[22] Selverston, A.I., and Moulins, M., "Oscillatory neural networks," *Ann. Rev. Physiol.*, Vol. 47, (1985), pp. 29-48.

[23] Siegelmann, H.T., Sontag, E.D., "On the Computational Power of Neural Networks," *Report SYCON-91-11, Rutgers Center for Systems and Control*, (1991).

[24] Wang X., "Period Doublings to Chaos in a Simple Neural Network: An Analytical Proof," *Complex Systems*, Vol. 5(4), (1991) 425-441.

[25] Watrous, R.L. and Kuhn, G.M., "Induction of Finite-State Languages Using Second-Order Recurrent Networks," *Neural Computation*, Vol. 4 (1992), 406-414.

[26] Wiggins, S., *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, Springer-Verlag, New York, 1990.

[27] Williams, R.J. and Zipser, D., "A Learning Algorithm for Continually Running Fully Recurrent Networks," *Neural Computation*, Vol. 1 (1989), 270-280.

Information Processing Systems 2, D.S. Touretzky (ed), Morgan Kaufmann, San Mateo, CA, p. 380, 1990.

[10] Hertz J., Krough A., and Palmer R.G., *Introduction to The Theory of Neural Computation*, Addison-Wesley Publishing Company, Menlo Park, California, 1991.

[11] Kolen, J.F., "Fool's Gold: Extracting Finite State Machines From Recurrent Network Dynamics," *Advances in Neural Information Processing Systems 6*, J.D. Cowan, G. Tesauro, J Alspector (eds.), Morgan Kaufmann, San Mateo, CA, (1994), p. 501.

[12] Kristan, W.B.,Jr., "Generation of rhythmic motor patterns," *Information Processing in the Nervous System*, (1980) , 241-261.

[13] Lewis, R.L., Papadimitriou, C.H., *Elements of the Theory of Computation*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[14] McCulloch, W.P., and Pitts, W., "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bullentin of Mathematical Biophysics*, Vol. 5 (1943), pp. 115-133.

[15] Milnor J., "On the Concept of Attractor," *Communications in Mathematical Physics*, Vol. 99 (1985), pp. 177-195.

[16] Munkres, J.R., *Topology: a first course*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[17] Narendra, K.S. and Parthasarathy, K., "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, Vol. 1, (1990), pp. 4-27.

[18] Omlin, C., and Giles, C., "Constructing Deterministic Finite-State Automata in Sparse Recurrent Neural Networks," in *IEEE International Conference on Neural Networks (ICNN'94)*, (1994), pp. 1732-1737.

[19] Pollack J.B., "The Induction of Dynamic Recognizers," *Machine Learning*, Vol. 7 (1991), pp. 227-252.

[20] Tsung, F-S., Cottrell, G.W. and Selverston, A.I., "Some experiments on learning stable network oscillations," *International Joint Conference on Neural Networks, San Diego, IEEE*, 1990.

[21] Tsung, F-S. and Cottrell, G.W., "Learning Temporal Signals in Phase Space," *International Symposium on Nonlinear Theory and Its Application, Hawaii*, (1993).

Bibliography

[1] Casey, M., "Computation Dynamics in Discrete-Time Recurrent Neural Networks," in UCSD's Institute for Neural Computation's *Proceedings of the Annual Research Symposium*, June (1993), (pp 78-95).

[2] Casey, M., "How Recurrent Neural Networks Work," Tech. Report UCSD-MATH-95-1, University of California, San Diego, La Jolla, March 1995.

[3] Casey, M., "Relaxing The Symmetric Weight Condition for Convergent Dynamics in in Discrete-Time Recurrent Networks," Tech. Report UCSD-MATH-95-2, University of California, San Diego, La Jolla, March 1995.

[4] Cleermans, A., McClelland, J.L, and Servan-Schreiber, D., (1989) "Encoding Sequential Structure in Simple Recurrent Networks," Tech. Rept. CMU-CS-88-183, Carnegie-Mellon University, Pittsburgh, November 1988.

[5] Cummins, F., "Representation of Temporal Patterns in Recurrent Neural Networks," Preprint, submitted to *The* 15th *Annual Conference of the Cognitive Science Society.* (1993).

[6] Devaney, R.L., *An Introduction to Chaotic Dynamical Systems*, Addison-Wesley Publishing Company, Inc., New York, 1987.

[7] Elman, J.L., "Representation and Structure in Connectionist Models," Tech. Rept. 8903, Center for Research in Language, University of California, San Diego, La Jolla, CA, August 1989.

[8] Giles, C.L., Miller, C.B., Chen, D., Chen, H.H., Sun, G.Z., Lee, Y.C., "Learning and Extracting Finite State Automata with Second-Order Recurrent Neural Networks," *Neural Computation*, Vol. 4 (1992), 393-405.

[9] Giles, C.L., Sun, G.Z., Chen, H.H., Lee, Y.C., Chen, D., "Higher Order Recurrent Networks and Grammatical Inference," *Advances in Neural*

DFA diagram) that enables one to infer internal dynamics from studying the representation, we should be able to predict the organization of nervous system behaviors (on some level) by understanding the mapping from nervous system behavior to organism behavior, and by understanding the logical structure of the problem to be solved (nontrivial even in the case of DFA). One is then in a better position to measure nervous systems in order to find those aspects of nervous system behavior that are essential for computation as opposed to those essential for biological maintenance or some other function. The fact that more than one pattern of nervous system behavior may produce the same pattern of organism behavior (Kristan (1980) [12] and Selverston and Moulins (1985) [22] are examples) suggests that even with the added insight of knowing (on some level) what we seek, it is likely to still be nontrivial to map that onto actual nervous system functioning that is amenable to measurement. But there is always hope that the nervous system has a small number of standard "tricks" or methods for solving the problems which it faces.

Chapter VIII

Conclusions

We now have an outline for using the cognitive science approach to understanding the workings of $\mathcal{C}$-systems. First, we gain a clear understanding of the problem to be solved by casting it in the form of a language recognition problem and by finding the minimal DFA that recognizes the language. (Of course, this is a very difficult problem in general.) The fact that many problems that people have trained RNNs to solve can be so described (a notable exception being general pattern production, see Hertz et al. (1991) [10] pp. 63–67) suggests that minimal DFA are a good representation of many problems of interest to connectionists. Second, one analyzes the DFA in a manner that is suited to making connections between the problem and the dynamics of a family of dynamical systems that can model the DFA. Next, one studies the mapping between hidden unit activations and the production of the desired output (trivial for the RNNs studied here, but more complex when one moves to Elman networks, for example). Finally, one measures the RNN to make sure that it has the dynamics necessary for the computation.

We feel that the restriction our problem domain to modeling DFA does not severely limit the applicability of our theory. Many of the problems which RNNs have solved can be recast as DFA problems. For example, learning Reber grammars as in Cleermans *et al.* (1989) [4], and many of the problems in Williams and Zipser (1989) [27] are essentially DFA problems. Furthermore, it is not difficult to see, following the work of Siegelmann and Sontag (1991) [23], that DFA are the only class of computational models that can be robustly modeled by unaugmented RNNs. Another class of problems that RNNs can solve is pattern production, which has been studied by Tsung, Cottrell and others (Williams and Zipser (1989) [27], Tsung, Cottrell and Selverston (1990) [20], Tsung and Cottrell (1993) [21], Narendra and Parthasarathy (1990) [17]).

The difficulty of learning a DFA from examples of desired behavior (see Giles *et al.* (1992) [8]) serves as evidence that careful studies of behavior and useful classification of problems solved by organisms is a nontrivial aspect of the analogous approach to understanding computation in nervous systems. Once we find a task representation of the tasks performed by "intelligent" organisms (the analogue of the

```
****************************  gram.c  ***************************

***************************  end gram.c  *************************
```

Chapter VII

Simulations

Simulation can be a very important step in developing the mathematics necessary for understanding continuous state space systems modeling discrete computation. Simulation is useful for finding regularities which can then be explored analytically, and in the course of developing the work in this thesis, we developed many programs to explore various aspects of dynamical systems and $\mathcal{C}$-system.

This program is called **gram**, and is used to train RNNs to model given DFA models. **gram** uses the files called *params* and *dfa1*, which will be included before the listing for **gram.c**.

These programs have been taken out of this version of my thesis in an attempt to save paper. Please e-mail the author with requests to view the code.

VII.A Example Params File

```
*****************************  params  *****************************

*************************  end params  *************************
```

VII.B Example DFA File

```
*****************************  dfa1  *****************************

*************************  end dfa1  *************************
```

VII.C Gram.c Code

also might want to consider more general invariant sets than attractors, and, again, a dense orbit is sufficient for this result. ($x_{t+1} = 4x_t(1 - x_t)$, for example, does not have a chaotic attracting set, but rather has only a chaotic invariant set.)

The examples which Kolen gives in Kolen (1994) [11] are all maps with dense orbits in their invariant sets, and hence if they were a part of an RNN modeling a DFA, the invariant sets would correspond to loops in the DFA diagram. Sensitivity to initial conditions itself does not pose any problem for DFA extraction. If the reachable points in a partition state are mapped to more than one other partition state and the RNN models some DFA, then the target partition states must be equivalent.

John Kolen (1994) [11] has recently raised questions concerning finite state machine (FSM) extraction in RNNs which have chaotic dynamics. The problem which has been suggested is that extraction systems will fail for systems displaying sensitive dependence on initial conditions (Devaney (1987) [6]) since such systems can be interpreted as either a nondeterministic FSM with few states or a deterministic FSM with a large number of states. He further argues that no matter how small one partitions the state space, sensitivity to initial conditions will eventually force the extracted state to split into multiple trajectories independent of the future input sequence, which suggests a nondeterministic interpretation of the state.

One way to resolve this problem is to require that the extracted FSM be deterministic. In this case, for any behavior that is a part of the computation [1], will correspond to a loop in the minimal DFA diagram, so there is no problem with sensitivity to initial conditions causing states to split. We will need another definition before formally stating and proving this fact.

Definition VI.1 *A point in the RNN phase space is said to be reachable if it is in the image of the f_b, $f_b(\mathcal{I}^N)$, or in $f_I(f_b(\mathcal{I}^N))$ for some $I \in \{0,1\}^*$. A behavior is called reachable if any of its points are reachable.*

Theorem VI.2 *Suppose that a RNN robustly models a DFA, $\mathcal{M}$, and one of its f_I has a behavior which is reachable, then the behavior corresponds to a cycle in $\mathcal{M}$'s minimal state diagram.*

In order for the single attractor to represent more than one state, its f_I would have to satisfy the condition that $(f_I)^n$ has n distinct attractors for some $n > 1$, as in the case of an attracting periodic orbit or an attractor where each point in an attracting periodic orbit bifurcates into an invariant circle with a dense orbit (which one would find after a Hopf bifurcation of the nth composition of the map with itself).

Proof: If the behavior is reachable, then there is some point, x, in the behavior which is in $f_S(f_b(\mathcal{I}^N))$ for some string S. It follows that x is in Q_i for some i, else the RNN will not model $\mathcal{M}$. Suppose further that there is no number of I strings which $\mathcal{M}$ can read in order to return to q_i (i.e. q_i is not in an n-cycle for the f_I). But, since x is in a behavior of the f_I, it is ϵ-pseudoperiodic for f_I, and hence there is an m and a pseudo-orbit of x such that the m_{th} iterate of x is x, and hence in Q_i. It follows that if the RNN models $\mathcal{M}$ and $\mathcal{M}$ starts in state q_i, then after $\mathcal{M}$ reads m copies of I, $\mathcal{M}$ will have returned to q_i. It follows that q_i is in an m-cycle of $\mathcal{M}$'s minimal diagram. ■

The robustness assumption is not essential to the argument. It is sufficient that the attractor have a dense orbit. If we are not requiring robustness, then we

[1] since the Q_i will not fill the state space when there are more than one of them, it is possible for the maps to have behaviors in the complement of the union of the Q_is which have nothing to do with the computation being performed

partitions created by dividing each unit's range into q partitions of equal length to always succeed in extracting the DFA that is being modeled.

This may be a nontrivial statement given that the basins of dynamical systems can have very complicated fractal structure.

For the proof we need the following lemma:

Lemma VI.1 *Let U and V be disjoint compact subsets of $\mathcal{I}^N$, then if we divide $\mathcal{I}^N$ into smaller N-dimensional cubes, called partition states, having sides of length 2^m, there exists a finite m such that none of the partition states contain points in both U and V.*

Proof: Consider a sequence of points generated by increasing the partition parameter by powers of two and then choosing one point in U from any partition state that contains points in both U and V. If we can define an infinite sequence this way, then since U is compact our sequence will have an accumulation point. But, since the distance from points in the sequence to points in V goes to zero by the fact that they are chosen from partition states of vanishing size, that accumulation point is also an accumulation point of V, contradicting the assumption that U and V are disjoint. So we have that there are only a finite number of partition parameters which yield partition states containing points in both U and V. ■

We now return to proving VI.1.

Proof: Theorem IV.1 says that there are a finite number of mutually disjoint, closed (hence compact) subsets of $\mathcal{I}^N$, Q_i, that correspond to the states of the DFA. By lemma VI.1 we can separate any of the Q_i from the others since their union is a compact set if the the partion parameter is a power of two greater than or equal to some 2^{N_i}. Hence, if we choose the largest of these N_i then no partition state containing points in one of the Q_i will contain points in any of the others, and therefore they will be separated from one another. ■

Theorem IV.1 then implies that for a sufficiently large partition parameter the partition states contained in some Q_i will not be equivalent to the partition states in any Q_j for i not equal to j.

To see that the second of the conjectures in Giles *et al.* (1992) [8], that the extracted minimal DFA will be independent of the choice of the partition parameter for well-trained $\mathcal{C}$-systems, is false, choose a DFA with n states, for $n > 2$, that can be modeled by a one-dimensional $\mathcal{C}$-system. In order to extract the correct DFA, the partition must separate the Q_i in the $\mathcal{C}$-system phase space. Hence, ther second conjecture is false since we can choose $q = 2$. We replace this second conjecture with the conjecture that there may be only a finite number of partition parameters that yield different minimal DFA for a $\mathcal{C}$-system robustly modeling a DFA, in which case their conjecture is true for q sufficiently large.

Chapter VI

DFA Extraction

Our theory provides a theoretical foundation for studying DFA extraction as defined in Giles *et al.* (1992) [8], and in particular, allows us to study two of their conjectures. The first of their conjectures is that during training the $\mathcal{C}$-system begins to partition its state space into fairly well separated, distinct regions or clusters which represent corresponding states in some finite state automaton. Another of their conjectures is that for what they call "well-trained" $\mathcal{C}$-systems, the extracted minimal DFA are independent of the choice of partition parameter. Note that one must have a theory such as ours, that allows for an understanding of all of the infinitely many possible $\mathcal{C}$-system solutions to a given language recognition task in order to productively study these conjectures.

Giles *et al.* (1992) [8] define DFA extraction as a four part procedure: (1) cluster DFA states, (2) construct a transition diagram by connecting these states with the alphabet labeled arcs, (3) combine transitions to form the full digraph loops, and (4) reduce the digraph to a minimal representation. They partition the state space by dividing each unit's range into q partitions of equal width.

The first important step in discussing DFA extraction is to note that there are examples of $\mathcal{C}$-system that robustly model DFA and values of q for which the extraction procedure described in Giles *et al.* (1992) [8] will not produce a DFA, let alone one that reduces to the unique minimal DFA that the $\mathcal{C}$-system models. The problem with their procedure is that one of the $\mathcal{C}$-system's f_i may send points in one partition state to more than one other partition state and, when this happens, the corresponding DFA digraph will not define a function of the partition states to themselves. DFA extraction will work once we modify the process to correct the problem of having points in one partition state being sent to more than one partition state. To correct it we make all partition states that are in the range of one partition state equivalent. We need to check that this equivalence is well-defined, which follows from the following theorem:

Theorem VI.1 *For $\mathcal{C}$-systems that robustly model a given DFA, for sufficiently low levels of noise in the model and sufficiently large q, it is sufficient to consider only*

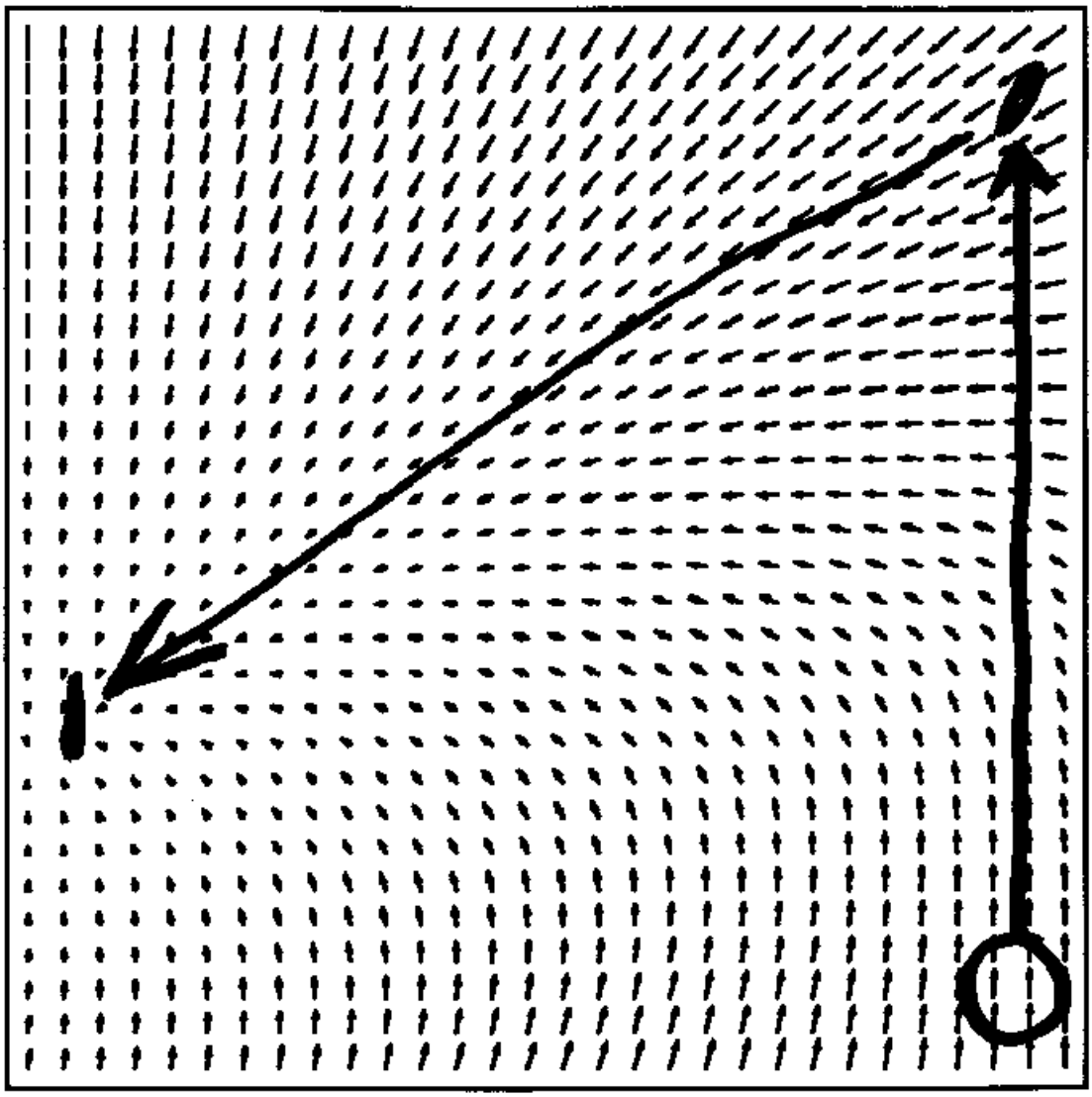

Figure 6: (C) Phase space dynamics of f_1.

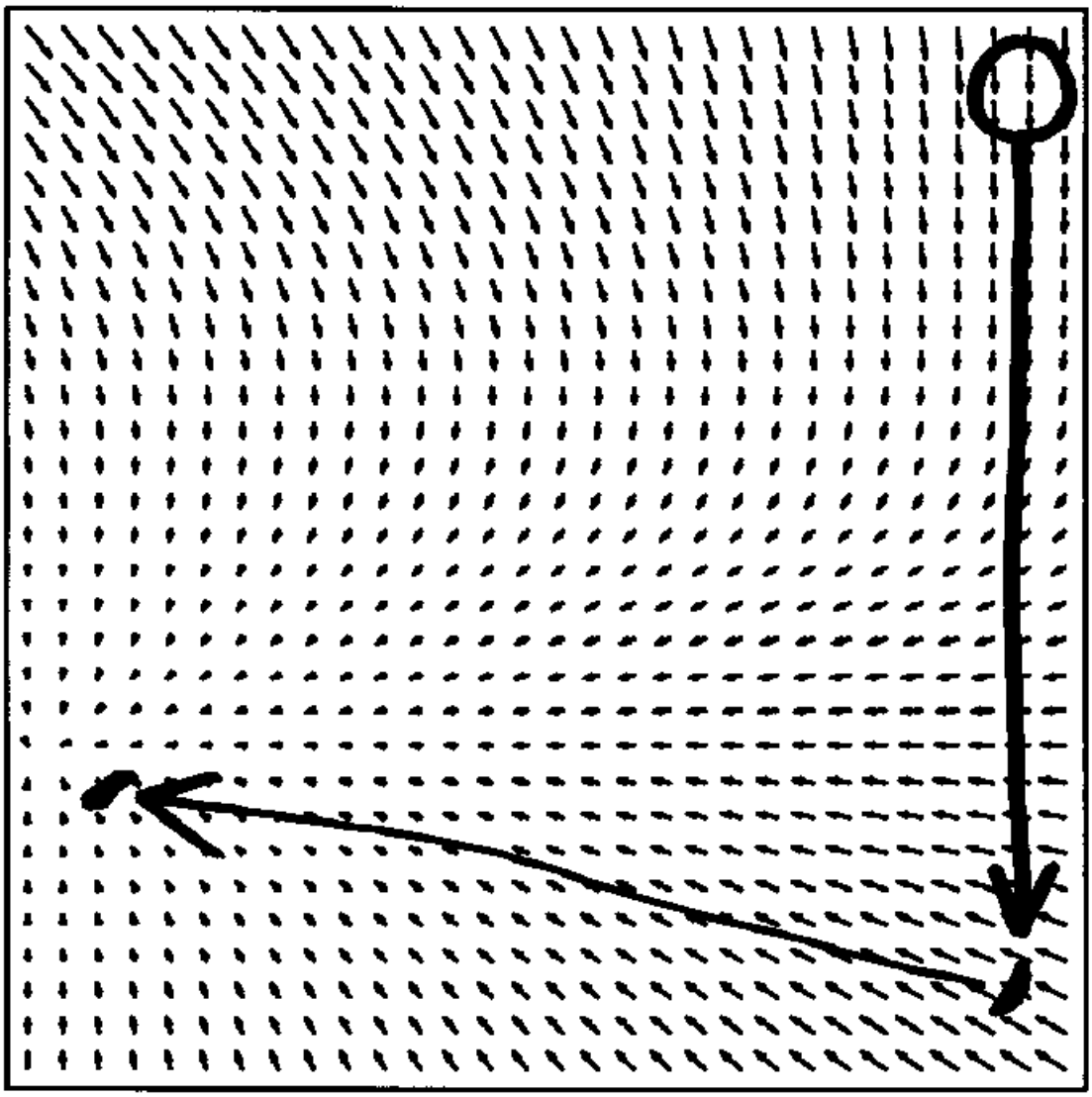

Figure 6: (B) Phase space dynamics of f_0.

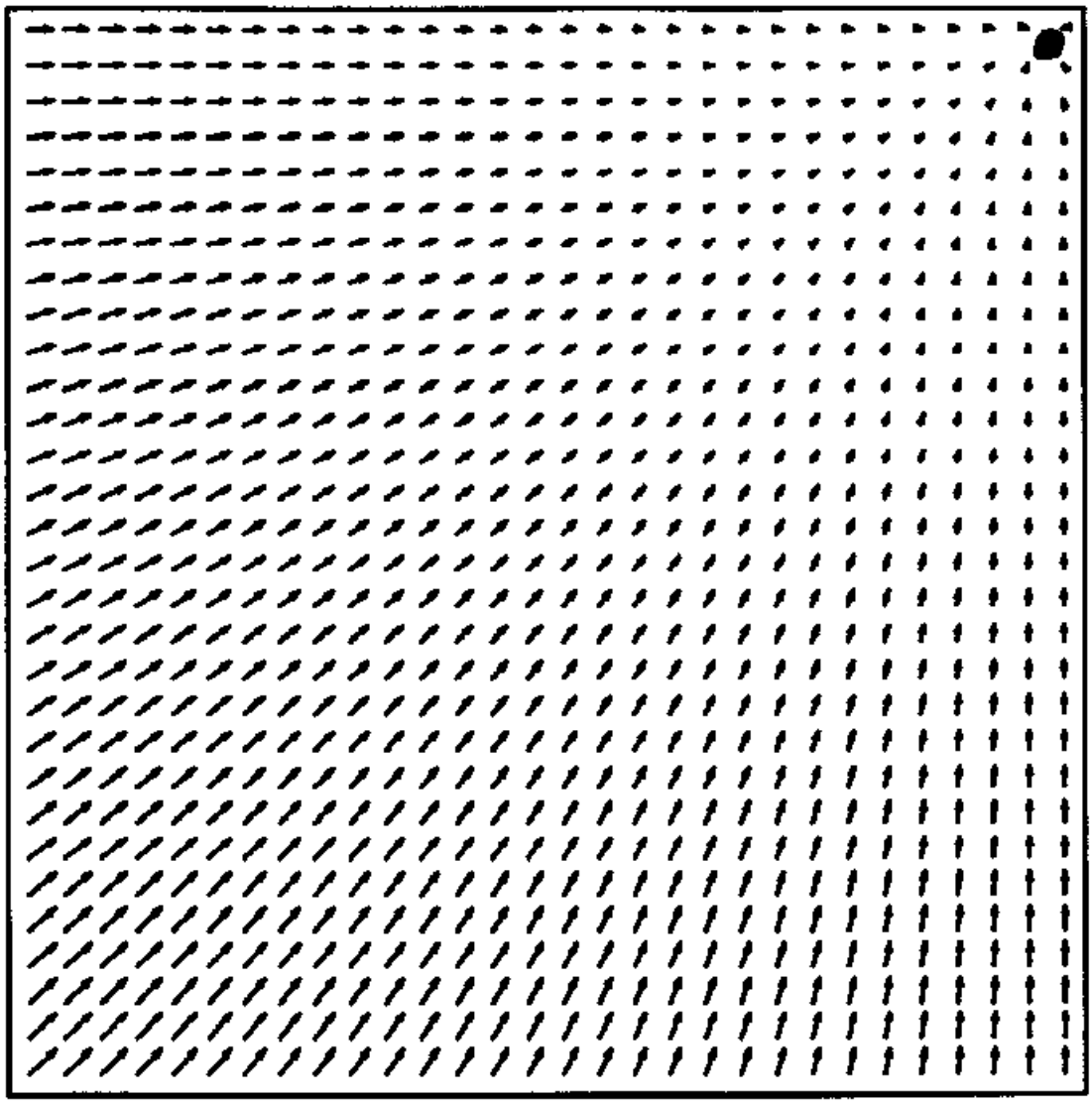

Figure 6: (A) Phase space dynamics of f_b of a two-dimensional $\mathcal{C}$-system that recognizes the language L_3.

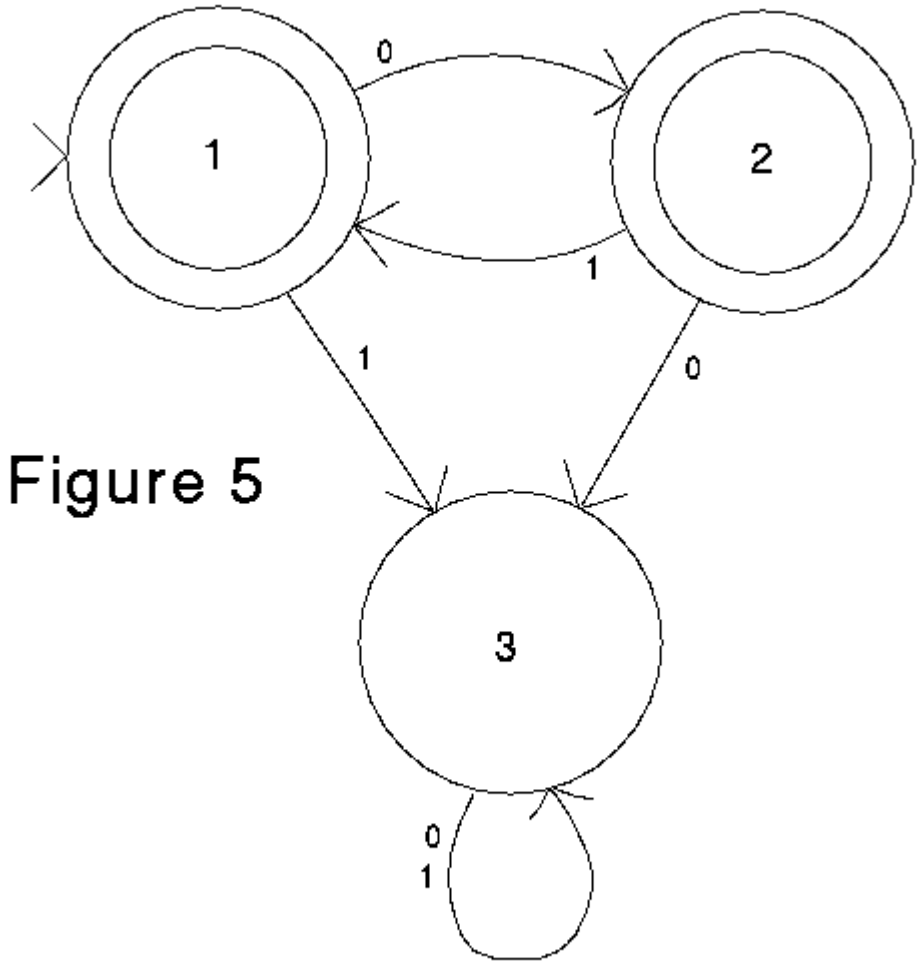

Figure 5

Figure 5: The minimal DFA which recognizes the language containing all words with no 00 or 11 substrings that begin with 0.

one iteration of f_1. f_0 has only one fixed state, and its corresponding fixed point is in the reject region of phase space. To recognize the language then, it must use transient computation. Points near the bottom right side of the phase space iterate to points in the upper right side of the phase space (still in the accept region), which are then mapped to the reject region (hence we correctly reject words containing two consecutive zeros). This is shown in the figure by showing the image of a circle initially in the region of phase space to which f_b sends all states. f_1 complements the dynamics of f_0 so that the composition of the two, f_{01}, will have a fake period two orbit in the accept region, since points on the upper right side of the phase space are mapped to points in the lower right side. Note that if one considers f_{01} to be a dynamical system and iterates it like any other, (i.e. without watching the state variables after applying only f_0) then a fake periodic orbit can be a fixed point, as it is in this case.

It is possible to recognize L_2 with a one-dimensional $\mathcal{C}$-system with activation functions more complicated than sigmoids. If, however, one attempted to use teacher-forcing during training, the problem would not be solvable in one dimension. Teacher-forcing would restrict the $\mathcal{C}$-system to mapping between two points (one representing the fact that the DFA is in a nonfinal state, and the other representing a final state), and therefore it would be necessary for f_1 to map the nonfinal state point to itself and also to the final state point, which is impossible. If we move to a two-dimensional $\mathcal{C}$-system then this problem with teacher-forcing disappears.

V.C Example 3

Our final example language, L_3, is the set of all strings that begin with 0 and have no 00 or 11 substrings. Figure 5 shows the minimal DFA for L_3, where states q_1 and q_2 are transient states and q_3 is a fixed state of both f_0 and f_1. We also see that $\{q_1, q_2\}$ and $\{q_3, q_3\}$ are fake periodic orbits for f_{01}. As noted in Giles *et al.* (1992) [8] and Watrous and Kuhn (1992) [25], in order to check a $\mathcal{C}$-system's ability to generalize to unseen strings, it is important to check the dynamic stability of the $\mathcal{C}$-system. Our theory allows for a much more precise and general determination of this stability. While Giles *et al.* (1992) [8] equate generalization to long unseen strings with fixed point behavior, our theory shows that more general attractors may be used for the same purpose. More importantly, since Giles *et al.* (1992) [8] and Watrous and Kuhn (1992) [25] check for fixed point behavior by simply reading a long previously unseen string (Watrous and Kuhn (1992) [25] use strings of length 100), they falsely assume that if the $\mathcal{C}$-system produces the correct output for a string of length N for N sufficiently large, it will produce the correct output for strings of arbitrary length. Our theory allows for a complete understanding of the $\mathcal{C}$-system's ability to generalize to strings of arbitrary length by determining the properties that the individual f_i and compositions of these maps must have to ensure that the $\mathcal{C}$-system will recognize the entire language rather than simply recognizing some finite subset of the language. Furthermore, our theory makes it clear that one needs to check the stability of fake periodic orbits, which is not obvious when looking at the grammar description.

As noted in Giles *et al.* (1992) [8] and Watrous and Kuhn (1992) [25], the lack of stability of fixed states is likely to cause errors in performance when the $\mathcal{C}$-system is checked on long, unseen strings. Lack of stability is an even more likely source of error for periodic orbits and fake periodic orbits since this lack of dynamic stability is less likely to cause errors while checking $\mathcal{C}$-system performance in this way. The reason they cause difficulty is that iterated substrings of length greater than one are less likely to appear in both training and testing data sets. For a $\mathcal{C}$-system recognizing L_3, we must check that f_{01} is hysteretic, since the minimal state DFA for L_3 has two fake period orbits.

Again, we study a two-dimensional solution. f_b puts the $\mathcal{C}$-system in the part of the phase space that will go to the reject region after two iterations of f_0 or

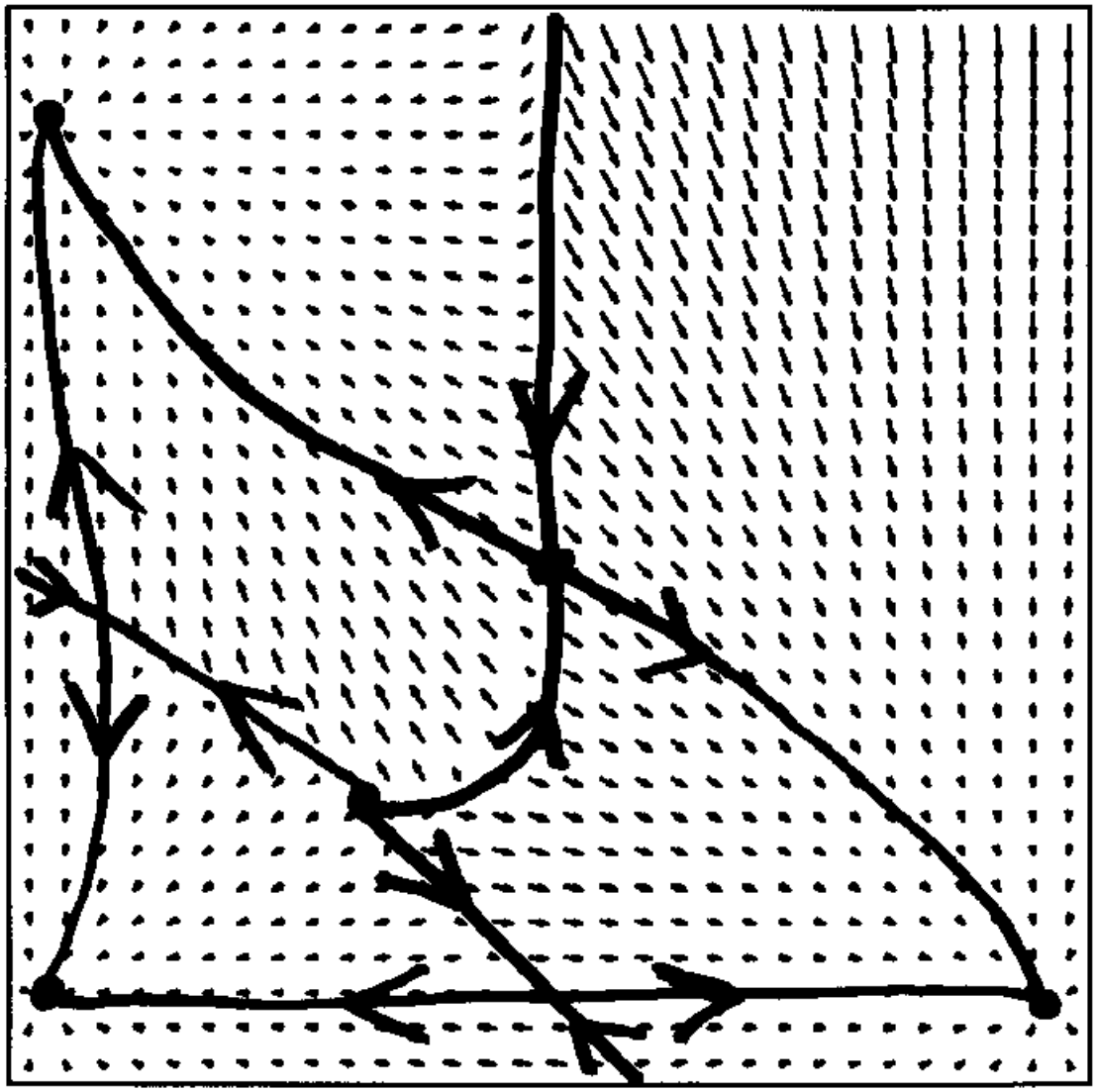

Figure 4: (C) Phase space dynamics of $(f_1)^3$.

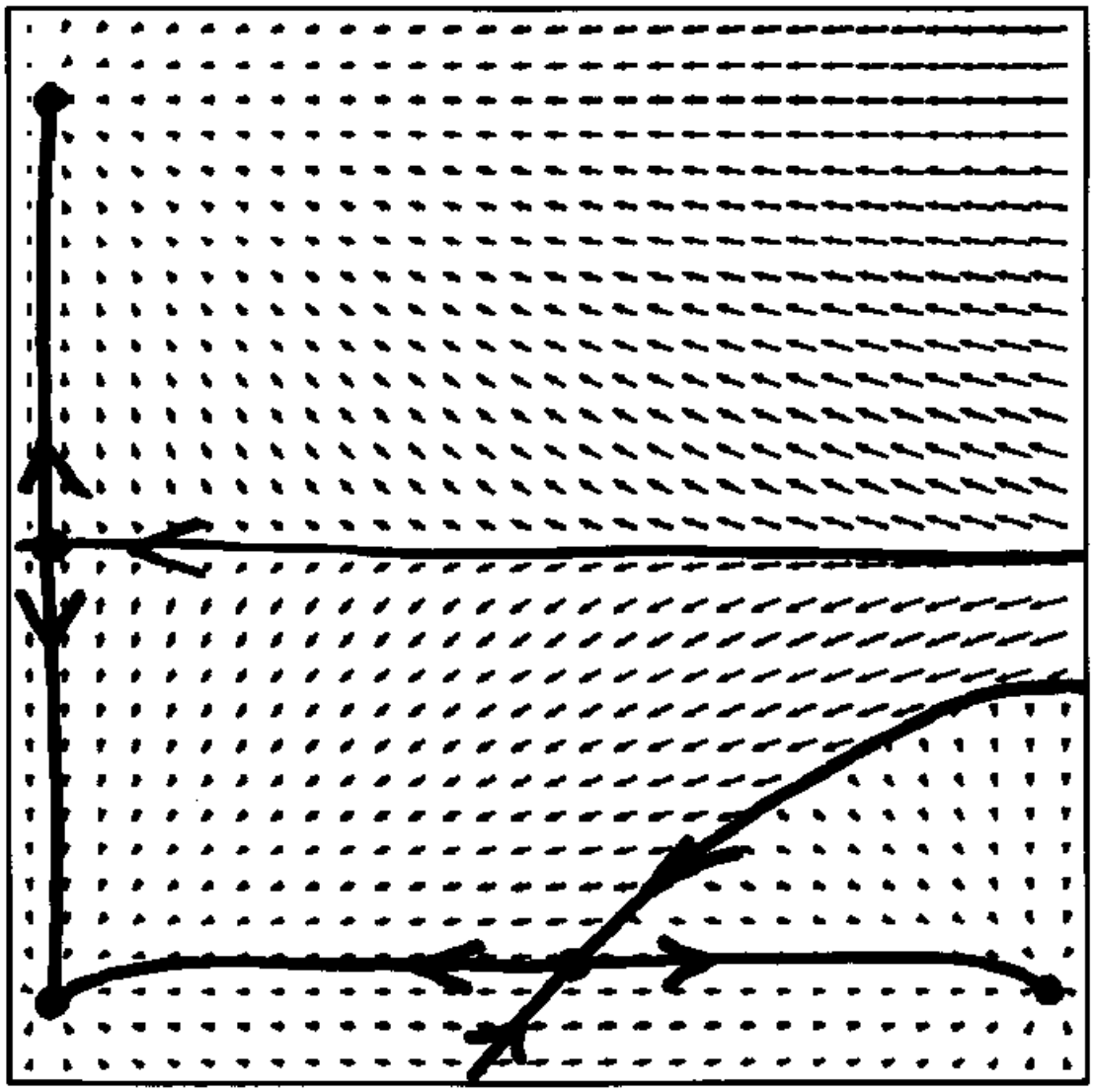

Figure 4: (B) Phase space dynamics of f_0.

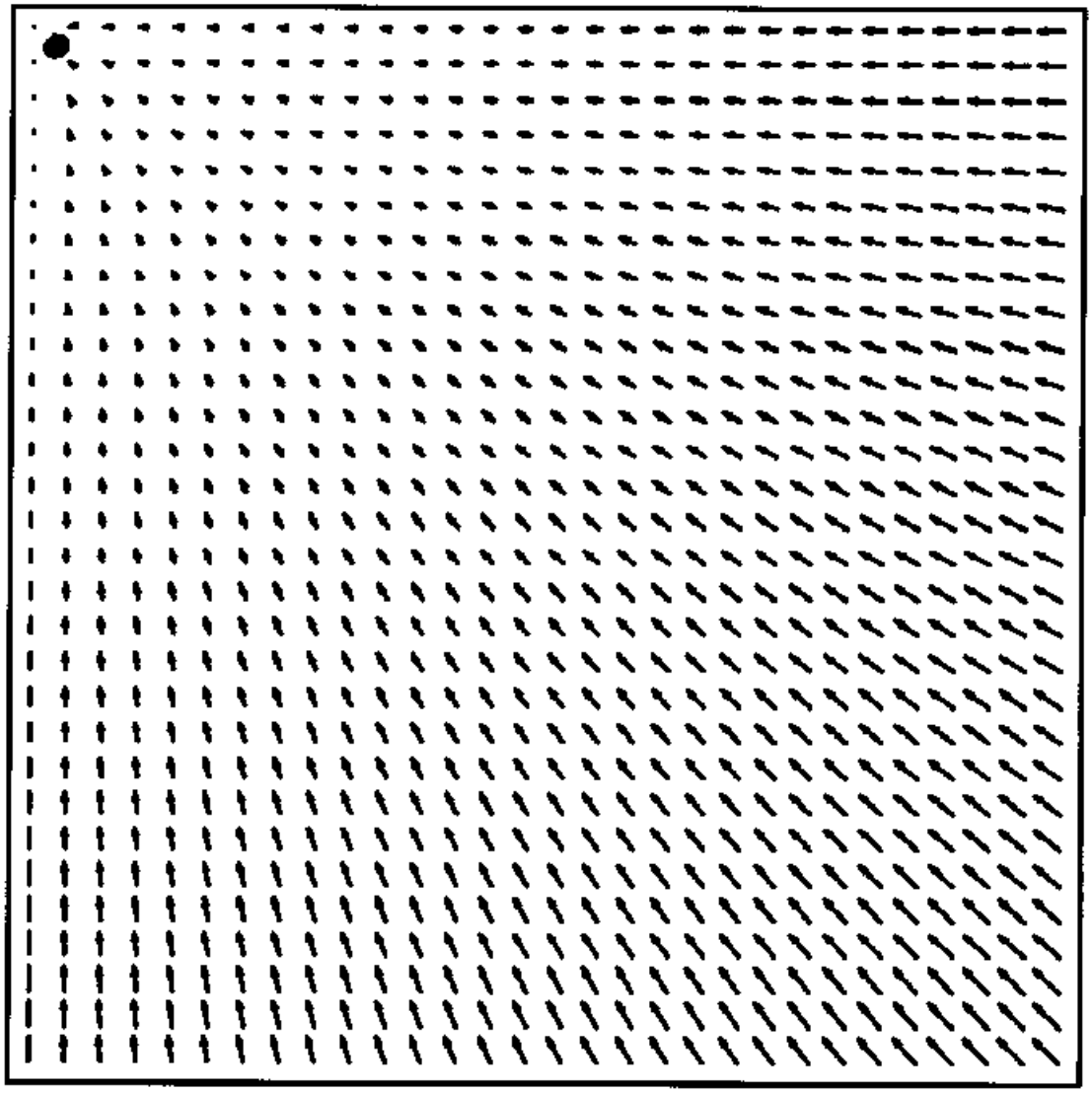

Figure 4: (A) Phase space dynamics of f_b of the $\mathcal{C}$-system that recognizes the language accepted by the automata in Figure 3.

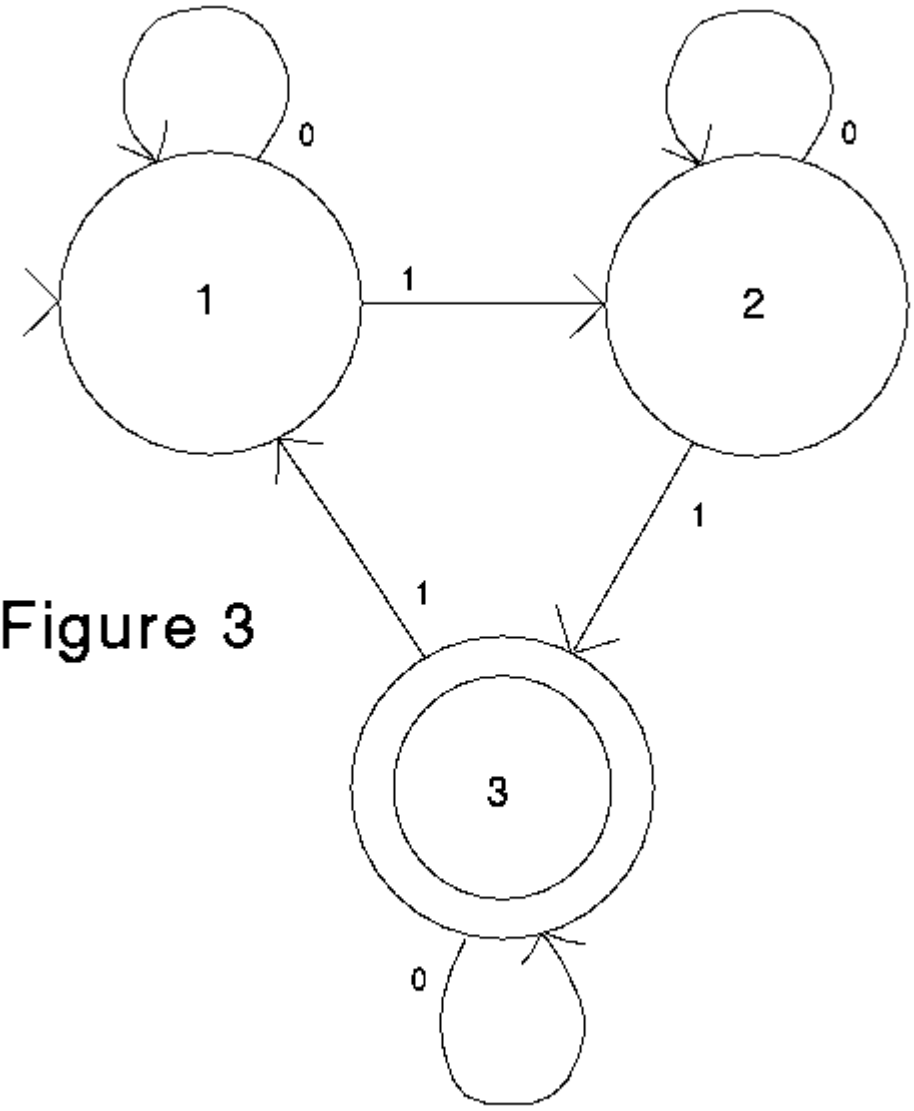

Figure 3

Figure 3: The minimal DFA which recognizes the language containing all words in {0,1}* which contain $3k + 2$ 1's for some k $\geq$ 0.

that such a solution will be found while evolving the $\mathcal{C}$-system. It is also clear from this example that Theorem IV.2 does not hold for $\mathcal{C}$-systems which simply model rather than robustly model a DFA.

One can use the intermediate value theorem to prove that a one-dimensional RNN with a sigmoidal discriminant function (as a one-dimensional monotone dynamical system) cannot have these dynamics, so we know that the two-dimensional RNN solution shown in the figures is minimal with respect to dimension. We emphasize here that the solution that the $\mathcal{C}$-systemfound could have had complexities that the simple prediction did not consider. Besides the possibility of having many behaviors in the same class, it is also possible that rather than using fixed points, the $\mathcal{C}$-systemcould have had "small" chaotic attractors (small in the sense that they stay in appropriate future basins of attraction), that might lead one to believe that those dynamics are somehow necessary for the computation. So not only might there be many internal states corresponding to the same computational state, but any of those individual behaviors might also be complicated. What might be true when we observe more complicated dynamics is that, while they were not necessary from a computational perspective, they were necessary for topological reasons from the perspective of the map dynamics, or when observing nervous system behavior that they were necessary for biological reasons.

matrices this is an unrealistic hope in general. For nonlinear systems it is even more rare that one can gain insight into the dynamics from observing the parameters, even after considerable effort to determine dependencies between parameters and dynamics, as in Casey (1995) [3] or Wang (1991) [24].

V.B Example 2

A problem that uses only hysteretic computation is recognizing the language, L_2, containing the words with $3k+2$ 1's for some $k \geq 0$. Figure 3 shows the minimal DFA for this language. f_0 for this $\mathcal{C}$-system is hysteretic. The DFA's states are a period three state orbit of f_1. Hence, to model this DFA, a $\mathcal{C}$-system might use a f_0 with three attracting fixed points, and a f_1 with an attracting period three orbit, where two points of this orbit are in the reject region, and the other is in the accept region, and the attracting fixed points of the f_0 coincide, or nearly coincide, with this period three orbit. This prediction does, in fact, turn out to be correct for the ten training trials that we have run. One of these solutions is seen in Figure 4.

f_b puts the $\mathcal{C}$-system in the part of the phase space which will go to the accept region after two iterations of f_1. f_0 has three fixed states, corresponding to the three fixed points in the picture, and two saddle points. It will be seen that the fixed points nearly "sit above" the periodic orbit of f_1. $f_1$has a period three orbit. It is more instructive to look at the dynamics of the third iterate of f_1, $(f_1)^3$, rather than looking directly at f_1 and trying to see that the arrows will all match up to give this orbit. The attracting periodic orbit shows up as fixed points for $(f_1)^3$ and the unstable period three orbit contains the saddle points. The existence of a periodic orbit implies (by Brower's plane translation theorem) the existence of a fixed point. In this case, the fixed point is the point at the intersection of the three basin boundaries (stable manifolds). If two of the fixed points of the f_0 were in one basin of $(f_1)^3$ then the $\mathcal{C}$-system would not be able to model the FSA since a word containing many zeros between each one would put f_1's period three orbit "out of phase" with the FSA.

To see the utility of using ϵ-pseudo-orbits rather than true orbits for our theory we will now consider a solution to modeling this DFA which uses unstable invariant sets and discuss its properties. Suppose that f_b is a constant map, mapping all points to the fixed point of $f_0$that is in the upper left-hand corner of the phase space. Further suppose that the fixed points of f_0 are unstable (repelling rather than attracting) and that the periodic orbit of f_1 is unstable and maps the three fixed points of f_0 to one another. So we have that rather than approximately sitting above the fixed points of f_0 the periodic orbit of f_1 lies exactly above them. This $\mathcal{C}$-system clearly models the DFA, but it does not robustly model the DFA since if there is any perturbation to the orbit, then even a string of several zeros with moderate perturbations at each step is likely to fail to produce the correct output. Furthermore, if we perturb the maps at all and move the fixed points or the periodic orbit, then the RNN will fail to model the DFA. From this is follows that it is exceedingly unlikely

The arrows in Figure 2 show the direction that the point at the base of the arrow moves under iteration of the map. The distance that the point is moved is not shown. In the case of f_b, the point moves very close to the fixed point of the map (the point at which all arrows point). Note that the right side of the phase space (as you are looking at it) in the figure is the accept region (i.e. the region that projects to the interval $[1-\delta, 1]$) and the left side is the reject region, so the b symbol sets the network to accept, as it should, meaning that the system correctly classifies the empty string as being in 0^*1^*.

f_0 has two attracting fixed points (one in the accept, and one in the reject region, corresponding to q_1 and q_3 respectively), and one saddle-type fixed point. The curve that separates the basins of attraction for the two fixed points is the stable manifold of the saddle point, and is an example of an unstable invariant set (so one never "sees" it under iteration of f_0) which nevertheless can be found, and plays an important role in computation, since changing its shape and position can make the network fail to solve the problem by changing the basins of attraction for the fixed points. If, for example, we bent the right part of the curve down until it intersected the middle of the bottom side of the phase space, for example, then 0000001111111100000 would not be rejected.

f_1 has two attracting fixed points (one in the accept, and one in the reject region - corresponding to q_2 and q_3 respectively), and one saddle-type fixed point. Again, the curve which separates the basins of attraction for the two fixed points is the stable manifold of the saddle point. Note that the "accept" fixed point for f_1 is in the basin of attraction of the "reject" fixed point of f_0, as it must be, since, for example, 111110 is not in the language.

Thus we see that observing phase space dynamics can give a clear understanding of how the RNN operates. Observing a time series of the activations or clusters of activations gives a incomplete understanding of the performance of the network since, for example, it says nothing of the robustness of the solution or whether or not it will generalize to strings of arbitrary length. Furthermore, traces from phase space say little about the "difficulty" of evolving the ability to perform the task since some DFA with n states require more complicated dynamical systems to model them than do other n-state DFA.

Another common approach to understanding RNN solutions is to study connectivity strength patterns. This approach is analogous to a common neurobiological approach used to understand networks of neurons, and has many problems. Some of those problems stem from the fact that there exist many symmetries that can act upon the system while preserving the dynamics, some of which are discussed in Casey (1995) [3]. Other problems derive from the fact that this approach is analogous to attempting to look at the parameters of a high dimensional dynamical system and infer from them the dynamics. If the system were linear, this would be equivalent to looking at a matrix and directly, i.e. without explicit calculation, determine its eigenvalues (at least up to multiplication by a positive real number). Even for 2×2

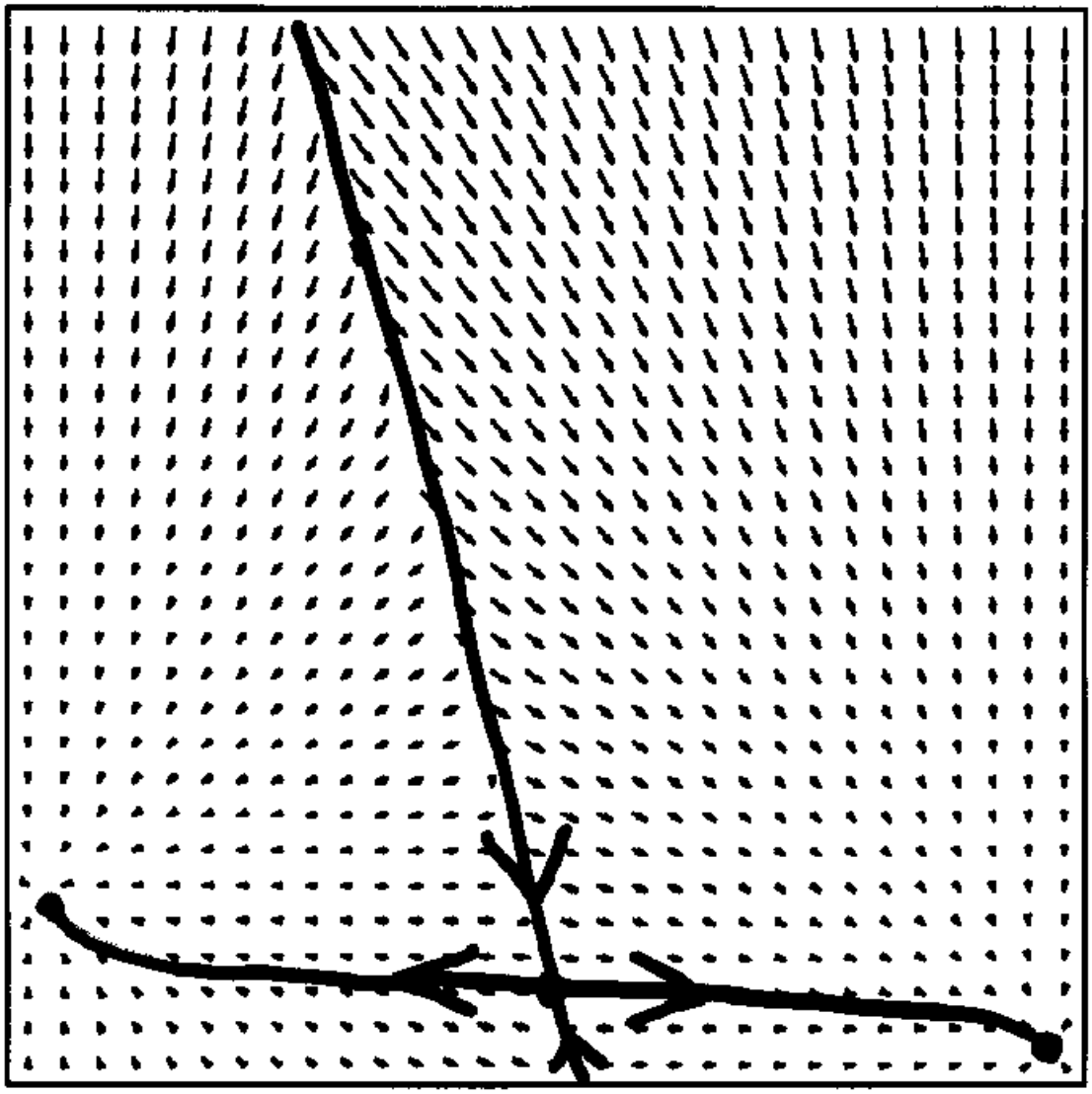

Figure 2: (C) Phase space dynamics of f_1.

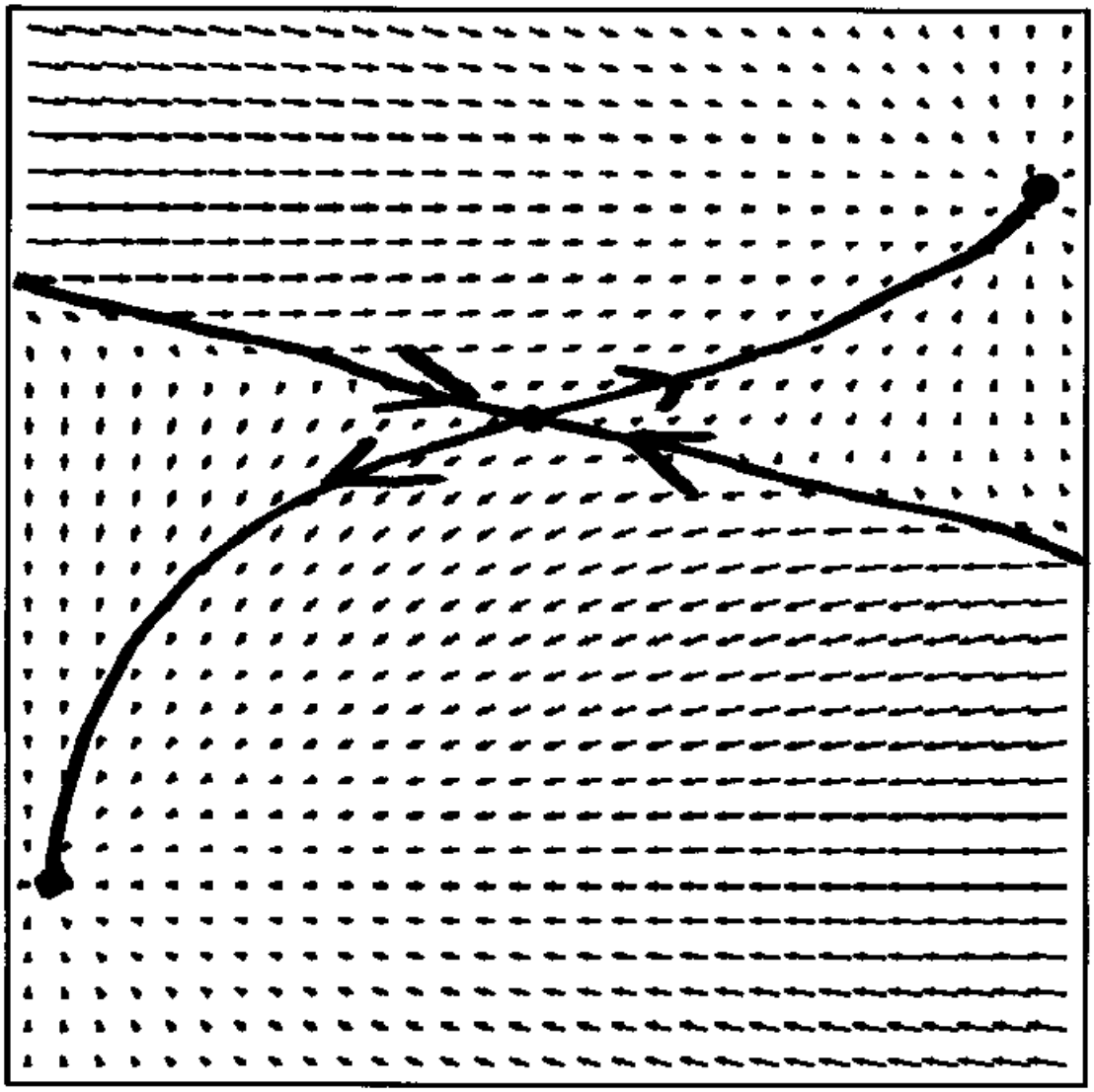

Figure 2: (B) Phase space dynamics of f_0 of the $\mathcal{C}$-system that recognizes the 0^*1^* language.

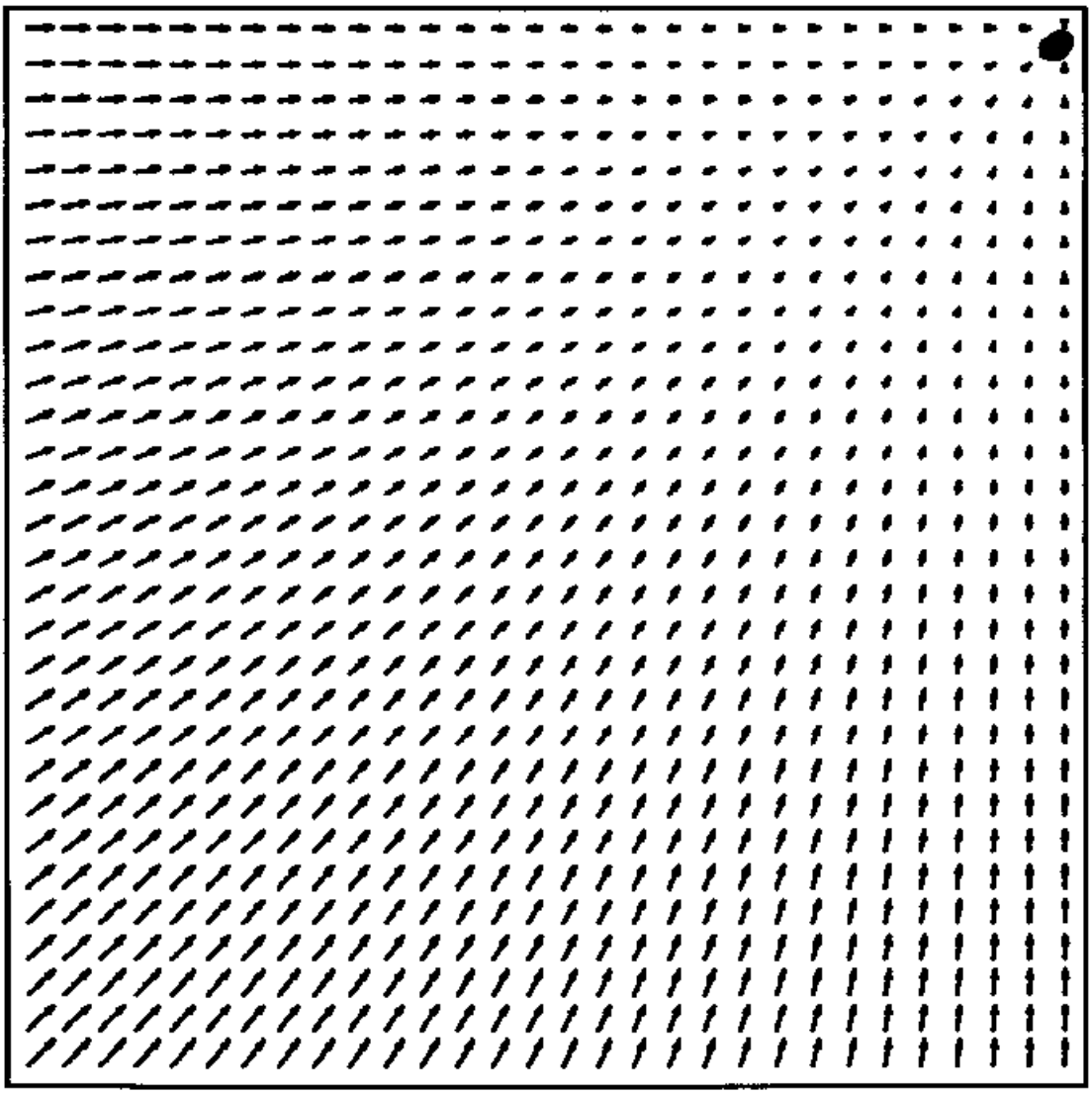

Figure 2: (A) a representation of the phase space dynamics of f_b of a two-unit $\mathcal{C}$-system that recognizes the 0*1* language.

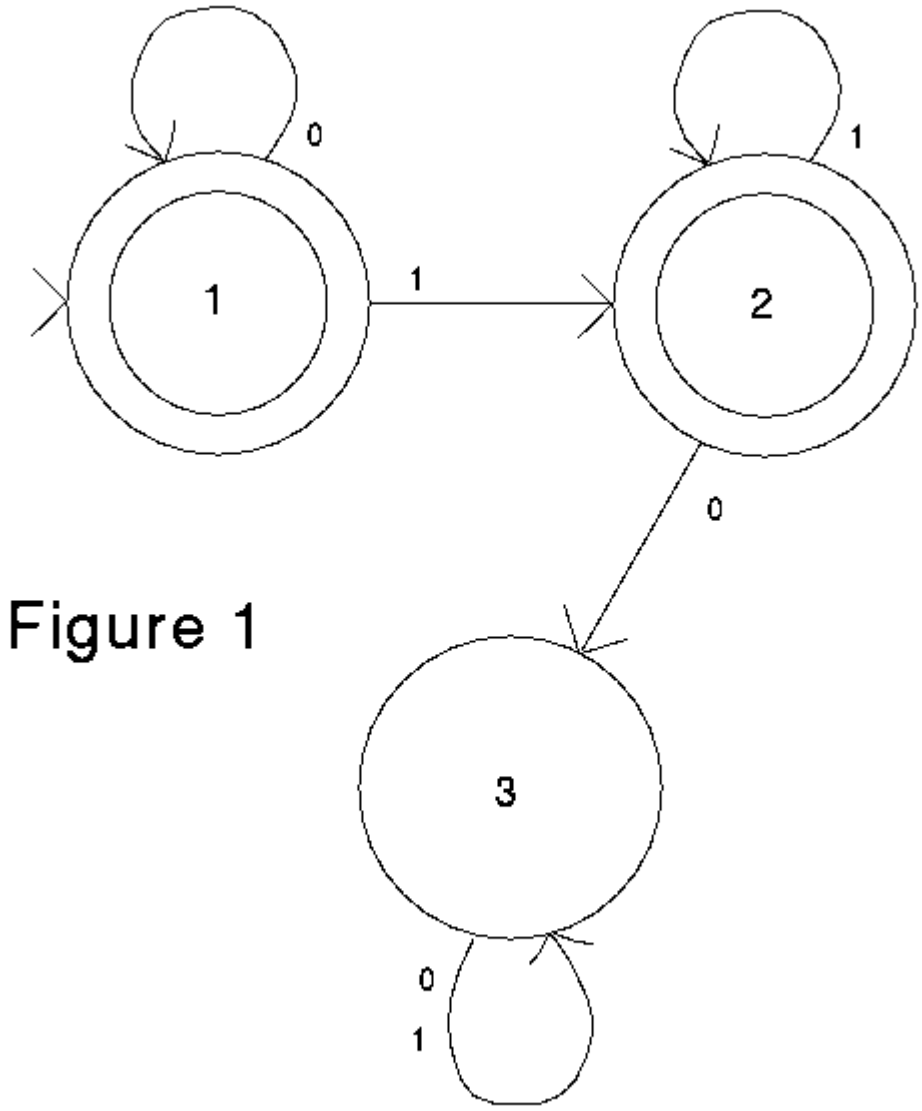

Figure 1

Figure 1: The diagram of the minimal DFA that recognizes the 0*1* language.

Chapter V

Examples

For the following problems, similar to the format in Giles *et al.* (1992) [8], the $\mathcal{C}$-system receives a unary encoding, with one input line for each character in $\{b, 0, 1\}$, where b marks the beginning of a string. The task is to recognize a regular language by robustly modeling a given DFA. The accept and reject regions are the regions of the phase space that project within δ of 1.0 or 0.0 on the target unit (i.e. the non-hidden, output unit) axis, for some $0 < \delta < 0.5$.

V.A Example 1

A problem that uses hysteretic computation is recognizing the 0^*1^* language[1]. The minimal DFA that recognizes the 0^*1^* language is shown in Figure 1. We see q_1 and q_3 are fixed states of f_0 and q_2 and q_3 are fixed states of f_1.

[1] 1^* could also be examined, but its simplicity makes it less instructive.

judicious choice of a family of functions to use to model DFA while inferring a DFA from examples; specifically, a family of functions whose complexity depends on degrees of freedom in choosing the one-dimensional maps, rather than in the dimension of the maps. The use of one-dimensional maps would greatly simplify the analysis we describe for evaluating the performance of the $\mathcal{C}$-system. The use of one-dimensional maps may also reduce the scaling of the complexity of the $\mathcal{C}$-system with the complexity of the DFA since increasing the dimension of the $\mathcal{C}$-system increases the number of parameters cubically for second-order $\mathcal{C}$-systems, but the number of parameters should increase only linearly with the degrees of freedom for a well-chosen family of one-dimensional maps. This question requires further investigation.

Another question that may be investigated using our work is whether or not teacher-forcing should be used. Teacher-forcing seems to work well for learning to perform hysteretic computation, but can lead to difficulties when used to learn transient computation. One potential problem with teacher-forcing is that it can restrict activation dynamics to a subset of the phase space, making the desired computation impossible. This possibility will be discussed with the examples.

The theorems in this sections show that $\mathcal{C}$-systems that model DFA have many constraints on their dynamics. These constraints outline an algorithm for studying $\mathcal{C}$-systems modeling DFA. First, one examines the minimal state diagram of the DFA to find all cycles. By theorem IV.2, these cycles correspond to behaviors required of the trained $\mathcal{C}$-system that correctly recognizes the language. One can then estimate the size of the $\mathcal{C}$-system needed to model the DFA, and determine whether or not to use teacher-forcing during training by considering the number of behaviors the $\mathcal{C}$-system must have. One should make sure that there are a sufficient number of training strings that contain iterated substrings corresponding to the cycles of the minimal DFA diagram. After training, one examines the stability of invariant sets of the f_I, including their ϵ-pseudobasins of attraction[1]. Finally, one checks that transient states (which are finite in number) map between appropriate ϵ-pseudobasins of attraction of the behaviors. If the trained $\mathcal{C}$-system performs correctly during these tests, then it will generalize to all unseen strings.

[1]This involves finding the "shapes" of unstable invariant sets, which is very difficult for high dimensional mappings.

complexity, and $\mathcal{C}$-systems must be given many degrees of freedom in order to learn to model such DFA.

At this point, it is useful to note that a fake periodic orbit can exist when its individual f_i are contraction mappings, so the existence of a single fake periodic orbit for a f_I should not make it more difficult to learn to model a DFA, which is another reason to call them fake orbits.

Theorem IV.3 *If a minimal DFA has a state that is a transient state of f_i then f_i must move a point from the ϵ-pseudobasin of attraction of one behavior to the ϵ-pseudobasin of attraction of a behavior not in the same class as the first, for some f_j, a composition of f_j or the projection map, where the basins of attraction of the projection map are the accept and reject regions.*

This is the weakest useful statement we can make about transient states. It is useful because it states the important feature that f_i must act productively on transients. We prove this by contradiction: if the f_i never move points from one ϵ-pseudobasin of attraction to another, then they will never produce a discernible change in the output of the $\mathcal{C}$-system, contradicting the minimality of the DFA. It is the weakest statement because it does not characterize the action of the f_I on points in phase space corresponding to an arbitrary transient state, which we can characterize in terms of intersections of iterated pre-images of ϵ-pseudobasins of attraction under the f_i.

Hence, a $\mathcal{C}$-system uses transient computation if it has an f_i with at least one transient state. The proof is to notice that if we are in a transient state for an f_i that is a recurrent point from the perspective of f_i dynamics, and if we then read a string of i's to iterate f_i and get back to the initial state, then the state of the DFA will have changed but the state of the $\mathcal{C}$-system will remain the same, contradicting theorem IV.1.

The distinction between transient and periodic states is important because periodic states are relatively well understood in the RNN case, and for a RNN with fixed discriminant functions, the dimension needed to have a given combination of fixed and periodic states should be an answerable but difficult question. Transients, however, are not well-understood, and the size of the RNN required to have certain kinds of transients is very sensitive to the shape of the discriminant function. Empirically, large numbers of hidden units are not necessary to model some DFA with large numbers of transient states, while the number of hidden units roughly scales with the number of fixed states and distinct periodic orbit states.

Here we note that the size of the RNN needed to solve a given problem can depend critically on the discriminant function chosen. If we allow the discriminant functions to be arbitrary by letting each input vector correspond to an arbitrary dynamical system, then it is always possible to model a DFA using only one-dimensional maps. It is easy, given a minimal DFA diagram, to construct by hand a family of dynamical systems that will robustly model that DFA. This fact suggests a more

Proof: The proof of this theorem relies on the observation that we may test the $\mathcal{C}$-system on a string of an arbitrary number of i's, and reading such a string corresponds to iterating f_i an arbitrary number of times. Suppose that f_i has more than one fixed state (which is a statement about the DFA), but f_i has only one behavior. By theorem IV.1 we know that the fixed states have corresponding disjoint, closed sets Q_i and Q_j. It was shown above that behaviors are open sets, so the complement of the behavior is closed. By the wandering lemma we know that some ϵ-pseudo-orbit of the $\mathcal{C}$-system will eventually attain the behavior under iteration of f_i. If the behavior contains points in both Q_i and Q_j, then there is an ϵ-pseudo-orbit of f_i that will change the state of the $\mathcal{C}$-system to a point corresponding to whichever of the fixed states the DFA is not in, contradicting theorem IV.1. Hence, the behavior is contained in either Q_i or Q_j. Without loss of generality, we may assume it is contained in Q_i. Hence, if the DFA is in state q_j and we read a string of N i's for N sufficiently large, then the $\mathcal{C}$-system will be in Q_i and therefore the $\mathcal{C}$-system fails to model the DFA. Simple induction on the number of fixed states completes the proof. ■

There are also the obvious generalizations to the period n case, $n \geq 2$, where we consider $(f_i)^n$ (the n-fold composition of f_i with itself) instead of f_i alone and the fake periodic case, where we consider general f_I.

It follows from theorem IV.2 that if an f_i has a period n state orbit and a period m state orbit, disjoint from one another (n may or may not equal m) then the $\mathcal{C}$-system is hysteretic. (Note that the existence of periodic states only depends on the DFA, but the conclusion that the $\mathcal{C}$-system is hysteretic is a statement about the $\mathcal{C}$-system that models the DFA.) Hysteresis implies the existence of unstable invariant sets. Hence, if hysteresis is required for a computation, then one can deduce that the maps in the $\mathcal{C}$-system must be far from constant mappings (since in a neighborhood of the constant maps, all of the corresponding f_i are contraction mappings, and hence, by the contraction mapping theorem, have unique behaviors). Hence, if we use the minimum distance from the constant mappings as a measure of difficulty of modeling a DFA or a measure of geometric complexity of the $\mathcal{C}$-system, then computations requiring hysteresis are more difficult to learn than those which do not require it. Of course, there will be a new set of problems related to computing minimum distances from the constant maps, especially in high dimensions, but this may be more approachable than trying to directly determine the complexity of the $\mathcal{C}$-system modeling a given DFA.

Another measure of geometric complexity of the $\mathcal{C}$-system that may be useful is the minimal number of behaviors for f_I. If we assume that we initialize maps sufficiently close to constant functions, then all of the f_I will be contraction mappings, and hence have invariant sets consisting of unique attracting fixed points. By our generalization of Theorem IV.2, if a $\mathcal{C}$-system models a hysteretic DFA, then at least one f_I will undergo a bifurcation during evolution. Hence, $\mathcal{C}$-systems that model DFA that are hysteretic with respect to many distinct strings will have high geometric

Lemma IV.1 *Let f be a continuous function of $\mathcal{I}^N$ into itself, and let B be a closed set of $\mathcal{I}^N$ with nonempty interior and define T to be the set of all $x \in B$ such that no ϵ-pseudo-orbit of x ever leaves B. Then the closure of T is a trapping region.*

Proof: To see that T is open, let $x \in T$ and let N be an ϵ-neighborhood of $f(x)$. Note that N is contained in B, since otherwise x could map out of B in one time step, which would contradict our assumption that x is in T, and N is in T since all points in N are in a possible ϵ-pseudo-orbit of x. By continuity of f, $f^{-1}(N) \bigcap B$ contains a neighborhood of x (since $f^{-1}(N)$ is open) and $f^{-1}(N) \bigcap B$ is in T since $f(f^{-1}(N) \bigcap B)$ is contained in N, which is contained in T.

Let C be the closure of T. To see that $f(C)$ is contained in T, and hence that C is a trapping region, first observe that if $x \in T$, then $f(x)$ is in T by the definition of T. Now suppose that x is in the boundary of C and let N be an ϵ-neighborhood of $f(x)$. Then $f^{-1}(N) \bigcap T$ is nonempty by the continuity of f and the openness of T, so there exists a $y \in f^{-1}(N) \bigcap T$. $f(y) \in N$, therefore $f(x)$ lies in an ϵ-pseudo-orbit of y. Hence $f(x)$ is in T since y is in T. ■

Lemma IV.2 (Wandering Lemma) *Suppose some closed set B with nonempty interior contains no behaviors. Then for any $\epsilon > 0$ and for any point $x \in B$ there is some finite number N such that the Nth point in some ϵ-pseudo-orbit of x will leave B.*

Proof: Assume there exists an $x \in B$ such that no ϵ-pseudo-orbit of x ever leaves B, then T (from the previous lemma) is nonempty, and therefore its closure is a trapping region, which is a contradiction. ■

The following theorem goes deeper into the representation issue by giving a coordinate-independent property of the f_i of the $\mathcal{C}$-system modeling a DFA which can be used as a measure of complexity of the DFA in terms of the complexity of the $\mathcal{C}$-system that models it.

Theorem IV.2 *Suppose that a $\mathcal{C}$-system robustly models a given DFA, then each f_i must have at least one distinct behavior for each of its fixed states.*

Furthermore, each behavior must be contained in either the accept or reject region of phase space according to whether or not its corresponding fixed state is also a final state of the automaton. There may be redundancy in the solution in the sense that there may be several behaviors in the $\mathcal{C}$-system which correspond to a single fixed state of an f_i. For this reason, the following definition will be useful:

Definition IV.2 *When there is more than one distinct behavior corresponding to a single fixed state, we say that these behaviors are <u>in the same class</u>.*

Chapter IV

Results

Our first theorem shows why clustering and DFA extraction succeed and provides a theoretical foundation for representation issues related to clustering and extraction:

Theorem IV.1 *The state space of a $\mathcal{C}$-system robustly modeling a given connected, minimal DFA must have mutually disjoint, closed sets, Q_i, with nonempty interior, corresponding to the states q_i of the DFA.*

The correspondence, of course is that the $\mathcal{C}$-system initialized with any point in Q_i will produce the same output as the DFA initialized to state q_i for all possible input.

Proof: The Q_i are nonempty by the fact that the $\mathcal{C}$-system is assumed to robustly model the DFA. To see that the Q_i must be mutually disjoint, assume that there is some point, x, in the intersection of two of these sets, Q_i and Q_j for i not equal to j, but that the $\mathcal{C}$-system still robustly models the DFA. Since the DFA is assumed to be minimal, there exists a string, which we call F, such that after reading F, the DFA will be in a final state if the DFA was in q_i, and in a nonfinal state if the DFA was in q_j. Now assume that the $\mathcal{C}$-system is in state x. After reading F, the $\mathcal{C}$-system will be in either the accept or reject region or neither. It follows that it must fail to model the DFA since if the DFA was initially in state q_i then after reading F it would be in a final state, in which case the $\mathcal{C}$-system must be in the accept region. But if the DFA was initially in state q_j then the DFA would be in a nonfinal state after reading F, in which case the $\mathcal{C}$-system must be in the reject region. But since the accept and reject regions are disjoint, this is impossible, and, therefore, the $\mathcal{C}$-system fails to model the DFA. Since the accept and reject regions are defined to be closed and the f_i are continuous, we can make this same argument for an accumulation point of any point in one of the Q_i, hence they are closed. The Q_i have nonempty interior by the fact that we are concerned with ϵ-pseudo-orbits. ■

Definition IV.1 *A trapping region for a map f is a closed set, T, such that $f(T)$ is contained in the interior of T.*

$y \in [x_u, x_{s1}]$ such that $f(y) - y > \epsilon$ that there is an ϵ-pseudo-attractor, $\mathcal{B}$, containing x_{s1}. As b decreases further, $\mathcal{A}$ is destroyed by the symmetric process.

Bifurcations in noisy systems are seen to be global and depend upon how contractive trapping regions are rather than depending upon local properties of the mapping such as derivatives at fixed points. Furthermore, it can be seen that the ϵ-pseudo-attractors and their ϵ-pseudobasins of attraction will rarely occupy full measure in noisy systems since the existence of an unstable invariant set, for example, will imply that their complement will have non-empty interior.

Definition III.16 *A transient of an f_I is a point not in $\mathcal{R}_\epsilon(f_I)$.*

Using the above definitions, we divide computation dynamics into categories: hysteretic and transient computation.

Definition III.17 *Hysteretic computation is computation that can occur using periodic orbits (of periods $\geq$ 1, where a periodic orbit of period one is a fixed point) of f_I of a $\mathcal{C}$-system.*

Definition III.18 *Transient computation is computation using transients of f_I.*

Definition III.14 *We say that a subset of $\mathcal{R}_\epsilon(f)$, $\mathcal{A}$, is an ϵ-pseudo-attractor or behavior if*

1) it is a subset of $\mathcal{R}_\epsilon(f)$ having a dense ϵ-pseudo-orbit which is maximal in the sense that there is no $\underline{x}$ which is dense in a set $\mathcal{B}$ of which $\mathcal{A}$ is a proper subset, and

2) $\mathcal{A}$ is a proper subset of a set, $\rho(\mathcal{A})$, called the ϵ-pseudobasin of attraction, which is the set of all points whose ϵ-pseudo-orbits are all contained in $\mathcal{A}$.

Intuitively then, an ϵ-pseudo-attractor corresponds to an oscillation or a steady state of the system which is stable in the sense that a slightly perturbed system will settle to one of these asymptotic behaviors.

This definition for an attractor has many satisfying properties. First, ϵ-pseudo-attractors are open sets, and hence have relatively simple (nonfractal) geometry. Because we are concerned with mappings of compact metric spaces into themselves, there can be only a finite number of ϵ-pseudo-attractors for any given mapping, and each of those can have only a finite number of connected components. By the second property, the existence of an ϵ-pseudo-attractor implies the existence of a trapping region.

This definition incorporates a very strong notion of robustness in the sense that the existence of an ϵ-pseudo-attractor implies the existence of at least one of Milnor's attractors, but the converse is not true. To further understand the nature of ϵ-pseudo-attractors we will look at an example of a bifurcation in a one-dimensional dynamical system, but first we make the following definition.

Definition III.15 *A mapping f_I is said to be hysteretic if it has more than one ϵ-pseudo-attractor, and a $\mathcal{C}$-system is hysteretic if for some string of inputs I, f_I is hysteretic.*

Hence a hysteretic map is a dynamical system that embodies several different robust, nonlinear oscillations that are selected based upon initial conditions.

For the sake of concreteness, our example will use, $f(x)$, the logistic squashing function with one parameter, b. so $f(x) = \frac{1}{1+e^{-5x+b}}$ and a low level of noise, say $\epsilon = 0.01$. If we now let $b = 2.5$ then f will have an unstable fixed point at 0.5 and two attracting fixed points, one near zero, called x_{s0}, and the other near one, called x_{s1}. Assume that initially b is much larger than zero, and hence will have one ϵ-pseudo-attractor, $\mathcal{A}$, approximately equal to an interval twice the length of ϵ , centered on x_{s0}. The ϵ-pseudobasin of attraction for $\mathcal{A}$ is the entire real line. As we decrease b, there are points, b_0 and x_0 such that $f(x_0) - x_0 = \epsilon$. For $b < b_0$ the ϵ-pseudobasin of attraction for $\mathcal{A}$ will contain only points less than x_0. For b near b_0 but less than b_0, $\mathcal{R}_\epsilon$will have a second connected component, near one, that will not be an ϵ-pseudo-attractor. As we further decrease b, f undergoes a saddle node bifurcation to create the stable fixed point, x_{s1}, near one and an unstable fixed point, x_u, between x_{s0} and x_{s1}. However, it is not until b decreases to the point at which there exists some point

To rule out solutions which may require noiseless operation of the neural network or physical system, we will require that the computation be robust in the following sense:

Definition III.9 *We say that the system is operating in a noisy environment if, for some fixed $\epsilon > 0$, the state variables of the $\mathcal{C}$-system, $\mathbf{x}(t)$, may be replaced by any point within ϵ of $\mathbf{x}(t)$ at each time step, t.*

That is, we are concerned with ϵ-pseudo-orbits rather than the true orbits of our state variables under the action of the maps.

Definition III.10 *Fix an $\epsilon > 0$, let f be a map of $\mathcal{I}^N$ into itself and let $x \in \mathcal{I}^N$. An ϵ-pseudo-orbit of x under f is a sequence of points $\underline{x} = \{x_0, x_1, \ldots\}$ such that $|f(x_{i-1}) - x_i| < \epsilon$ for all $i > 0$, where $x_0 = x$.*

Unless noted otherwise, whenever we speak of the iteration of a point we will assume that the system is operating in a noisy environment.

Definition III.11 *If the $\mathcal{C}$-system always produces the correct output while operating in a noisy environment for some fixed $\epsilon > 0$, then it robustly models the DFA or, if L is the language associated with the DFA, recognizes L.*

See Wiggins (1990) [26] or Devaney (1987) [6] for additional definitions and background from dynamical systems theory. Since we are concerned with systems operating in the presence of noise, we will need to define an attractor for such systems, and to do that we need a notion of recurrence. For ϵ-pseudo-orbits, the recurrent points are those that are ϵ-pseudoperiodic:

Definition III.12 *A point, x, is said to be ϵ-pseudoperiodic for a map f if there exists an ϵ-pseudo-orbit, $\underline{x}$, such that $x = x_n$ for some $n > 0$.*

One set that is often useful for talking about ϵ-pseudo-orbits is the chain recurrent set, defined to be the set of all points which are ϵ-pseudoperiodic for all $\epsilon > 0$. Since we will always have a finite amount of noise in our systems, the following will be more useful:

Definition III.13 *For some $\epsilon > 0$, the ϵ chain recurrent set, $\mathcal{R}_\epsilon(f)$, is the set of all points which are ϵ-pseudoperiodic for ϵ , where $\epsilon > 0$ is fixed.*

Intuitively, the attractors of a dynamical system are the sets of points to which most points evolve under iteration, but there is no one universally accepted definition of an attractor. Milnor (1985) [15] reviews some of the existing definitions and proposes another, but none of these are well-suited to noisy systems. The following definition seems to correspond to the intuitive notion of an attractor for systems with noise:

Definition III.5 *Call a state which is not in a period n state orbit ($n \geq 1$) for an f_i a transient state of f_i.*

Definition III.6 *For all other I, i.e. those which contain at least two distinct symbols, we say that an n-cycle, Q, is a fake period n orbit of f_I.*

Such orbits are called "fake," because they can be made of transient states of the intermediate f_{i_j}.

In order to speak of the $\mathcal{C}$-system modeling a DFA, we need to have some way of connecting the state of the system to the state of the DFA. In particular, we need to have some way of determining whether the state of the system corresponds to a final or nonfinal state of the DFA. To this end, we need the following definitions.

Definition III.7 *Let A and R be disjoint, closed subsets of $\mathcal{I}^N$, called the accept and R the reject regions of the $\mathcal{C}$-system or simply the accept and reject regions, respectively.*

If necessary, we can easily generalize this definition to DFA with more than two types of output, thereby extending the theory to modeled systems that have many distinct outputs.

Definition III.8 *A $\mathcal{C}$-system is said to model a DFA if, for any input string S, the $\mathcal{C}$-system is in the accept region if the DFA is in a final state after reading S, and in the reject region if the DFA is in a nonfinal state after reading S.*

Under this definition it is possible for a $\mathcal{C}$-system to model a DFA using unstable invariant sets of the individual f_I. An example will be given in the examples chapter, but before continuing we will explain why it is desirable to consider only solutions with a finite amount of noise. The first thing is that for organisms behaving in and interacting with a complicated environment there will inevitably be perturbations of the organism from sources other than those which are directly related to the computation in which we are interested, and we would like to consider only systems which can still perform correctly in these circumstances. In other words, it is desirable, for example, to consider only systems which can still add even though the wind is blowing. A second reason to consider solutions with finite amount of noise is that if we are implementing the system on a digital computer then there will be noise in the system due to round-off errors. So we would like to restrict our attention to solutions which work in spite of round-off errors. A third reason to consider robust solutions is that if we are using a optimization algorithm to find the solution (as in the neural network case), then we would like to consider solutions which will be well behaved with respect to small perturbations in the parameters of the system. As will be seen in the example, in order to model a DFA with unstable invariant sets it is often necessary to "match-up" invariant sets for the various f_i, and therefore if we perturb one of these maps and move the invariant set slightly then the invariant sets will no longer match-up and the $\mathcal{C}$-system will no longer model the DFA.

Chapter III

Definitions

Let $\mathcal{I} = [0,1]$, the unit interval of the real line.

Definition III.1 *Let N be a fixed, finite number. Consider $\mathcal{I}^N$ and a collection of mappings, $\{f_I\}$ for $I \in \Sigma^*$ over some finite alphabet Σ. For each symbol, $i \in \Sigma$, f_i will be required to be a continuous map of $\mathcal{I}^N$ into its interior. Furthermore, if $I = i_1 i_2 \cdots i_n$, then $fI = f_{i_n} \circ \cdots \circ f_{i_2} \circ f_{i_1}$ This collection, ($\mathcal{I}^N$, $\{f_I\}$, Σ) will be called an N-dimensional $\mathcal{C}$-system.*

Hence, each of the $\{f_I\}$ is a dynamical system mapping $\mathcal{I}^N$ into itself. A $\mathcal{C}$-system then is our abstraction of a biological organism. $\mathcal{I}^N$ is the phase space, after possible rescaling, of N separate measured aspects of the organism's behavior which are relevant for producing the behavior of interest. $\{f_i\}$, for $i \in \Sigma$, is the collection of update rules for the system when the organism is given different inputs (external and/or internal) corresponding to each of the elements of Σ.

We now turn to DFA, which can also be viewed as very simple models of biological organisms. Assume that all DFA models are minimal with respect to the number of states, for the language that they recognize.

Definition III.2 *Define an n-cycle ($n \geq 1$) of f_I to be an ordered set of n states $Q = \{q_1, \ldots, q_n\}$ such that when I is read, the DFA will pass through these states, beginning and ending with q_1.*

Note that in this definition we are combining f_I of the $\mathcal{C}$-system and the states of the DFA, which are not related *a priori*, but we will show that the two correspond in a well-defined manner.

Definition III.3 *If I is a single symbol, i, then we will call a 1-cycle of the f_i, $\{q_1\}$, a fixed state of f_i.*

Definition III.4 *If I is the composition of a symbol, i, with itself n-times, then we say that Q is a period n state orbit of f_i.*

1. $\emptyset$ and each member of Σ is a regular expression.
2. If α and β are regular expressions then so is $(\alpha\beta)$.
3. If α and β are regular expressions then so is $(\alpha \bigcup \beta)$.
4. If α is a regular expression then so is α^*.
5. Nothing is a regular expression unless it follows from (1) through (4).

Examples of regular expressions over {0,...,9} are 5, 4^*, 3827, $(3 \bigcup 8^*)$ and $(274 \bigcup 12^*)3(45)^*2884(2 \bigcup 6)$.

The **regular languages** are exactly those languages which can be described by regular expressions. For example, the regular expression $(3 \bigcup 8^*)$ corresponds to the language consisting of the empty string, 3 and all strings containing only 8's. The language corresponding to $(274 \bigcup 12^*)3(45)^*2884(2 \bigcup 6)$ contains the strings 274345454528846, 12121212328842 and 3454528846 among others.

Intuitively we will prefer to think of finite state transducers rather than DFA. A finite state transducer is like a DFA except that its purpose is not to accept or reject strings, but rather it is to produce output as a function of the input strings that it is given. On each step a finite state transducer will write to an output tape (or otherwise produce output) zero or more symbols based upon its present state and the input symbol. In particular, we will think of a DFA as outputting a zero if the DFA is in a nonfinal state and a one if it is in a final state. Thus the process of correctly classifying strings can be viewed as a dynamic response to time varying input which had arbitrary dependencies upon previous input.

What we mean by arbitrary dependencies upon previous input is that given two strings s_1 and s_2 which agree in all but their M^{th} position then for any $N \geq M$ there is a DFA $\mathcal{M}$ such that $\mathcal{M}$ will be in a nonfinal state after reading N symbols of s_1 and in a final state after reading N symbols of s_2. This corresponds to the intuitive notion of having memories of events from any past time which can influence present behavior.

Another way that a DFA can be thought of as a system which has arbitrary dependencies is that there are fixed DFA such that there is no finite number M such that, independent of the input string, the output of the DFA will depend only upon the last M characters read. An example of this is the DFA which accepts the language 1^*. There is no such M for this DFA since we can consider a string of M ones preceded by either a zero or a one. If the next symbol is a one, then in the first case the DFA will continue to output a zero, and in the latter case it will continue to output a one. Furthermore, if the next symbol is a zero, then in the first case the DFA will continue to output a zero, and in the latter case it will stop outputting ones and output a zero. So the output of the DFA can change depending upon the input at arbitrary past times.

The generalization to Turing machines then is to simply exchange the read from left to right only head of the DFA for a read/write head with the ability to move in either direction under the control of the DFA.

is, a language is any subset of Σ^*. For example, if $\Sigma = \{0,\ldots,9\}$, then one language is {0,1,2,2112}. Another language over Σ would be the collection of all strings containing exactly one 0. Yet another would be the collection of strings containing the same finite number of 0's as 1's.

We are now ready for a formal model of a computing machine. Since Turing machines are composed of a finite state control and a tape, and we will be interested in understanding the finite state control (the analog of the biological organism), we will use finite state machines as our models of computing machines. Here and following, when applicable, we shall use the notation of Lewis and Papadimitriou (1981) [13], including the following definition of a finite state control or deterministic finite state automaton.

Definition II.1 *A* **deterministic finite state automaton (or DFA)** *is a quintuple* $M = (Q, \Sigma, \delta, s, F)$ *where*

Q *is a finite set of* **states**,
Σ *is an alphabet,*
$s \in Q$ *is the* **initial state**,
$F \subseteq Q$ *is the set of* **final states**,

and δ, *the* **transition function**, *is a function from* $Q \times \Sigma$ *to* Q.

In order to define computation we will first need to intdroduce the notion of a *configuration.* A **configuration of M** is an element of Q $\times$ Σ^*corresponding to the present state of M and the rest of the string on M's input tape. A **computation** is then a sequence of N configurations corresponding to some DFA reading N consecutive symbols from its input string. Hence a computation keeps track not only of the sequence of internal states of the machine but also the future input which the machine will receive.

The computations which can be performed by DFA are exactly those which are equivalent to accepting a class of languages called *regular languages.* Hence any DFA represents a regluar language. Since we will concern ourselves with modeling DFA, the formal computations that we will focus on in this dissertation will be those required to recognize regluar languages. In order to define a regular language, we first need to define several operations which can be performed on languages. The first will be denoted by $\bigcup$ which is just the union of languages as sets of strings. Hence if L_1 = {5,666} and L_2 = {1,1000}, then $L_1 \bigcup L_2$ = {1,5,666,1000}. Another language operation is the **Kleene star** of a single langauge L, denoted by L^*. L^* is the language obtained by concatenating zero or more strings from L. For example, if $L = 1$, then $L^* = \{e,1,11,\ldots\}$ where e is the empty string. If L = {1, 1000} then 100011111000 $\in L^*$ since it is equal to 1000 $\circ$ 1 $\circ$ 1 $\circ$ 1 $\circ$ 1 $\circ$ 1000.

We are now ready to introduce the notion of a regluar expression.

Definition II.2 *The* **regular expressions** *over an alphabet* Σ *are strings of the alphabet* $\Sigma \cup \{), (, \emptyset, \bigcup, *\}$ *such that the following hold:*

Chapter II

Models of Computation

Biological organisms can be seen as machines with fixed size memories capable of combining internal and external memories to produce appropriately structured output. One abstraction of this computational process is embodied by the Turning machine. Turing machines are made up of two main parts. First, there is the finite state control unit, which, combined with a read/write head is the abstraction of the organism producing the computation. The second part is the tape which is written upon and read by the head. The tape is the abstraction of the environment.

In some ways, Turing machines are very attractive as models of computation. First, and probably most importantly is the Church-Turing thesis, which states that computational procedures are exactly those procedures that can be presented as a Turing machine. So Turing machines are sufficiently powerful to carry out very difficult computations, but perhaps more surprisingly, it may not be possible to add any finite amount of resources to Turing machines to make them more powerful. Another attractive quality of Turing machines is that the division of resources between the finite state control and the tape is somewhat satisfying from the perspective of separating the organism from its environment for the sake of understanding the role of each.

The main problem with Turing machines from our perspective is that they do not seem to compute in a manner resembling the way that computation is carried out in the nervous system of biological organisms. It will be seen later that the type of analysis in this thesis carries over to computations which are similar to the formal computations which we use.

In order to speak of the operation of computing machines we need to represent input to the machine. To do this we will use the notion of a *string of symbols.* The symbols come from an **alphabet** Σ which is a finite set of symbols. One example of an alphabet is the set of numbers from 0 to 9 $\{0,\ldots,9\}$. A **string** over an alphabet is a finite ordered sequence of elements from the alphabet. For example, 6195580665 is a string over the alphabet $\{0,\ldots,9\}$. The collection of all strings over and alphabet Σ is denoted Σ^*.

A **language** over some alphabet Σ is any collection of strings over Σ. That

give a sort of minimal complexity in terms of dynamic properties of the RNN. We have greatly benefited from the framework for solving problems with complicated temporal constraints that Giles *et al.* (1992) [8] have constructed. Hence, when it is useful to speak of a particular type of model, we will use their general framework, though our results are more general and apply to any collection of discrete-time dynamical systems acting on a metric space, including first-order and the simple recurrent networks (as in Elman (1989) [7]), and easily extend to problems with any finite number of inputs or outputs.

More precisely, we use the following equations for the examples:

$$x_k(t+1) = F(\mathbf{x}(t), \mathbf{y}(t)) = S(\sum_{i=0}^{N}\sum_{j=0}^{M} W_{ij}^k x_i(t) y_j(t))$$

where $\mathbf{x}(t)$ is the state vector of activations of the network's N units, $\mathbf{y}(t)$ is the vector of M inputs to the network, and W_{ij}^k is the second-order weight between the ith unit and the jth input for the kth unit. S is a fixed discriminant function. For the examples, we will use the logistic squashing function $\frac{1}{1+e^{-x}}$, but the details of the models in our examples are not important. In order to use our theory the important requirements of a model are that there are as many distinct discrete-time dynamical systems as there are possible inputs (as is the case for first- and second-order RNNs, Elman networks and many variations of these basic RNN architectures).

in the mediating system.

It is useful to speak of structure in terms of coordinate-independent dynamical properties, as we do here, since coordinate-dependent properties of the system (e.g., sizes and shapes of phase space orbit clusters and distances between them) are confounded by symmetries and redundancies of possible solutions, so no good, general theory can be based upon them. Knowledge about the system performing the calculation might eliminate some of the potential solutions and therefore give a more precise description of solutions to the problem, and in particular what all solutions must have in common. That is, one may be able to give a more detailed description of the system and its behavior by eliminating some of the possible solutions.

The main result of this thesis is a description of how to study a DFA's minimal state diagram in order to determine up to some equivalence what the system must be doing. Our theory identifies the aspects of a DFA that are important in terms of relating it to the system that models it. More generally, we provide an example of how to determine the salient features of a behavior in order to say which parts of the behavior imply the greatest internal complexity; knowledge that may be useful for exploring the evolution of behaviors in biological organisms or learnability in RNNs. That is, we may have a behavior who's description is given in terms very different from those that will be important for determining how difficult it will be to produce when evolving a physical system, but we may be able to translate that description into one which is more suited to such measures. For example, one abstract description of the behavior of regular language recognition with which is very well suited to understanding the workings of a physical system producing that behavior is given by the minimal DFA diagram of a behavior. This is explored further in the results of this thesis.

Yet another way to view this work is as a discussion of representations, an issue central to cognitive science. In particular, we address the representation of relationships and dependencies used during some computations. We give representations in terms of properties of the individual dynamical systems (those dynamical systems that we get by giving the system a constant input), rather than in terms of traces of phase space activities or in terms of the connectivity strengths between units, which may change drastically as the model changes.

Many types of RNNs have been shown to be capable of modeling DFA (Siegelmann and Sontag (1991) [23], Giles *et al.* (1992) [8]); in particular, RNNs of the type studied in Williams and Zipser (1989) [27] and Giles *et al.* (1992) [8] can be used to model DFA. Omlin and Giles (1994) [18] provide a constructive proof that an $N+2$-unit second-order RNN is capable of modeling any given N-state DFA. Rather than constructing specific solutions to the problem of modeling various DFA, we solve the converse problem by showing what all solutions must have in common and thereby

an organism's behavior, which we can never determine empirically if there are an infinite number of potential situations to which the organism must react, and then exploring the complexity that each of the hypotheses would imply in the system.

Many authors have directly or indirectly approached the problem of understanding their inner workings (Cleermans *et al.* (1989) [4], Cummins (1993) [5], Giles *et al.* (1992) [8]) but have provided only a limited understanding for several special cases of the problem. Rather than answering the question for several classes of RNNs modeling specific DFA, we address the deeper question: "what *must* a RNN do in order to model a given DFA?" and we present a unified approach to understanding a wide variety of problems that can be formulated in terms of DFA. By describing the sense in which trained networks are unique, this approach allows one to effectively predict RNN dynamics in a network that models any given DFA. Furthermore, the approach outlined in this thesis allows informed guesses about how to train RNNs with respect to the size a RNN must be in order to model a DFA, whether to use teacher-forcing (see Williams and Zipser (1989) [27]), and roughly what distribution of training strings to use. Our approach also allows for exploration of DFA learnability without resorting to empirical results from training attempts. We also describe a finite series of tests that determine whether the RNN solution will generalize to correctly classify all possible inputs. Additional details of these results will be given in future papers.

Recently proposed measures of RNN performance aim to elucidate the workings of RNNs. For example, Cleermans *et al.* (1989) [4], Elman (1989) [7]and Giles *et al.* (1990) [9], have proposed that cluster analysis of hidden unit activation dynamics yields an adequate understanding of RNN behavior. Others (Pollack (1991) [19]) have taken time series from hidden unit phase space. Still others, for example Williams and Zipser (1989) [27], have looked directly at the parameters which determine which collection of dynamical systems in the family of RNNs defined by the RNN equations has evolved under the influence of gradient decent optimization over the parameters. In the language of RNN studies, they studied connectivity strengths between units after the back-propagation learning rule had found the weights.

In this thesis we will show that none of these measures are sufficient for understanding RNN behavior in general, and we will describe more useful measures. During the preparation of this work, we found that Cummins (1993) [5] had independently discovered some similar results, but our theory includes his results as a special case, answers several questions he poses, and demonstrates why he got the results that he did by clarifying which "environmental constraints" are important.

Cognitive scientists would like to understand complicated natural and artificial systems that were not explicitly designed, e.g. biological organisms or artificial systems with many parameters, that produce some interesting (e.g. "intelligent") behavior. The cognitive science approach to understanding these systems involves carefully observing the behavior of the system and then attempting to infer system structure in order to elucidate its inner workings. Our work is consistent in spirit with this approach, since we start out assuming that we have a complete understanding of the task[1] (the task in this case is to "act like" a DFA), and then we infer structure

[1]This assumption, of course, means that we first hypothesize the full task structure underlying

Chapter I

Computation Dynamics: Background

One day it became apparent to us that there are many stumbling blocks to understanding computation in biological organisms. One of those problems is in describing the enormously complex processes involved in such tasks as pattern recognition and real world planning. Another is to understand how the organism itself instantiates the computation that has been captured by some more abstract description. This thesis is an approach to answering the second question for a rather simple (from a biological perspective) class of computation which nontheless may be useful as a building block for understanding more realistic models of computation in biological organisms.

In this thesis, we will build upon ideas first presented in Casey (1993) [1], and focus on computation equivalent to that of regular language recognition, where a regular language will be seen to be those languages which are recognizable by deterministic finite state machines. The mathematical abstraction of a biological organism which we shall use will be a collection of continuous mappings of the N-cube, $\mathcal{I}^N$, into its interior. $\mathcal{I}^N$corresponds to N state variables of the biological system which capture the essential internal behaviors which produce the organism level behaviors of interest. Each map describes how that system will change when given a corresponding fixed input.

This abstraction of a biological system is a generalization of connectionist style recurrent neural networks (RNNs) which have been studied since McCulloch and Pitts (1943) [14], and have more recently been studied in Giles *et al.* (1992) [8] and elsewhere. Since there has been work done in this special case of the problem we address in this dissertation we will discuss this existing work as background for our work. A paper is in preparation which more directly deals with the RNN treatment of these ideas Casey (1995) [2].

The interest and progress in the study of RNNs over the last several years has seen some progress in understanding their working, but RNNs are still often seen largely as black boxes without a solid theoretical understanding of how they compute.

ABSTRACT OF THE DISSERTATION

Computation In Discrete-Time Dynamical Systems

by

Michael P. Casey
Doctor of Philosophy in Mathematics
University of California, San Diego, 1995
Professor Michael H. Freedman, Chair

Deterministic finite state automata (DFA) can be robustly modeled using collections of mappings of a metric space into itself. We examine the extent to which perfect knowledge of the computation to be performed constrains possible phase space dynamics during computation. We show that if a collection of mappings is initialized to near constant maps, then some map must undergo a bifurcation at some point as the system evolves if it is to model a DFA with properties we will describe later. We also show how to effectively predict phase space dynamics in a such system which models a DFA, and thereby allow one to understand computation dynamics in spite of the possible existence of unnecessary complexity in phase space dynamics.

We also address issues in using the cognitive science approach to understanding systems that produce a behavior that is of interest but not a result of explicit system design, e.g. when an optimization algorithm has been used to produce the system or when the system is not man-made.

To Arnie

ACKNOWLEDGEMENTS

Without Arnie Mandell, the work in this thesis would certainly not exist. His ideas and direction have been the one guide to which I have always tried to adhere during my academic career. He and Karen have not only inspired and encouraged me over the years, they have also made me feel a part of their family.

I would like to thank Michael Freedman for his time and patience in teaching me some mathematics. I would also like to thank him for his inspiration and resources, and for giving me freedom enough to pursue the ideas and results in this thesis. I would like to thank him for laughing when my best friend, Steve, answered my phone with "Mikey's love palace.... How may we serve you?"

I would like to thank: Arnie, Mike Freedman, Bruce Driver, Gary Cottrell, Kimberly Claffy, David Zipser, Don Norman, Martin Paulus, Jeff Elman, Fu-Sheng Tsung, Peter Rowat, Cecilia, Elad and everyone else who has been a part of my formal education and training.

For helping me learn about the rest of life, I thank Mama, Dear Old Dad, Stevie, Orna, Julie, Sue, Wilma, Meg, Skip, Sally, Study Steve, Kelly, Theresa, Elad, Taffy, Christine, Jennifer, Mastro and others who'd I've thought of as homies.

PUBLICATIONS

"Computation Dynamics in Discrete-Time Recurrent Neural Networks," in UCSD's Institute for Neural Computation's *Proceedings of the Annual Research Symposium,* June, 1993, (pp 78-95).

"How Recurrent Neural Networks Work," Tech. Report UCSD-MATH-95-1, University of California, San Diego, La Jolla, March 1995.

"Relaxing The Symmetric Weight Condition for Convergent Dynamics in Discrete-Time Recurrent Networks," Tech. Report UCSD-MATH-95-2, University of California, San Diego, La Jolla, March 1995.

FIELDS OF STUDY

Major Field: Mathematics

Studies in Geometric Topology.
Professor Michael H. Freedman
Studies in Dynamical Systems Theory.
Professors Michael H. Freedman and Arnold J. Mandell

VITA

Jan. 31, 1967	Born, Las Vegas, Nevada
June 1989–Sept. 1990	NSF's Research Experience for Undergraduates at the San Diego Supercomputer Center (three terms) University of California, San Diego
Sept. 1989–Sept. 1990	Regents Scholarship, University of California, San Diego
Sept. 1989–Sept. 1990	President's Undergraduate Fellowship University of California, San Diego
Sept. 1990	B.A. in Mathematics (Summa Cum Laude), University of California, San Diego
Sept. 1990	Third College Provost's Award for Academic Achievement, (Third College's highest academic honor, given to one individual from each graduating class) University of California, San Diego
Sept. 1990	National Science Foundation Graduate Fellowship (declined) University of California, San Diego
Sept. 1990–Sept. 1993	National Defense Science Graduate Fellowship University of California, San Diego
Sept. 1993–Dec. 1993	Teaching Assistant, Department of Mathematics University of California, San Diego
Dec. 1993	M.A. in Mathematics, University of California, San Diego
April 1994	C.Phil. in Mathematics, University of California, San Diego
Dec. 1993–Sept. 1995	Center for Nonlinear Studies, Los Alamos Labs, Graduate Assistantship University of California, San Diego
Dec. 1993–Sept. 1994	National Science Foundation, Graduate Assistantship University of California, San Diego
Sept. 1994–Dec. 1994	Associate In Mathematics, Department of Mathematics University of California, San Diego
Mar. 1995	Doctor of Philosophy in Mathematics University of California, San Diego

TABLE OF CONTENTS

Signature Page iii

Table of Contents iv

Vita v

Publications and Fields of Study vi

Acknowledgements vii

Dedication viii

Abstract ix

I Computation Dynamics: Background 1

II Models of Computation 5

III Definitions 8

IV Results 13

V Examples 18
A. Example 1 18
B. Example 2 24
C. Example 3 30

VI DFA Extraction 36

VII Simulations 40
A. Example Params File 40
B. Example DFA File 40
C. Gram.c Code 40

VIII Conclusions 42

Bibliography 44

The dissertation of Michael P. Casey is approved, and it is acceptable in quality and form for publication on microfilm:

Halbert White

Terrence Sejnowski

Arnold J. Mandell

Zheng-Xu He

Peter Doyle

Michael H. Freedman, Chair

University of California, San Diego

1995

Copyright

Michael P. Casey, 1995

All rights reserved.

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Computation In Discrete-Time Dynamical Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Mathematics

by

Michael Patrick Casey

Committee in charge:

Professor Michael H. Freedman, Chair
Professor Peter Doyle
Professor Zheng-Xu He
Professor Emeritus Arnold J. Mandell
Professor Terrence Sejnowski
Professor Halbert White

1995