta}^{-}(H) < \frac{1}{3}$$

$$\textit{correct string classification}: \; \phi_{\Delta}^{-}(H) < \frac{1}{n}$$

Unlike in the case of the worst case analysis for partially recurrent networks,the condition for correct string classification dominates the conditions for stable finite-state dynamics. As a matter of fact, stable finite-state dynamics alone does not require $H \rightarrow \infty$; however, correct string classification requires $\phi_{\Delta}^{-} \rightarrow 0$ and thus $H \rightarrow \infty$ for $n \rightarrow \infty$.

6 CONCLUSION

We investigated how deterministic finite-state automata (DFAs) can be encoded into sparse second-order recurrent neural networks. The operation performed by the second-order architecture is akin to DFA state transitions, making DFA encoding a straightforward operation. We have proven that our algorithm can construct a *sparse* recurrent network with $O(n)$ state neurons, $O(mn)$ weights and limited fan-out of size $O(m)$ from any DFA state with n states and m input symbols such that the DFA and the constructed network accept the same regular language. The DFA dynamics is achieved by programming some of the weights to values $+H$ or $-H$. A worst case analysis has revealed a quantitative relationship between the rule strength H_{pred} and the maximum allowed network size such that the network dynamics remains robust for arbitrary string length. This is only a proof of existence, i.e. we do not make any claims that such a solution can be *learned.*

Our empirical results suggest, that the weight strength $H_{exp} \approx 6$ is independent of the network size for typical DFAs. Extreme DFAs can be constructed for the weight strength scales with the network size, i.e. H_{exp} approaches H_{pred}.

7 ACKNOWLEDGMENT

We would like to acknowledge useful discussions with B.G. Horne, L. R. Leerink, and T. Lin.

References

[Alon et al., 1991] Alon, N., Dewdney, A., and Ott, T. (1991). Efficient simulation of finite automata by neural nets. *Journal of the Association for Computing Machinery*, 38(2):495–514.

[Elman, 1990] Elman, J. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

[Frasconi et al., 1991] Frasconi, P., Gori, M., Maggini, M., and Soda, G. (1991). A unified approach for integrating explicit knowledge and learning by example in recurrent networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, page 811. IEEE 91CH3049-4.

[Frasconi et al., 1993] Frasconi, P., Gori, M., and Soda, G. (1993). Injecting nondeterministic finite state automata into recurrent networks. Technical report, Dipartimento di Sistemi e Informatica, Università di Firenze, Italy, Florence, Italy.

[Giles et al., 1992] Giles, C., Miller, C., Chen, D., Chen, H., Sun, G., and Lee, Y. (1992). Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3):380.

[Giles and Omlin, 1993] Giles, C. and Omlin, C. (1993). Extraction, insertion and refinement of symbolic rules in dynamically driven recurrent neural networks. *Connection Science*, 5(3 & 4):307–337.

[Gori et al., 1994] Gori, M., Maggini, M., and Soda, G. (1994). Insertion of finite state automata in recurrent radial basis function networks. Technical report, Dipartimento di Sistemi e Informatica, Università di Firenze, Italy.

[Horne and Hush, 1994] Horne, B. and Hush, D. (1994). Bounds on the complexity of recurrent neural network implementations of finite state machines. In *Advances in Neural Information Processing Systems 6*, pages 359–366. Morgan Kaufmann.

[Minsky, 1967] Minsky, M. (1967). *Computation: Finite and Infinite Machines*, chapter 3, pages 32–66. Prentice-Hall, Inc., Englewood Cliffs, NJ.

[Omlin, 1995] Omlin, C. (1995). *Symbolic Knowledge in Recurrent Neural Networks: Issues of Training and Representation.* PhD thesis, Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180.

[Omlin and Giles, 1994] Omlin, C. and Giles, C. (1994). Stable encoding of large finite-state automata in recurrent neural networks with sigmoid discriminants. *Neural Computation.* Submitted.

[Pollack, 1991] Pollack, J. (1991). The induction of dynamical recognizers. *Machine Learning*, 7:227–252.

[Servan-Schreiber et al., 1991] Servan-Schreiber, D., Cleeremans, A., and McClelland, J. (1991). Graded state machine: The representation of temporal contingencies in simple recurrent networks. *Machine Learning*, 7:161.

[Tino, 1994] Tino, P. (1994). Personal communication.

[Watrous and Kuhn, 1992] Watrous, R. and Kuhn, G. (1992). Induction of finite-state languages using second-order recurrent networks. *Neural Computation*, 4(3):406.

[Zeng et al., 1993] Zeng, Z., Goodman, R., and Smyth, P. (1993). Learning finite state machines with self-clustering recurrent networks. *Neural Computation*, 5(6):976–990.