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

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];

* To appear in *Neural Computation.*

methods for constructing networks with sigmoidal and radial-basis discriminant functions are discussed in [Alquézar and Sanfeliu, 1995, 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*.

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, $R \epsilon Q$ 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 input neurons. 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.

For the remainder of this paper, we assume a one-hot encoding for input symbols a_k, i.e. $I_k^t \in \{0, 1\}$.

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. For each DFA state transition $\delta(q_j, a_k) = q_i$, we set W_{ijk} to a large positive value $+H$. The self connection W_{jjk} is set to $-H$ (i.e. neuron S_j changes its state from high

to low) except for state transitions $\delta(q_j, a_k) = q_j$ (self-loops) where W_{jjk} is set to $+H$ (i.e. state of neuron S_j remains high). 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 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. All weights that are not set to $-H$, $-H/2$ or $+H$ are set to zero.

The question this paper addresses is whether the value of H can be chosen such that the finite-state dynamics in a recurrent network remains indefinitely stable.

3 ANALYSIS

We prove the stability of DFA encodings in recurrent neural networks for strings of arbitrary length. Due to space limitations, we only give the proofs of the theorems which establish our results; for proofs of auxiliary lemmas see [Omlin and Giles, 1994].

3.1 Fixed Point Analysis for Sigmoidal Discriminant Function

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:

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

The bias term $-H/2$ is common to all state neurons. Hx_i^t 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 and Giles, 1994].

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 $x < \phi^0$ ($\phi^0 < x$), $h^p(x, H)$ (with $H > 4$) converges to ϕ^- (ϕ^+).*

Lemma 3.1.4 *For arbitrary $H > 4$, the two fixed points ϕ^- and ϕ^+ are related as follows:*

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

3.2 Network Dynamics as Iterated Functions

When a network processes a string, the state neurons go through a sequence of state changes. The network state at time $t+1$ is computed from the network state at time t, its current input and its weights. Since the discriminant function $h(.)$ is the same for all state neurons, these network state changes can be represented as iterations of $h(.)$ for each state neuron:

$$S_i^{t+1} = h^{t+1}(x_i^t, H) \;=\; \begin{cases} h(x_i^0, H) & t = 0 \\ h(h^t(x_i^{t-1}, H), H) & t > 0 \end{cases} \tag{3}$$

A network will only correctly classify strings of arbitrary length if its internal DFA state representation remains sufficiently stable. Stability can only be guaranteed if the neurons are shown to operate near their saturation regions for sufficiently high gain of the sigmoidal discriminant function $h(.)$. One way to achieve stability is thus to show that the iteration of the discriminant function $h(.)$ converges toward its fixed points in these regions, i.e. points for which we have i.e. $h(x_i, H) = x_i$. This observation will be the basis for a quantitative analysis which establishes bounds on the network size and the weight strength H which guarantee stable internal representation for arbitrary DFAs.

Given a stable DFA encoding where neurons operate near their saturated regions, each neuron can send two kinds of signals to other neurons:

(1) High signals: If neuron S_i^t represents the current DFA state q_i, then S_i^t will be high (S_i^t: high).

(2) Low signals; Neurons S_j^t which do not represent the current DFA state have a low output (S_j^t: low).

Recall that the arguments of the discriminant function $h(x, H)$ were the sum of unweighted signals x and the weight strength H. We now expand the term x to account for the different kinds of signals that are present in a neural DFA.

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

$low \rightarrow high$:

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

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

$high \rightarrow high$:

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

$high \rightarrow low$:

$$S_i^{t+1} = h(-S_i^t + \sum_{S_l \in C_{i,k}} S_l^t, H) \quad (S_i^t : high, S_l^t : low) \tag{7}$$

$low \rightarrow low$:

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

$$S_i^{t+1} = h(S_i^t + \sum_{S_l \in C_{i,k}} S_l^t, H) \quad (S_i^t, S_l^t : low) \tag{9}$$

where

$$C_{i,k} = \{S_l \mid W_{ilk} = H, l \neq i, l \neq j\} \tag{10}$$

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 these types of neuron state changes are shown in figure 1.

The signals S_i^t and S_j^t represent the *principal contributions* to the neuron S_i^{t+1} which are responsible for driving the output of neuron S_i^{t+1} low or high. All other terms are the *residual contributions* to the input of neuron S_i^{t+1}. The term $\sum S_l^t$ 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 $-S_i^t$ for the current DFA state transition $\delta(q_j, a_k) = q_i$ (assuming $q_j \neq q_i$), i.e. only S_i^t can drive the signal S_i^{t+1} 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.

For a worst case analysis, it suffices to investigate the cases of minimum and maximum neuron inputs for high and low signals, respectively. The equations (4)-(9) condense to the following two equations:
$low \rightarrow low$:

$$S_i^{t+1} = h(S_i^t + \sum_{S_l \in C_{i,k}} S_l^t, H) \quad (S_i^t, S_l^t : low) \tag{11}$$

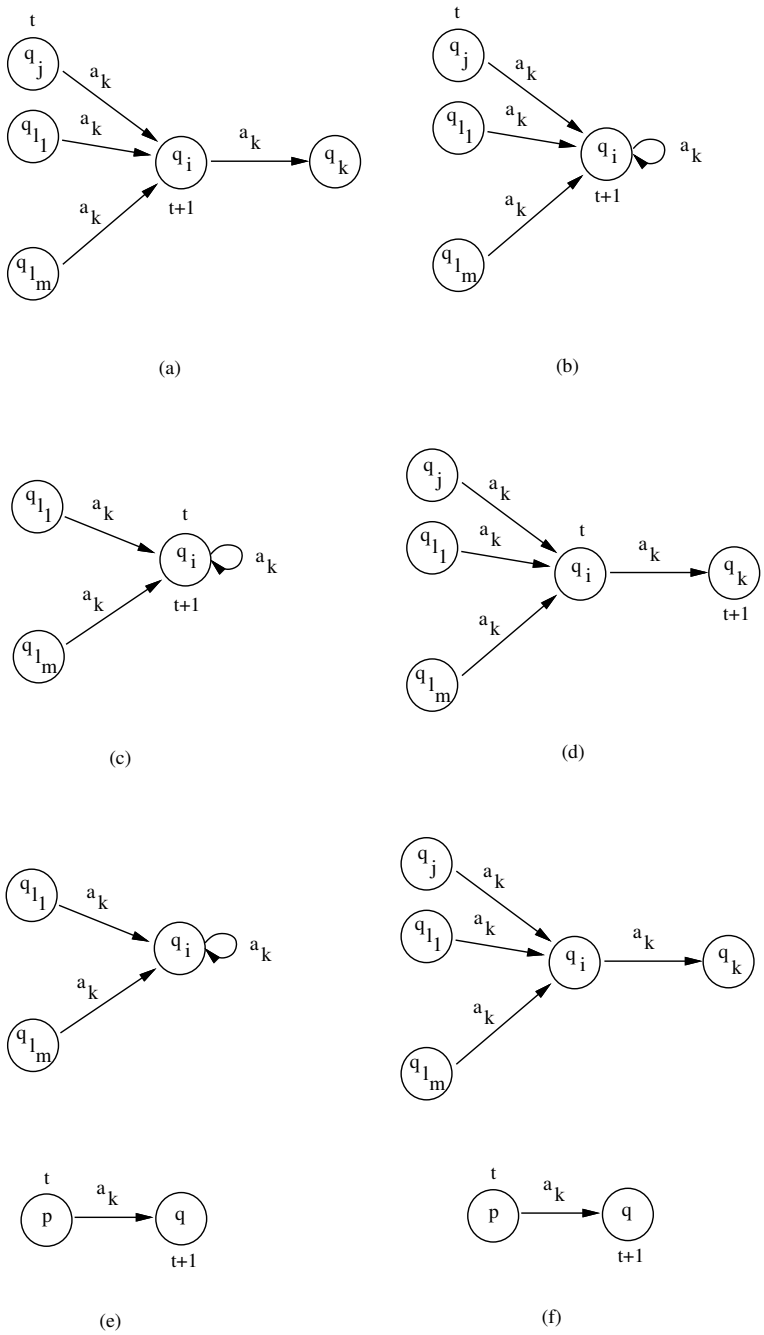

Figure 1: **Neuron State Changes and Corresponding DFA State Transitions**: The figure (a)-(f) illustrate the DFA state transitions corresponding to all possible state changes of neuron S_i; the DFA state(s) participating in the current transitions are marked with t and $t+1$. (a) $low \rightarrow high$ (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 $S_{l_1}, \ldots, S_{l_m}$.

$low \rightarrow high$:

$$S_i^{t+1} = h(S_j^t + \sum_{S_l \in C_{i,k}} S_l^t - S_i^t, H) \quad (S_j^t : high, S_l^t, S_i^t : low) \tag{12}$$

We now define a new function $h_\Delta(x_i, H)$ which takes the residual inputs into consideration. Let Δx_i^t denote the residual neuron inputs to neuron S_i^{t+1}. Then, the function $h_\Delta^{t+1}(x_i^t, H)$ is recursively defined as

$$x_i^{t+1} = h_\Delta^{t+1}(x_i^0, H) \;=\; \begin{cases} h(x_i^0, H) & t = 0 \\ h(h_\Delta^t(x_i^{t-1} + \Delta x_i^{t-1}, H) + \Delta x_i^t, H) & t > 0 \end{cases} \tag{13}$$

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

The magnitude of the residual inputs Δx_i depend on the coupling between recurrent state neurons. Neurons which are connected to a large number of other neurons will receive a larger residual input than neurons which are connected to only a few other neurons. Consider the neuron S_m which receives a residual input Δx_m from the most number r of neurons, i.e. $\Delta x_i \leq \Delta x_m$. In order to show network stability, it suffices to assume the worst case where all neurons receive the same amount of residual input for given time index t, i.e. Δx_m^t. This assumption is valid since the initial value for all neurons except the neuron corresponding to a DFA's start state is 0.

We can quantify Δx_i 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*

$$x_i^{t+1} = h_{\Delta^-}^{t+1}(x_i^0, H) \;=\; \begin{cases} h(0, H) & t = 0 \\ h(r \cdot h_{\Delta^-}^t(x_i^{t-1}, H), H) & t > 0 \end{cases} \tag{14}$$

i.e. we have $\Delta x_i^t = r \cdot h_{\Delta^-}^{t-1}(x_i, H)$ since $x_i^0 = 0$ for low signals in equation (13). This lemma can easily be proven by induction on t.

The graph in figure 2 shows the convergence of the function $h_{\Delta^-}^t(x_i^{t-1}, H)$ towards its fixed points ϕ_Δ^- and ϕ_Δ^+ for different choices of r.

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*

$$x_i^{t+1} = h_{\Delta^+}^{t+1}(x_i^0, H) \;=\; \begin{cases} h(1, H) & t = 0 \\ h(h_{\Delta^+}^t(x_i^{t-1}, H) - h_{\Delta^-}^t(x_i^{t-1}, H), H) & t > 0 \end{cases} \tag{15}$$

Again, this lemma is easily proven if we assume the worst case for neuron state transitions $low \rightarrow high$ where that neuron receives no residual inputs that would strengthen the high signal.

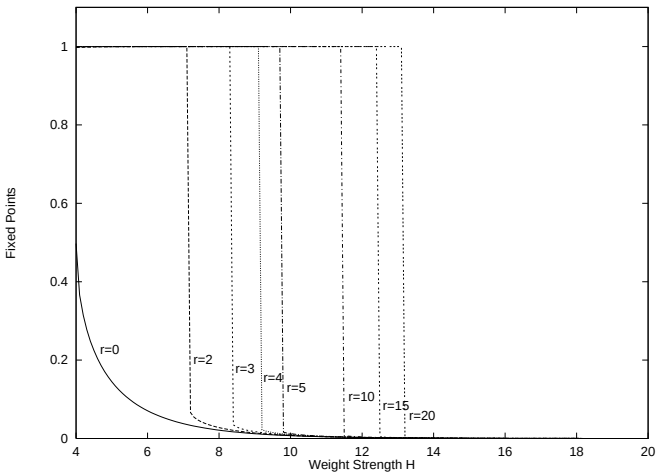

Figure 2: **Effect of Residual Inputs on Low Signals**: Convergence of the function $h^t_{\Delta^-}(x_i, H) = h(r \cdot h^{t-1}_{\Delta^-}\Delta(x_i, H), H)$ toward fixed points $\phi^-_\Delta > \phi^-$ and $\phi^+_\Delta < \phi^+$ as a function of weight strength H and number r of residual values Δx (shown are $r = \{2, 3, 4, 5, 10, 15, 20\}$). The function converges toward a low fixed point if its argument $x_i + \Delta x_i$ is less than 0.5; otherwise, it converges toward ϕ^+_Δ.

Notice that the functions $h_{\Delta^-}(x_i, H)$ and $h_{\Delta^+}(x_i, H)$ have identical fixed points ϕ^-_Δ and ϕ^+_Δ since they are defined in terms of the same function $h(x_i, H)$. They only differ in their initial values for x_i and the residual input Δx_i.

3.3 Network Stability

We now define stability of recurrent networks constructed from DFAs:

Definition 3.3.1 *An encoding of DFA states in a second-order recurrent neural network is called* <u>stable</u> *if all the low and high signals are less and larger than 0.5, respectively.*

Before we analyze the conditions for stable low and high signals, we introduce the following notations: Let D_{ik} denote the number of states q_j that make transitions to state q_i for input symbol a_k. We further define $D_i = max\{D_{ik}\}$ (maximum number of transitions to q_i over all input symbols) and $D = max\{D_i\}$ (maximum number of transitions to any state over all input symbols). Then, $\rho = D/n$ denotes the maximum fraction of all states q_j for which $\delta(\{q_j\}, a_k) = q_i$.

Consider equation (11). In order for the low signal to remain less 0.5, the argument of $h^t_{\Delta^-}(.)$ must be less than 0.5 for all values of t. Each neuron receives residual inputs from at most $D = \rho * n$ other state neurons. Thus, we require the following invariant property of the residual inputs:

$$-\frac{H}{2} + H * \rho * n * \phi^-_\Delta < \frac{1}{2} \tag{16}$$

where we assumed that all low signals have the same value and that their maximum values is the fixed point ϕ_Δ^-. This assumption is justified since the output of all state neurons with low values are initialized to zero.

A similar analysis can be carried out for state transitions of equation (12). The following inequality must be satisfied for stable high signals:

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

where we assumed that there is only one DFA transition $\delta(q_j, a_k) = q_i$ for chosen q_i and a_k, and thus $\sum_{S_l \in C_{i,k}} = 0$.

Solving inequalities (16) and (17) for n and ϕ_Δ^+, respectively, we obtain conditions under which a constructed recurrent network implements a given DFA. These conditions are expressed in the following theorem:

Theorem 3.3.1 *For some given DFA M with n states and m input symbols, let D denote the maximum number of transitions to any state over all input symbols of M, and let $\rho = D/n$. Then, a sparse recurrent neural network with $n+1$ sigmoidal state neurons and m input neurons can be constructed from M 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*

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

for a proper choice of H. Furthermore, the constructed network has at most $3mn$ second-order weights with alphabet $\Sigma_w = \{-H, 0, +H\}$ $(H > 4)$, $n+1$ biases with alphabet $\Sigma_b = \{-H/2\}$, and maximum fan-out $3m$.

Stable encoding of DFA state is a necessary condition for a neural network to implement a given DFA. The network must also correctly classify all strings. The conditions for correct string classification are expressed in the following corollary:

Corollary 3.3.1 *Let $L(M_{DFA})$ denote the regular language accepted by a DFA M with n states and let $L(M_{RNN})$ be the language accepted by the recurrent network constructed from M. Then, we have $L(M_{RNN}) = L(M_{DFA})$ if*

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

Proof: For the case of an ungrammatical strings, the input to the response neuron S_0 must satisfy the following condition:

$$-\frac{H}{2} - H * \phi_\Delta^+ + (n-1) * H * \phi_\Delta^- < \frac{1}{2} \tag{18}$$

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 the state q_i of the state transition $\delta(q_j, a_k) = q_i$ is the only rejecting state; then the output neuron's residual inputs from all other state neurons is positive, weakening the intended low 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:

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

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

$$-2 * H * \phi_\Delta^+ + 2 * (n-1) * H * \phi_\Delta^- < 0 \tag{20}$$

Solving for ϕ_Δ^-, we get the following condition for the correct output of a network:

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

Thus we have the following conditions for stable low signals and correct string classification:

$$\phi_\Delta^-(H) \;<\; \begin{cases} \frac{1}{2\rho n}\,(1+\frac{1}{H}) & \text{(dynamics)} \\ \frac{1}{n} & \text{(classification)} \end{cases} \tag{22}$$

We observe that

$$\frac{1}{2\rho n}(1+\frac{1}{H}) > \frac{1}{2\rho n} > \frac{1}{2n} \quad \text{for} \quad \rho < 1$$

Choosing $\phi_\Delta^-(H) < \frac{1}{2n}$ thus implies the condition for stable low signals in partially recurrent networks.

4 EXPERIMENTS

4.1 Simulation Results

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 3. 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 4-6. A misclassification of these long strings indicates 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

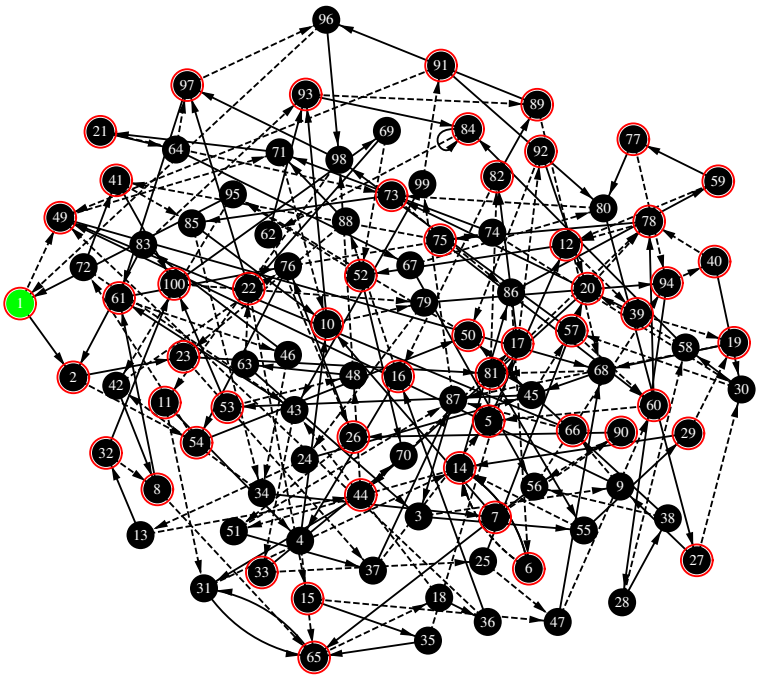

Figure 3: **Randomly generated 100-state DFA:** The minimal DFA has 100 states and alphabet $\Sigma = \{0, 1\}$. State 1 is the start state. States with and without double circles are accepting and rejecting states, respectively.

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.

4.2 Discussion

In our simulations, only few neurons ever exceeded or fell 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 $h(0, H)$ 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 $h(0, H) < \phi^-$ 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 $S_j^t < \phi^+$. For the majority of neurons we have $S_j^t \leq \phi^-$ and $S_j^t \geq \phi^+$. Since constructed 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.

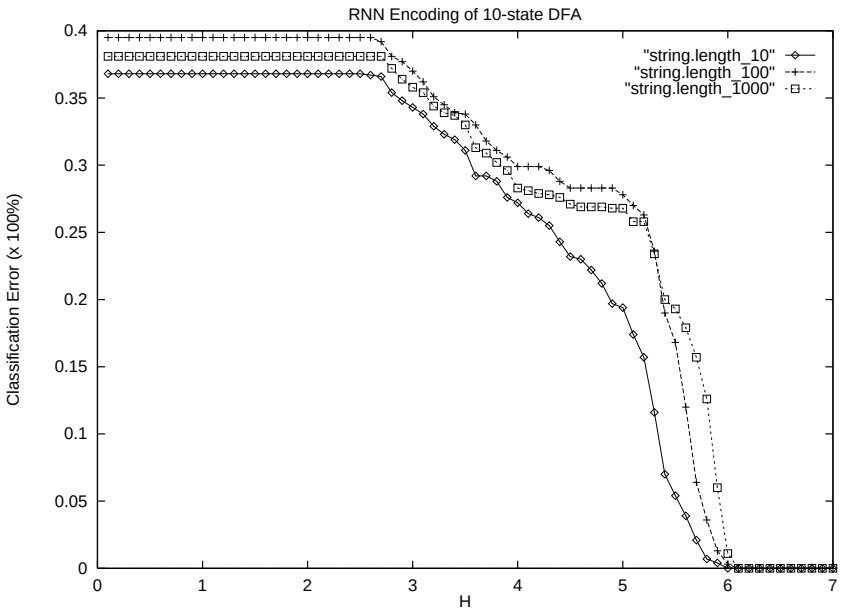

Figure 4: **Performance of 10-state DFA:** The network classification performance on three randomly-generated data sets consisting of 1,000 strings of length 10 ($\diamond$), 100 ($+$), and 1,000 ($\Box$), respectively, as a function of the rule strength H (in 0.1 increments) is shown. The network achieves perfect classification on the strings of length 1,000 for $H > 6.0$.

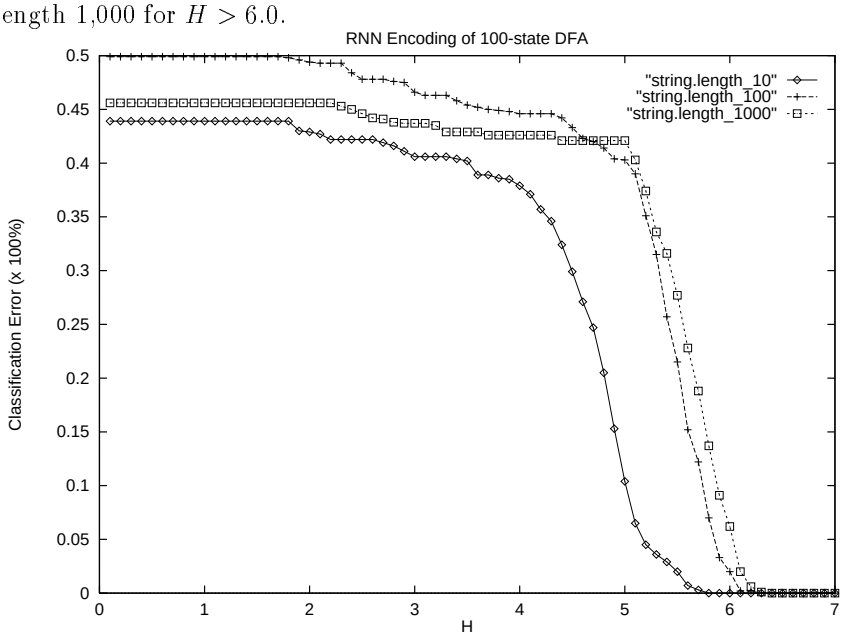

Figure 5: **Performance of 100-state DFA:** The network classification performance on three randomly-generated data sets consisting of 1,000 strings of length 10 ($\diamond$), 100 ($+$), and 1,000 ($\Box$), respectively, as a function of the rule strength H (in 0.1 increments) is shown. The network achieves perfect classification on the strings of length 1,000 for $H > 6.2$.

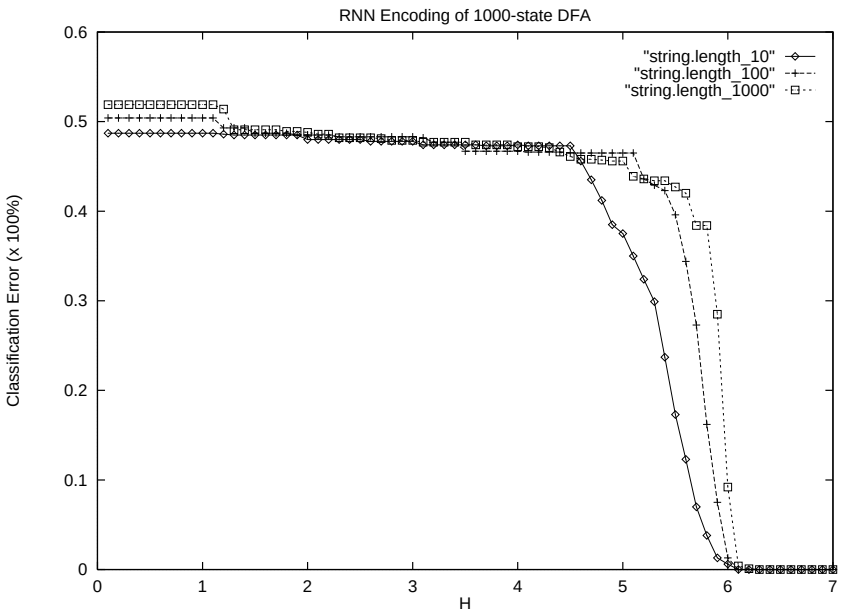

Figure 6: **Performance of 1000-state DFA:** The network classification performances on three randomly-generated data sets consisting of 1,000 strings of length 10 ($\diamond$), 100 ($+$), and 1,000 ($\Box$), respectively, as a function of the rule strength H (in 0.1 increments). The network achieves perfect classification on the strings of length 1,000 for $H > 6.1$.

5 SCALING ISSUES

5.1 Preliminaries

The worst case analysis in section 3 supports 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, network stability is achieved for values of H that are smaller than the values required by the conditions in theorem 3.3.1,

(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 in the above experiments for DFAs of different sizes, we find that $H \approx 6$ 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 $H = 6.3$. This value corresponds to a DFA whose states have 'average' indegree $r = 1.5$. [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

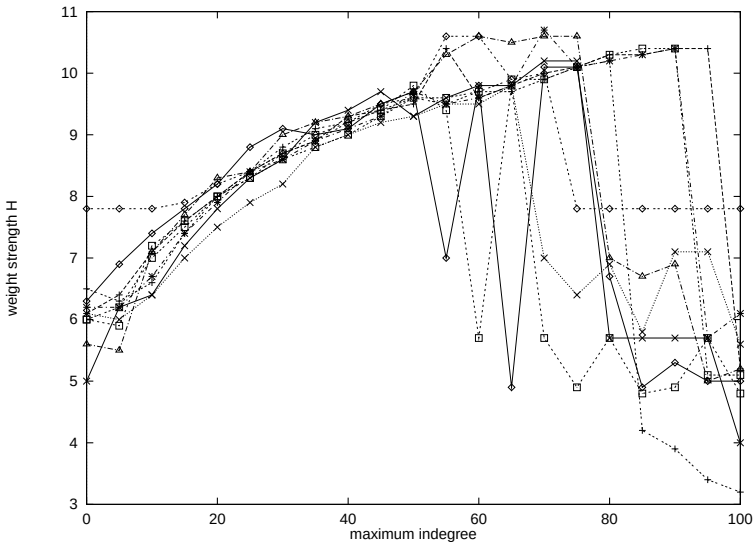

Figure 7: **Scaling Weight Strength**: An accepting state q_ρ in 10 randomly generated 100-state DFAs was selected. The number of states q_j for which $\delta(q_j, 0) = q_\rho$ was gradually increased in increments of 5% of all DFA states. The graph shows the minimum value of H for correct classification of 100 strings of length 100. H increases up to $\rho = 75\%$; for $\rho > 75\%$, the DFA becomes degenerated causing H to decrease again.

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.

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 the fraction 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 7 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 generally increases with increasing values of ρ; in all cases, the hint strength H 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

$$-\frac{H}{2} + H * S_\rho^t + \sum_{q_l \in F} S_l^t * H - \sum_{q_l \notin F} S_l^t * H \tag{23}$$

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 $n \rightarrow \infty$ 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 $p \rightarrow 0$ 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 $n \rightarrow \infty$, the languages accepted by the DFA M and the constructed neural RNN are identical only for $H \rightarrow \infty$.*

Proof: Recall the worst case equations (11) and (12) 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 apply here also; however, the residual inputs are zero. Thus the following two conditions for stable low and high signals, respectively, must be satisfied:

$$-\frac{H}{2} + H * \phi_{\Delta}^{-} < \frac{1}{2} \tag{24}$$

and

$$-\frac{H}{2} + H * \phi_{\Delta}^{+} - H * \phi_{\Delta}^{-} > \frac{1}{2} \tag{25}$$

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:

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

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

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 $H \to \infty$; however, correct string classification requires $\phi_{\Delta}^{-} \to 0$ and thus $H \to \infty$ for $n \to \infty$.

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 $H \approx 6$ is independent of the network size for typical DFAs. Extreme DFAs can be constructed for the weight strength scales with the network size.

The stability analysis presented in this paper can be extended to the case where DFAs are embedded into fully recurrent networks. This may be desirable in case where a network is initialized with partial prior knowledge to be refined through learning on training data. In that case, the weights which are not programmed to $-H, -H/2$, and $+H$ are initialized to small random values drawn according to some distribution from an interval $[-W, W]$. Then, the significance of stable DFA encoding is that the parameters H and W can be chosen such that knowledge is not destroyed by the presence of the randomly initialized weights.

It would be an interesting question to investigate whether a denser binary representation of DFA states is possible, thus requiring fewer than $n + 1$ state neurons to encode a DFA with n states in a second-order recurrent neural network. We hypothesize that dense neural DFA construction is a NP-complete problem [Hopcroft and Ullman, 1979].

7 ACKNOWLEDGMENT

We would like to acknowledge useful discussions with B.G. Horne, L. R. Leerink, and T. Lin. We would especially like to acknowledge helpful suggestions from the reviewer.

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.

[Alquézar and Sanfeliu, 1995] Alquézar, R. and Sanfeliu, A. (1995). An algebraic framework to represent finite-state machines in single-layer recurrent neural networks. *Neural Computation*. To appear.

[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, Florence, Italy.

[Hopcroft and Ullman, 1979] Hopcroft, J. and Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, Inc., Reading, MA.

[Horne and Hush, 1994] Horne, B. and Hush, D. (1994). Bounds on the complexity of recurrent neural network implementations of finite state machines. In Cowen, J., Tesauro, G., and Alspector, J., editors, *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 and Giles, 1994] Omlin, C. and Giles, C. (1994). Constructing deterministic finite-state automata in sparse recurrent neural networks. Technical Report 94-3, Computer Science Department, Rensselaer Polytechnic Institute.

[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). Private 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.