A Temporal Sequence Processor Based on the Biological Reaction-Diffusion Process

Sylvian R. Ray and Hillol Kargupta
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Abstract

Temporal sequences are a fundamental form of information and communication both in natural and engineered systems. The biological control process which directs the generation of iterative structures from undifferentiated tissue is a type of temporal sequential process. A quantitative explanation of this temporal process is **reaction-diffusion**, initially proposed by Turing [1952] and later widely studied and elaborated.

We have adapted the reaction-diffusion mechanism to create a novel network and algorithm based on a chemical "neuron" model, which performs storage, associative retrieval and prediction for temporal sequences. Experiments demonstrate the ability of the device to achieve any desired *depth* limited only by storage capacity, to remember and predict on the basis of count to any length, and to learn and recognize embedded Reber grammar strings to 98% accuracy. The network is also capable of preserving time extent of stored symbols, such as in a musical melody, and permitting retrieval of both the symbols and their temporal extent. Redundancy can be controlled during retrieval or recognition permitting flexible grouping of stored sequences into broader classes during readout.

1 Introduction – Temporal Sequences

Temporal sequences arise naturally when one attempts to process any time domain signal, for the purpose of recognizing or predicting the presence of features relevant to the particular application. Human speech, biomedical signals, music, or any function of time originating from a sensor constitutes a temporal sequence, whose meaning or intelligence content depends not only on the existence of certain features, but also on their particular temporal order. A time domain signal is most conveniently abstracted as a temporal sequence of feature vectors, often called a *spatio-temporal sequence*,

$$S = \{\vec{s}(t_i)\}, \quad t_i = 1, 2, \ldots, \infty.$$

where each vector $\vec{s}(t_i)$ is a feature vector, a condensed representation of the signal in the vicinity of time t_i. The feature vectors originating directly from speech and signal applications usually span a space of high dimensionality e.g., R^{15}. In practical applications, the infinite number of possible feature vectors are transformed and reduced to a finite number of classes by a clustering operation[1] to reduce the number of distinct vectors to a

[1] A self-organizing feature map or an ART network are examples of input transforming devices of connectionist type.

manageable value, thus providing a *transformed input* for the temporal sequence processor. This transformed input of feature vectors are conveniently viewed and symbolized as characters from a finite alphabet, $\mathcal{A}$. Hence, the temporal sequence, S, is abstracted to appear, for example, as **cjkltprmmbos**. The actual cardinality of the finite alphabet, if too large, can become a major consideration but this is a question of scale which we will ignore.

2 The Objectives of Temporal Sequence Processing

There are two distinct ways that we want to deal with sequences and the associated algorithm or device whose design we are pursuing here. Our primary motivation is feature extraction from signals as a foundational procedure for semantic content discovery.

1. Embedded Sequence Recognition(ESR)

   A number of short pattern sequences, $(ps_1, ps_2, \ldots) = PS$, are stored in the device as shown in Fig.1. An unbounded argument sequence, ARGSEQ, is compared to the set, PS, contained in the sequence memory. When a subsequence of ARGSEQ is found which matches any member of the set, PS, to within a prespecified accuracy, a successful recognition is indicated. As an example of this case in speech recognition, the PS are some transformed representation of phonemes or quasiphonemes, and the ARGSEQ is a similarly preprocessed speech signal.

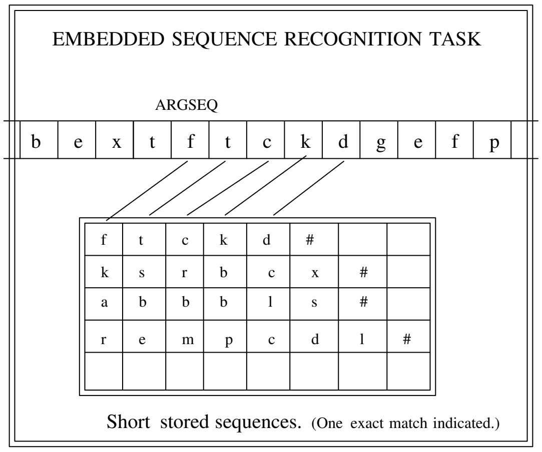

Figure 1: The task defined as embedded sequence retrieval is the location of stored subsequences within a long argument sequence such as an unbounded string.

2. Sequence-Addressable Sequence Memory(SASM)

   'Long' sequences, ST, are stored(see Fig.2). Short 'address' sequences are to be compared to ST for the purpose of locating regions of ST which match the applied address sequence(call this the Address Comparison subproblem). If a sufficiently exact match is found, we want to recognize the condition and possibly read out the continuation of ST from the point of successful match, or, possibly, find the beginning of ST and read the whole stored sequence. This case is the traditional CAM problem, modified by the requirement of *sequential* presentation of the address and *sequential* readout from the match point.

   Address Comparison is a subproblem which is distinct from the continuation of readout of a sequence from the point that the address comparison ends. Address Comparison is accomplished by *guided sequence retrieval(GSR)* while the continuation of readout if a *free sequence retrieval(FSR)*. The sequence retrieved during FSR can be seen as a prediction operation, a case of primary interest with financial data, recall of a musical melody, and so forth[Weigend 93].

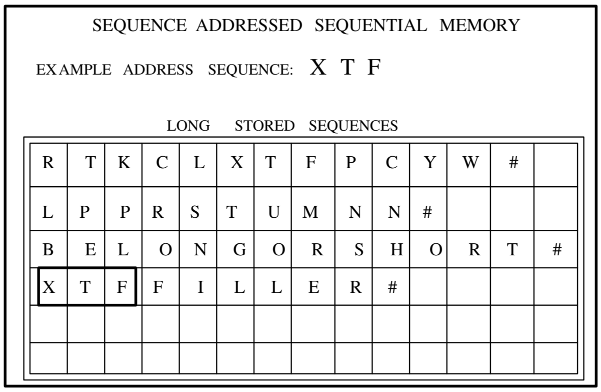

Figure 2: The task defined as sequence-addressed sequential memory is defined as the location of short sequences within long stored strings, followed, possibly, by retrieval of the identified sequence.

In Guided Sequence Retrieval, which occurs in ESR and the AddrComp operations, the comparison of the external sequence to stored information involves a "guided" inquiry where the external symbol, s_i, is presented and the basic question is whether the transition, $s_{i-1} \rightarrow s_i$, is permissible with sufficient strength.

In the FSR problem, there is no external guidance (hence, "free" retrieval) and the internal transition information determines what transitions correspond, in fact, to stored sequences. These distinct operations will be discussed in some detail in later sections.

2.1 Categories of Sequences to be Distinguished

There are a number of types of sequence similarities as well as differences which can occur in the wide variety of applications which we desire to deal with here. To facilitate the discussion, we define an irredundant sequence is a sequence in which, for two consecutive symbols, s_i and s_{i+1}, there is no value of i such that $s_i \equiv s_{i+1}$. Note, however, that under the assumption of recoding [Andreyev 92], any redundant sequence can be mapped into an irredundant one by rewriting. Thus, $s_i s_{i+1} \rightarrow \alpha$ will rewrite any two adjacent symbol occurrences by a new symbol, α. Recursive application of this procedure can transform any redundant sequence into an irredundant one. Nevertheless, since it may be practically desireable to use redundant sequences in some cases e.g., to limit the total number of symbols in use, we will assume they do exist.

Two other terms, *depth* and *resolution* need definition.[2] Depth is the number of symbols or time quanta over which a symbol occurrence influences future symbols. If the maximum depth is N, s_i is effective in predicting s_{i+N} but not s_{i+N+1}. Resolution is a measure of the ability of the sequence processing system to distinguish adjacent symbols in the sequence (including successive redundant symbols).

What are the capabilities that we desire that a temporal sequence processing network have? We enumerate them next.

Desireable Qualities of the Network

1. At a minimum, the network must be capable of treating an irredundant sequence, ST_1, as distinct from another irredundant sequence, ST_2, when the two sequences differ, strictly, in one or more character locations.

   Examples:
   abcdefgh $\neq$ **abcxefgh,**
   abe $\neq$ **ace**

2. In irredundant sequences, the network should be able to predict a successor symbol based on a predecessor symbol when intervening characters are identical. (This is a question of 'depth').

   Examples:
   axyzn, bxyzp
   crotspy, drotspz
   In spite of the intermediate subsequence, **xyz** in the first case, and **rotsp** in the second case, the lead character must predict the final character.

3. The network should be able to predict a successor based on the *count* of the number of appearances of a symbol or symbol group in either redundant or irredundant sequences.

[2] Depth and resolution are discussed at length in [de Vries and Principe 92, Mozer 93].

Examples:
(redundant) **abbg, abbbh, abbbbj, ccdddde, cccddddf.**
(irredundant) **abcdedeg, abcdededeh, abcdedededej.**

The network should be capable of distinguishing any one of the presented sequences from all others.

4. On the contrary to cases 1,2, and 3, the network should be able to *equate* sequences which are closer than some prescribed closeness to one another. Stated alternatively, the resolution should be adjustable.

5. It should be possible to locate (address) any stored subsequence, identify the complete sequence and retrieve the continuation of the sequence.

6. (Storage Efficiency) The network should be able to store long sequences efficiently. The objective is to have a large ratio of (number of stored transitions/number of cells or storage elements). High efficiency implies that a single cell should appear in many different sequences or many positions in the same sequence without sacrificing the resolution.

7. (Temporal Flexibility) The network should (1)tolerate variation in the absolute time scale and (2) in individual (local) variations in time separation between successive symbols. The degree of tolerance required is application dependent.

8. In some cases, storage of the relative temporal length of each symbol is desireable.

We will show that the network proposed here largely satisfies the objectives stated above although compromises among some features are unavoidable. Mainly item 5 falls somewhat short of desired goals.

2.2 The Fundamental Network Properties

What exactly are the properties that a network must possess to achieve the outward features listed above (in Sec. 2.1)?

A careful analysis of this question yields instructive insights on the subject of TSP.

First, in order to have maximum efficiency(item 6 above), we want to minimize the number of cells (or neurons) per symbol times the average number of weights per cell. In our case, we choose to use only one cell per symbol of the alphabet and control the number of weights per cell by a user-defined resolution tolerance. Secondly, to be able to distinguish sequences such as **bbbh** from **bbbbg**, *each transition must be distinct* e.g., the transition from the 2nd to the 3rd 'b' must result in a different internal state from that produced by the transition from the 3rd to the 4th 'b'. This implies that the one cell which identifies with the symbol, 'b', must be able to resolve the difference, however it may be represented, between n and n+1 previous occurrences of a symbol up to some maximum depth, D. (See [Cleeremans 93] for a thorough discussion). But, thirdly, to satisfy temporal flexibility (item 7), the ability of a cell to distinguish between the nth and (n+1)th previous occurrences of a symbol must not be rigidly dependent on the absolute or relative time scale.

Finally, the desire for adjustable resolution (item 4) translates to a need for an *adjustable acceptance window* so that the transition from the nth to the (n+1)th previous occurrence of a symbol can be either recognized or ignored according to a global control parameter. Any practical TSP is subject, also, to limitations on the total storage capacity as well as resolution-depth product.

Given these crystallized requirements for TSP, the merits of various proposed systems can be judged for breadth of capabilities.

We will show that the TSP proposed here does satisfy all of these requirements except parts of requirement 5, the addressing of internal subsequences.

2.3 Previous Work on Sequence Recognition by Networks

An excellent summary article on temporal sequence processing has been published recently by Mozer [Mozer 93]. The conclusion regarding recurrent feedforward network architectures is that they are extremely limited in depth, to the degree that they are impractical for most applications. Cummings [Cummings 93] concludes that there is no straightforward way for these networks to exhibit the property of counting. Our own experiments support this conclusion.

The recurrent cascade correlation network(RCC) of Fahlmann[Fahlmann 90] is another interesting approach to TSP which is actively constructive as is its parent, the cascade correlation network. It learns the Reber grammar efficiently and processes the test set with very low error rate. A major limitation of RCC is that its time delays are fixed which limits its usefulness with time-variable inputs (see item 7 of Sec. 2.1). Temporal flexibility or smearing is an important feature which we have noted in item 6.

In order to provide depth for the TSP, the delay mechanism can be represented by convolution. The convolution of the past values of an output in a 'delay kernel' brings the past history of a sequence into effect for the prediction of the succeeding output in a very general way. An additive convolution model giving the change expected at an output, x_j, can be written[de Vries and Principe 92]:

$$\tau \frac{dx_i}{dt} = -x_i + \sigma \left(\sum_j \int_0^t w_{ij}(t-s) x_j(s) ds \right) + I_1$$

In this continuous model, the net input is given by the quantity in large ()'s. The weights forming the convolution kernel are functions of time. The gamma model is a time delay neural net with trainable delays. It is suited especially to the problem we have called Embedded Sequence Retrieval where an argument signal streams past the recognition device.

An exceptionally interesting approach to modelling temporal smearing by simple delay kernels forms the basis of the TEMPO 2 model [Bodenhausen and Waibel, 91]. It makes use of an adaptive gaussian kernel to capture history by distributing the 'trace' of a symbol in time.

We have observed, however, that it is not necessary to specify the shape of the delay kernel in order to get a gaussian kernel. The effect of time can be introduced in a more fundamental, biologically plausible way. If we view the effect of time as an immediate consequence of spatial behavior, a gaussian kernel can be generated by simple *diffusion*. Diffusion is a natural physical process for chemical substances and, as stated by Katz[66], an important component of information processing in the central nervous system is *the emission of amounts of chemical transmitter substance(s) on arrival of an action potential*. This observation is a guiding principle in the present reaction-diffusion approach. Diffusion, of course, also underlies the time-dependent trail-off of action potentials, characterized by the diffusion of chemical ions, in the Hodgkin-Huxley model of neural activity [Hodgkin and Huxley, 52].

3 The Biological Basis

In 1952, Alan Turing made the important discovery that spatial concentration patterns can be formed if two substances with different diffusion rates react with one another. Moreover, Turing quantified the process, called a *reaction-diffusion* process, in the form of a pair of nonlinear differential equations expressing the concentrations of two "morphogens". Turing's equations had the form [Turing 1952]:

$$\frac{\delta g_1}{\delta t} = a_1 g_1 + b_1 g_2 + c_1 \frac{\delta^2 g_1}{\delta x^2} \tag{1}$$

$$\frac{\delta g_2}{\delta t} = a_2 g_1 + b_2 g_2 + c_2 \frac{\delta^2 g_2}{\delta x^2} \tag{2}$$

$g_1(x,t)$ and $g_2(x,t)$ are the morphogens which may be regarded as very similar, but distinguishable, chemicals. Eqn. 1 expresses the growth of g_1 in time as a function of the quantity of <u>both</u> g_1 and g_2 as well as the diffusion rate of g_1. The second of Eqns. 2 expresses the growth of g_2 in an essentially symmetric way (except as adjusted by the selection of constants). The important observation was that this type of growth system including diffusion could give rise to concentration maxima or focussed regions of activity. If, for example, $b_1 >> 0$ and $a_2 >> 0$, the growth of g_1 causes the growth of g_2 (lateral activation) and the growth of g_2 pushes the growth of g_1 upward. In engineering terminology, we have positive feedback which could result in local concentrations of g_1 and g_2 in the physical vicinity of one another. The downside is that the system, as hypothesized, is unstable. But growth (autocatalysis) of the primary morphogen can be combined with overall stability by causing the secondary morphogen to inhibit (lateral inhibition) the primary over a wide range. Details are discussed at length by Hans Meinhardt [Meinhardt 1982].

In the interest of pursuing our goal more directly, however, we will next discuss the specialization of Turing's reaction-diffusion concept to modelling the growth of sequences of structures by mutual induction of locally exclusive states.

3.1 Generation of Sequences of Structures by Mutual Induction of Locally Exclusive States

When an interior segment of an insect leg is surgically removed and regrafted, there results a regrowth of the interior segment. In some cases the leg grows back to its original form; in other cases, the regeneration is not the original form but is nevertheless controlled by some regular rules. For our limited purposes, it is sufficient to consider only the case of regeneration of the original structure. Fig.3 illustrates the experiment [Bohn 1971; French 1976].

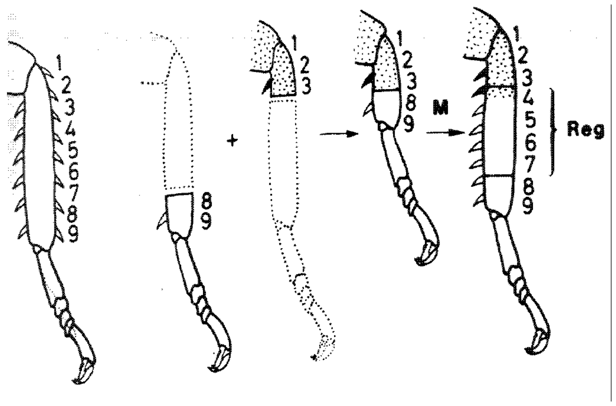

Figure 3: An experiment which demonstrates the regeneration of sequential segments of cockroach tibia. Correctly ordered regeneration of segments is explained by the reaction-diffusion model (from [Meinhardt 82]).

A cockroach tibia is originally constructed as a sequence of similar but not identical segments numbered 123456789 in proximo-distal order. If segments 4567 were surgically removed and the remaining portions of the tibia grafted together to form the sequence, 12389, after one or two moults(M) the missing elements are found to regenerate. Further experimental evidence shows that most of the regenerative information comes from the more distal elements, that is, 8 induces 7, 7 induces 6, etc. The obvious question is how the positional information is encoded and how it expresses as the correct sequence. The abstract quantitative explanation is contained in an extension of the reaction-diffusion concept.

3.1.1 Explanation of the Generation of Sequences

A well substantiated quantitative explanation of the cockroach tibia regeneration observation [Meinhardt 82] is expressed in equations 3-6 and in Fig.3.

$$\frac{\delta g_i}{\delta t} = \frac{c_i g_i'^2}{r} - \alpha g_i + D_{g_i} \frac{\delta^2 g_i}{\delta x^2} \tag{3}$$

$$\frac{\delta s_i}{\delta t} = \gamma (g_i - s_i) + D_s \frac{\delta^2 s_i}{\delta x^2} \tag{4}$$

$$\frac{\delta r}{\delta t} = \sum c_i g_i'^2 - \beta r \qquad (5)$$
$$g_i' = g_i + \delta^- s_{i-1} + \delta^+ s_{i+1} \qquad (6)$$

g_i represents the concentration of the ith morphogen, a set of distinct but closely related genes (which will translate as "characters" in our later scheme of temporal sequences). s_i is the concentration of the ith co-morphogen effectively at a discrete location, x_i. r is the concentration of a common inhibitor substance. All other quantities are assignable constants, the D's being the diffusion constants. Note also that the diffusion terms could be written in finite difference form as, for example,

$$\frac{\delta^2 g_i}{\delta x^2} \Rightarrow g_i(x_{i-1}) - 2\ g_i(x_i) + g_i(x_{i+1})$$

as did Turing, in fact, in his original formulation. A diagrammatic expression of this set of coupled equations is presented in Fig.4 where '+' implies a flow of growth-aiding substance and '-' implies an inhibiting flow of substance.

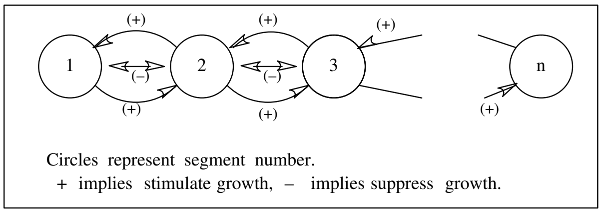

Figure 4: The presence of segment k may stimulate growth of segments k-1 and k+1 (+ arrows) depending on the values of parameters. (-) indicates the inhibitor or reactant chemical, limiting and stabilizing growth (from [Meinhardt 82]).

The positive feedforward from, for example, pos. 2 to pos. 3 is the helper morphogen s_2 which grows according to eqn.4 and appears at pos. 3 through the δ^- term in eqn.6 as a positive contribution to the first term of eqn. 3. Growth of g_3 is also aided by a positive diffusion of its own substance. Meanwhile, the generation of the common mutual inhibitor, r, is expressed in eqn.5 and its repression of substance g_3 acts through its appearance in the denominator of term 1 of eqn.3. Substance r also suppresses other would-be spurious states from expressing. In the case of the regeneration of the cockroach tibia which grows the sequence 7654, it would be necessary to have $(\delta^+ > \delta^-)$ to cause growth of g_7 due to the presence of s_8 rather than trigger growth from segment 3 to 4 first.

Thus in this biochemical reaction-diffusion scheme, a structure sequence is believed to grow by a combination of forward activation and diffusion of characteristic morphogens or substances, stabilized by lateral inhibition through a common, widely-diffused substance.

Our next step is to formulate a system analogous to the natural biological one described above and to investigate its properties when applied to the temporal sequence processing problem. We will not have to invent a special function to forward information from earlier to later times.

4 Reaction-Diffusion Temporal Sequence Processing Architecture

We will now outline an architecture which parallels the reaction-diffusion process described above, but which necessarily takes some liberties with the definition of the detailed properties of the cells of which it is constructed. First let us define the basic cell.

4.1 The Single Cell Model

Each cell is defined as follows (see Fig.5, inset).

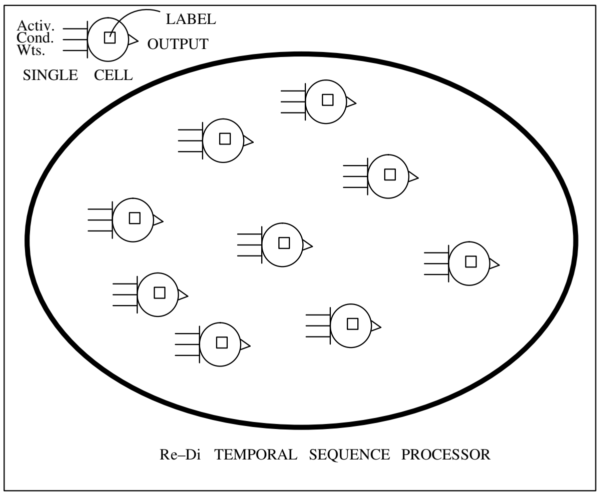

Figure 5: Architecture of the Temporal Sequence Processor. Identical cells distributed randomly in a 3-d (or 2-d) volume supporting diffusion
.

- Each cell may have one label. We assume all cells have blank labels initially and any cell can acquire and permanently be identified with a single character from the alphabet of possible symbols, $a_i \epsilon \mathcal{A}$.

- A cell with label a_i, at the moment of [1]activation, emits one unit of chemical designated, g_{a_i}, which increases the concentration of g_{a_i} locally by 1.

- A cell at position x is activated as follows. Let $\vec{g}(x)$ be the chemical concentration vector at position x. The cell has a set of receptors, $V_j, (j = 0, \ldots, K-1)$, which activate when the chemical environment equals (within a tolerance, σ) the value of V_j. More precisely, cell C is provisionally activated if $\min_j \sum_k \mid (g_k(x) - V_{jk}) \mid < \sigma$

  where k indexes the types of chemicals (the distinct a_i). The receptors, V_j, are capable of 'learning' or storing the value of the local chemical concentration, similar to the training of weights for an artificial neuron model.

4.2 The Architecture

The TSP consists of N simple cells distributed on a two-dimensional plane or in a 3-d volume and immersed in a 'fluid' which supports diffusion of the morphogens (Fig.5). Although the distribution is fundamentally random, we will assume a regular spacing on a rectangular grid purely for computational convenience.

The characteristic chemical or morphogen which each cell emits is identified one-to-one with the external symbols comprizing the sequences to be stored and processed. We will assume only one cell in the processor, C_p, emits chemical g_p. The cell is labelled 'p'.

Each of the cells is capable of being activated by the existence of specific chemical concentration vectors as described in the previous section. For a cell, C, at spatial position, (x,y), C is activated when:

$$AL = \min_j \{ (\mid \vec{v}_i - \vec{g}_i(x,y) \mid) \} < \sigma \tag{7}$$

where

$$\vec{g}(x,y) = g_1(x,y), g_2(x,y), \ldots, g_p(x,y), \ldots, g_K(x,y)$$

$$\vec{v}_i = v_{1,i}, v_{2,i}, \ldots, v_{K,i}$$

and AL is the activation level, and σ is a threshold which may be regarded temporarily as constant. Activation is a competitive activity– only one cell is the "active cell" at any moment. When a cell is activated and becomes the active cell, it emits a unit impulse of its characteristic label chemical, g_p, into the common 'fluid' environment at its location, (x,y), which becomes distributed by diffusion.

After a period of activations of a number of different cells at various times and various positions, the emitted chemicals have been distributed by diffusion, generating a chemical soup carrying information about the history of activations thoughout the entire system. The morphogen concentration stimulates emission of a common reactant chemical, r, which serves to regulate and stabilize the growth of chemical concentrations.

All diffusion of the chemicals, including the reactant chemical, is determined by the reaction-diffusion equations.

Specifically, the following set of equations governs generation and movement of chemicals.

The Reaction-Diffusion (Re-Di) Equations

$$\frac{\delta g_p(z)}{\delta t} = \frac{\epsilon g_p(z)}{r} - \alpha g_p(z) + D_g(\frac{\delta^2 g_p(z)}{\delta z^2}) \tag{8}$$

$$\frac{\delta r(z)}{\delta t} = \gamma \sum_p g_p^2(z) - \beta r(z) + D_r(\frac{\delta^2 r(z)}{\delta z^2}) \tag{9}$$

For the purposes of simulation of the spatial dimension of the reaction diffusion equations, we write the distance between x_1, y_1 and x_2, y_2 as $z = (\mid (x_1 - x_2) \mid + \mid (y_1 - y_2) \mid)$, that is, the manhattan distance with the 2nd derivatives written as discrete central derivatives.

The first equation appears K times, $p = 1, \ldots, K$ or once for each distinct chemical. The reactant chemical, r, emitted by some unspecified mechanism additional to the neuron-like cells, stabilizes the whole system by inverse feedback proportional to the sum of squares of all chemicals present.

4.3 A ReDi Temporal Sequence Processor

4.3.1 Training the ReDi TSP to Store Short Sequences

A training or storage sequence, $s_1 s_2 s_3 \ldots$, is presented at integral time points.[3] After the first occurrence of a symbol, s_i, a permanent one-to-one correspondence is established to one unique cell which is labelled, s_i, and which emits chemical g_{s_i}. Given any symbol in the alphabet, it is assumed that the corresponding cell, if it exists, can be located.

The Storage Algorithm

The external sequence to be stored is $s_1 s_2 s_3 \ldots$. All cell labels are initially blank. The number of cells is N. A cell is located at point z. There is a unique active cell (AC) after operation begins.

1. Set all $\vec{V}_i = 0, \quad i = 1, \ldots, N$.

2. Set all chemical concentrations, $g_i = 0$ (at beginning of each sequence).

3. If symbol is s_1.

   (a.) If no cell has label s_1, recruit a blank cell, label it s_1, mark it the active cell(AC), emit 1 unit of chemical g_{s_1} and step the ReDi equations.

   (b.) If a cell is already labelled s_1, mark it AC, emit 1 unit of g_{s_1} and step the ReDi equations.

4. If symbol $s_j, \quad j > 1$.

   (a) If no cell is labelled s_j, recruit an unused cell randomly, label it s_j, mark it AC, set $\vec{V}_i = \vec{g}(z)$ where V_i is an unused register of AC, emit 1 unit of g_{s_j}, and step the ReDi equations.

   (b) If some cell, m, is already labelled s_j, then

      (1) if one of its V registers is *activated*, then the transition is already stored(KNOWN). Mark cell m as AC, emit 1 unit of g_{s_j}, and step the ReDi equations.

[3] Integral time units do not imply an absolute or rigid time scale, but as in digital integration, is the fundamental unit of time.

(2) else if no V registers are activated, then the transition is UNKNOWN. Set an unused register, $\vec{V_i} = \vec{g}(z)$, mark cell m as AC, emit one unit of g_{s_j}, and step the ReDi equations.

5. Return to step 4 until the end of the current sequence is encountered.

6. Return to step 2 until all sequences have been stored.

Cell m is activated when

$$\min_i ||\vec{V_i} - \vec{g}(z_m)|| < \sigma$$

for $1 \leq i \leq K$ where K is the number of activation condition registers per cell.

The conceptual explanation of this algorithm is diagrammed in Fig.6. Briefly, each symbol in the applied sequence causes the emission of a characteristic chemical which diffuses spatially at some rate. The successive symbols build a chemical vector which varies with both time and space. History of the sequence is distributed throughout the volume of 'tissue' where it can be associated to the successor symbol during storage. Later, when testing to detect the existence of a stored sequence, the chemical history during training will be recapitulated, approximately, by 'guided sequence retrieval'.

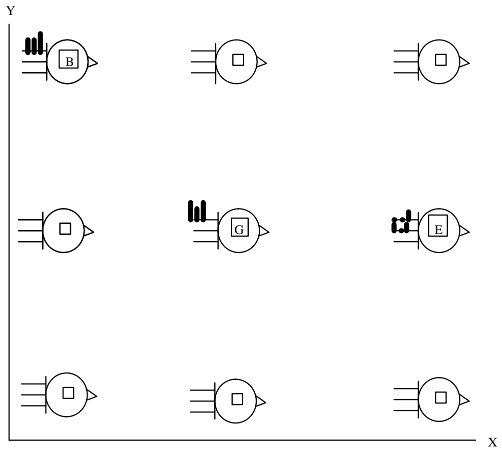

Figure 6: How the TSP Works. V-registers or weights(vertical bars) store the chemical levels which permit activation. Chemical concentrations diffuse throughout the medium.

The storage or training algorithm can be further illuminated by following the storage of a sequence step by step.

4.3.2 An Example of the Training Algorithm in Use

Take the case of a virgin device to which the very first sequence to be applied is **'cdddf'**. We will assume that the matching tolerance, $\sigma = 0$, as defined in eqn. 4.1. The architecture is a square 5x5 two-dimensional grid of cells. Rows are numbered top to bottom, 1 to 5, and similarly for columns, left to right. Initially all chemical vectors, $\vec{g}$ and r are initialized to zero.

Cell (2,2), the central cell, is first recruited[4] and labelled "c" (chemical #2). It becomes the active cell(AC). One of its V registers is set to zero, the amounts of the respective chemical concentrations, $g_i(0)$, present at (x,y) = (2,2). The Active Cell(AC) emits one unit of chemical #2. The Re-Di equations are then applied for one time quantum of diffusion and decay, causing chemical #2 to diffuse to adjacent cells. For example, $g_2(2,1) = g_2(2,3) = 0.25, g_2(2,2) = 0.88$ at t=1.

The 1st 'd' is presented. A cell is recruited, say Cell(2,3), labelled "d"(chemical #3), marked the Active Cell, and one of its V registers is set to $(0, 0, 0.25, 0 \ldots)$, the current local concentration vector at the coordinates (2,3). The Active Cell emits one unit of its label chemical, "d". The Re-Di equations are stepped, producing a modified set of chemical concentrations. Only chemicals #2 and #3 are non-zero. The reactant level grows also as the total morphogen ($\vec{g}$) level grows, acting to constrain the growth of the 'morphogens', g_i.

When the 2nd 'd' is presented, the active cell remains Cell(2,3) already labelled "d". But now the chemical concentration is different for this 'd' as compared to the first 'd'. Consequently, an unused V register of AC is recruited and set to the local conditions, for example, $(0, 0, 0.35, 0.88, 0, \ldots)$, only morphogens 2 and 3 having occurred. After diffusion, the amount of "c" chemical decays[5]while the "d" chemical continues to grow due to the injection of one additional unit of g_d at (x,y)= (2,3) each time the 'd' labelled cell is triggered in the role of Active Cell.

Consecutive d's in the input sequence continue to reactivate the same cell but the chemical concentration changes. Each 'd' is different and therefore *countable*. All the while, r acts to control concentration growth quite naturally.

Finally, when the 'f' is presented, another cell, say Cell (1,4) is recruited, labelled 'f', and made the AC. One of its V registers records the chemical concentration vector,$\vec{g}_i(1,4)$, at the time the cell is made AC so that when the same condition arises later, the same cell will be activated. This is the end of training or storage of the sequence 'cdddf' on its first presentation.

If any subsequence presented to the network for training coincides with a previously presented sequence e.g., **cddddf**, the network would follow exactly in the previously established path for the 1st 4 characters, 'cddd', activating the the same V registers which were trained during the first occurrence of 'cddd'. But when the 4th 'd' is presented, Cell (1,4) (labelled 'd'), would continue to be the Active Cell but activated by a unique level of chemical environment which would require an additional V register of Cell(1,4) to record it. Note that each 'd' is distinctly recorded so long as $\sigma = 0$ *so the <u>count</u> of recurrences of any symbol affects the future.*

Assuming $\sigma = 0$ and the chemical concentrations are totally deterministic(noiseless) and precise numerically; every transition in any sequence is recorded distinctly. *Resolution* approaches 0 so *depth* is unbounded when $\sigma = 0$ and assuming indefinitely precise, noise-

[4]In the actual algorithm recruitment is always random in location.

[5]The constants of the ReDi equations may be chosen to increase chemical concentrations for some time intervals so monotonic decay is not imperative.

less (and, therefore, deterministic) values of $\vec{g}_i$ and of the activation condition vectors, $\vec{V}_j$.

But if $\sigma > 0$ and sufficiently large, then the 4th appearance of 'd' may permit activation of Cell (1,4) by a previously set V register value, thus storing **cdddf** as equivalent to **cddddf**. The depth will be finite, because, at some point in the presentation of a repeated symbol, for example, the chemical concentration change from the mth to the (m+1)st appearance of 'd' will be less than σ so that the two adjacent appearances of 'd' will activate the same cell *by means of the same activation condition vector, V.* At the same time, fewer distinct activation vectors are used thus requiring lower storage or information capacity.

The generalization permitted by adjustment of σ during storage determines whether two sequences with a common internal subsequence are stored uniquely. But even if two such sequences are stored uniquely, it is possible to treat them as equivalent during retrieval by adjustment of the activation tolerance, σ_{TST}, but it is not possible to retrieve them uniquely if they were not stored uniquely. σ_{TST} is defined as is σ in eqn.4.1 but distinguished only by the operational condition, storage or testing(retrieval).

Storage with $\sigma = 0$ is analogous to eidetic memory.

4.3.3 The Network in the Embedded Sequence Recognition Mode

Suppose many short sequences are stored in a Re-Di TSP (as in Fig.1). Let a long argument sequence be presented beginning at some particular symbol, s_k, for the purpose of detecting whether the argument subsequence is stored in the network.

Assume the period of symbol presentation is one time step. The algorithm is:

Guided Sequence Retrieval

1. Set k=1. Clear all chemical concentrations to zero.

2. Find the cell, C_j, whose label is s_k. If none, terminate with failure.

   If there is an activation register of C_j such that

   $$\sum_i \mid (g_i(x) - V_{ji}) \mid \leq \sigma_{TST}$$

   then C_j becomes the new AC (active cell) and the transition from $s_{k-1} \rightarrow s_k$ is a *Known Transition.* Emit one unit of chemical s_k at the location of C_j, and perform one step of the ReDi equations. Else, if no cell satisfies the inequality, then the external sequence is known only from s_1 to s_{k-1}. Terminate the test.

3. If a Known Transition occurred, increment k and return to step 2.

The foregoing algorithm of the test phase recapitulates the training steps in the sense that the chemical conditions present at storage are revivified during the *successful* matching of the argument sequence to an originally stored sequence. Notice however that we are permitting a different activation tolerance measure, σ_{TST}, than we used during training.

As σ_{TST} rises from 0, the strictness of matching the original conditions during storage of a sequence is relaxed, eventually reaching the level that any external sequence appears to be an acceptable match.

5 Experiments

5.1 Count (or Depth) Test

The ability of the network to predict a successor based on count can be tested directly by demonstrating that the sequence, $\mathbf{cd^ne}$ can be distinguished from $\mathbf{cd^{n+1}f}$, that is, the occurrence of n **d's** is distinct from n+1 **d's**. An equivalent test consists of storing a sequence such as **cddddddd...** and noting the point where successive d's are not distinguished by utilizing a new condition register, V_i. This experiment was performed with various values of tolerance, σ. The results are shown.

COUNTING TEST

σ	n (largest depth)
.05	3
.04	4
.02	6
.01	10
0	∞

Metaphorically, larger values of σ correspond to paying less attention to the precise count. As the count or depth increases, the penalty is a proportionate increase in the required storage capacity which can be defined as: Capacity = $N_c N_V$ where N_c = the number of cells and N_V = the average number of occupied activation condition registers (V-registers) per cell. (See sec.6 for more discussion of storage capacity).

Now let us look at an example of the operations of training and test for a Reber grammar.

5.1.1 An Experiment with Embedded Reber Grammar

A finite state Reber grammar has been strengthened as a test for depth and prediction in recurrent neural nets by embedding copies in correlated beginning-ending phrase (Fig. 7)[Fahlman 91].

An embedded Reber grammar was encoded with transition probabilities all set to 0.5. The average sentence length generated is about 20 symbols, with the maximum of about 30 symbols in a sample of 500 sentences.

A ReDi TSP with 25 cells in a 5 x 5 grid was defined. Five hundred strings were generated and applied to the TSP in the training or storage mode. Since the grammar uses only 7 symbols, there were 18 unused cells.

Varying the activation tolerance measure, σ, demonstrates the range of generalization possible during storage . In the limit as $\sigma \to 0$, every transition with a unique history is

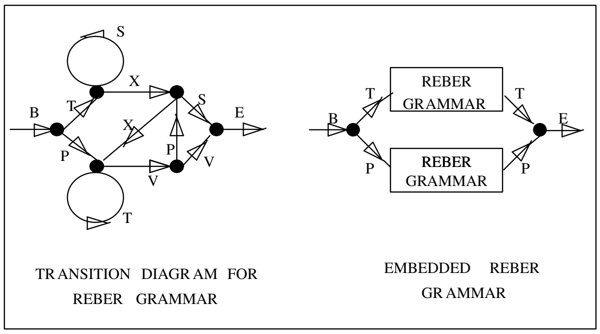

Figure 7: Embedded Reber Grammar. All non-unique transition probabilities set to 0.5
.

stored as a unique transition. The first appearance of a distinct transition is signalled as "UNKNOWN TRANSITION" which requires recruitment of a V-register. As σ increases, some transitions are treated as equivalent by the system. For example, when $\sigma = .05$, **bppttvvpe** is stored as equivalent to **bppttvvpe**. Higher values of σ result in automatic generalization eventually generalizing to the point that every transition is context-free.

Similarly, increased values of activation tolerance during retrieval, σ_{TST}, permit more tolerance in the affirmative decision that a particular transition with its (possibly) extensive history is permissible.

The accuracy as a function of both σ's is plotted in Fig.8.

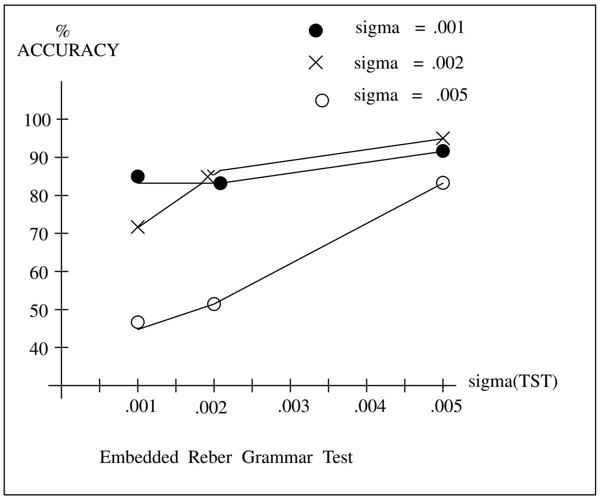

Figure 8: Accuracy of predicting Reber grammar strings as a function of σ, the storage tolerance, and σ_{TST}, the test phase tolerance
.

To retrieve a sequence, a randomly generated Reber sentence is presented sequentially by symbol and retrieval is attempted by the Guided Sequence Retrieval algorithm. The retrieval is counted as accurate only if every symbol transition in the sentence is found permissible within the bounds, σ_{TST}, that is, every transition is a "KNOWN TRANSITION". The number of activation condition registers required as a function of σ is displayed in Fig.9.

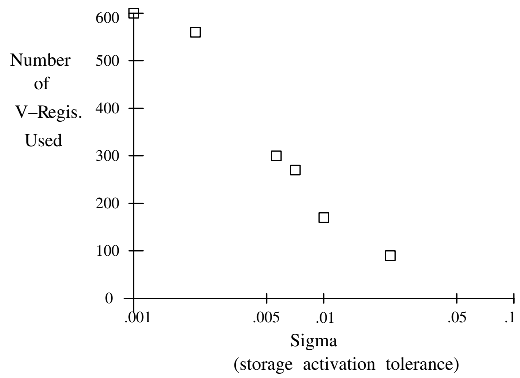

Figure 9: Number of activation condition registers (V-registers) used as a function of σ. Total symbol transitions were 4492.

5.1.2 Time Flexibility

In the preceding tests, the time step was constant at unity during storage and test. This condition permits the retrieval steps to generate exactly the same sequence of chemical concentrations as the network experienced during storage.

The ability of the network to tolerate time variation was demonstrated as follows. The sequences generated by the Reber grammar generator were stored with unit time steps which is to say, one diffusion step was promulgated for each input symbol transition. During retrieval, however, two or three diffusion steps were applied for each transition, thereby effectively making the retrieval time scale two or three times slower than the scale used during storage. The stored sequences could all be recalled by relaxing the activation tolerance, σ_{TST} (see Fig.10). In doing so, one is effectively permitting more error in readout much as one might accept a mildly erroneous instance of a statement during recall as equivalent to the original. For example, "He who has the last laugh, laughs best" might be an acceptable remembered equivalent of the originally stored aphorism, "He who laughs last, laughs best".

Using the activation tolerance as in this test has the effect of binding time flexibility to equivalence class which may be undesireable in the case where one wants time flexibility but precision readout. That is, the larger is the required relative time, the broader is the class of sequences treated as equivalent.

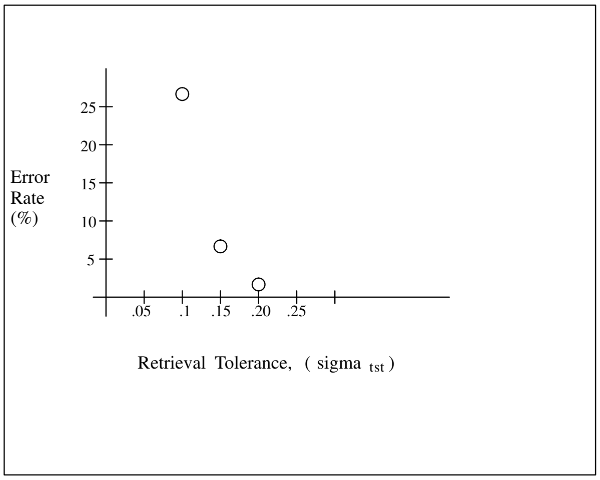

Figure 10: When the retrieval time scale is 2 times slower than the rate applicable during storage, the retrieval tolerance must be increased to lower the error rate.

5.1.3 Recoding to Decrease Depth Requirement

The concern with large depth is, in one sense, not of ultimate importance because, as has been pointed out [Andreyev et.al, 1992], common substrings can be recoded as distinct symbols. Thus, 'ddd' can be recoded by a different symbol of the alphabet than 'dddd'. Recoding would, of course, increase the number of cells but decrease the number of activation levels per cell. [1]

All of the foregoing examples and discussion has tacitly assumed the case of Embedded Sequence Recognition as defined in item 1 of Sec. 2.

5.2 Sequence Addressable Sequence Memory

A test of the sequentially addressed, sequence memory capability of the ReDi TSP system was performed. Desiring to demonstrate that both symbol(space) and time duration could be stored and recalled by TSP, we selected the universal medium of a musical score as the test item.

First, a system for encoding a single melody of a music score was devised (see Fig.11). A unit time step was chosen. For example, an eighth note equals one relative time unit corresponding to one diffusion step of the ReDi equations. The appearance of a transition in the base symbol ($C \rightarrow D, C \rightarrow C$) corresponds to the emission of a single unit of the corresponding chemical e.g., A,B,C. Note that CC^2 and C^3 result in completely distinct chemical concentrations since the former means (Emit C and Diffuse, then Emit C and Diffuse 2 times), whereas the latter means (Emit C and Diffuse 3 times).

Then, a set of six argument strings of average length 20 transitions was constructed (Fig.12a). The strings have a common core substring making it necessary to have a significant amount of 'depth' to keep the strands separated when being retrieved.

The six argument strings were stored in the TSP by the usual storage algorithm.

Various lengths of the initial parts of the argument strings were used as the address substrings (Fig.12b). The address substrings were applied to the TSP using the 'Guided Sequence Retrieval' algorithm which follows the steps of the storage algorithm except that no V registers are altered.

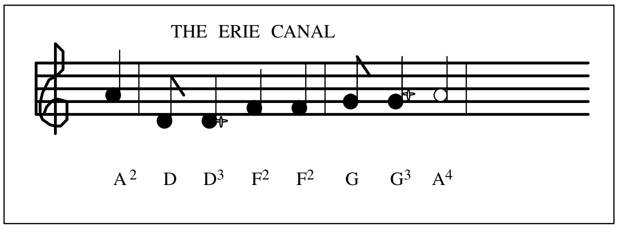

Figure 11: Example of musical melody encoding. Superscript represents time length in eighth notes which translate as integration time steps of the Re-Di equations.

F^2 C^3 A^4 D E F A^5 B C D^5
F^3 G^9 F^2 D E F C^2 B A A
B D F A^4 D E F B^4 C^2 E F G^2
C E G B^3 C D E F D^3 D D^2 F
G^2 A B^3 C D E F E F^3 G
A^2 E D^2 C B^3 D E F F^2 C

'LONG' STORED SEQUENCES

F^2 C^3 A^4
C E G B^3 C D E F D^3

EXAMPLE ADDRESS SEQUENCES
S

Figure 12: (a) Sequences stored in TSP to test combined symbol and time memory in SASM mode.(b) 'Short' addressing sequences.

At the end of the GSR stage, the Free Sequence Retrieval algorithm was followed.
The **Free Sequence Retrieval(FSR) algorithm** is:

1. Initially, a chemical concentration exists at every cell, $\vec{g}(x, y)$, as a result of the preceding Guided Sequence Retrieval. Also an activated cell exists, AC_k.

2. Compute the activation level of each cell, $AL_i \quad, i = 1, \ldots, N$ per eqn.7.

3. In winner-take-all fashion, choose the cell with largest activation as the next active cell, AC_{k+1}.

4. If $AC_{k+1} \neq AC_k$, emit one unit of the chemical, g_{k+1}, since a transition to a different chemical has occurred, and deactivate cell k. Else, no emission occurs since the active cell remains unchanged.

5. Perform one diffusion step. Return to step 2.

Note that whether step 4 results in a new active cell or not, step 5, diffusion, will cause a change in the overall chemical concentration vectors.

5.2.1 RESULTS

In this experiment, the FSR process follows the path of the correct stored sequence, thereby retrieving the stored sequence with one exception dissected below. The action is much like following Ariadne's thread except there are six threads in this experiment, one for each melody, all passing through a common region where only the color (chemical vector created by the earlier parts of the sequence) distinguishes the threads. The FSR algorithm recapitulates the timing and symbol sequence precisely with the one exception of sequence #4.

In the erroneous case, sequence #4 in Fig.12, the address string ends in the midst of the subsequence, D^3, D, D^2, F^1. Each reappearance of 'D' caused an EMIT operation in the storage process. But the FSR procedure can only know to EMIT when there is a *change* of chemical symbol. Since the GSR ends on D^3, the FSR algorithm would find in step 3 that the cell labelled D is the next activated cell, corresponding to D in the sequence. Now, the transition from $D \rightarrow D$ cannot be distinguished from the case of a diffusion step only, *but an Emit + Diffuse occurred in the storage of the sequence at this point.* Hence the chemical concentration vectors in the FSR diverge from that which was stored, causing the trace of the original sequence to be lost. In all cases except this special one, there is no crosstalk between stored sequences – they can be extracted uniquely regardless of common internal subsequences.

In the experiment just described, the addressed string was always started from the beginning of the stored string. The chemical concentrations, $\vec{g} = 0$, initially when both at the beginning of storing the long string and at the beginning of GSR. Therefore the values of g could be matched at every emit and diffusion operation. But when the address string begins in the internal parts of a stored string, e.g. as in Fig.2, there is no way to discover the chemical concentration which accompanied the stored string at the point the address match should begin. This is an open problem at this time. There may be some modification of the architecture which will permit the sequential addressing of an internal substring.

6 Storage Capacity

The power of the ReDi architecture rests on the huge number of states which it can represent. Counting states for one cell, letting
N = number of distinct vectors able to activate one cell,
γ = number of distinct chemicals(= nbr. of cells),
K = number of weight vectors per cell,
L = number of distinguishable levels of concentration of each chemical at a single location, then

$$N = \gamma \ K \ L \tag{10}$$

If $\gamma = 100, K = 100$, and $L = 10$ then N=10^5.

7 Conclusion

We have proposed a temporal sequence processing system based on the well founded biological process of reaction-diffusion. The Re-Di system does not utilize the typical McCullough-Pitts based neuron model. The Re-Di process, in order to be adapted for use for the mechanism envisioned, implies a neuron-like cell which operates primarily by chemical signals in order to capitalize on diffusion.

Any TSP, to achieve the properties observed as desireable in the applications sense, should be capable of a range of properties as specified loosely in Sec. 2.1, of which depth and resolution are the primary parameters. Because the chemical substances diffuse in all spatial dimensions, the system exhibits a time-delay kernel naturally, which handles both depth and resolution very appropriately. Additionally, the diffusion process essentially makes the output of every cell available at the input of every other cell in the same volume, thus providing a fully connected system somewhat reminiscent of a Hopfield network but naturally stabilized by the reactant chemical. In particular, it has been shown that the system is capable of *counting* as exhibited by being able to predict the symbol, z, in the sequence, ad^nz, if there are exactly n occurrences of 'd', but some other symbol if there are n-1 occurrences of 'd'. It is capable of *storing sequences of any length* with only one cell per symbol, assuming that the number of activation vectors per cell (analogous to the arborization complexity of a natural neuron) is unbounded. It is capable of *tolerating variations in both absolute time scale and intersymbol random variation* at the cost of increased error rate. The error manifests as the false appearance that any prescribed sequence is stored when it was, in fact, not stored. Time scale variation is totally flexible by adjustment of a single error tolerance parameter,σ_{TST}, although the error rate would certainly be unacceptable in most practical cases for time scale variations of more than two or three times. All of the foregoing properties involve compromises in the total storage capacity, resolution, and depth which are complex and which has not yet been analyzed. One measure of the total storage capacity of the proposed system is the product (C)(AV), where C is the number of cells, and AV is the number of activation vectors per cell. To some degree, the number of activation vectors is interchangeable with the number of cells, although an increased number of cells introduces some procedural problems since it implies that multiple cells are assigned to the same symbol or else, that a preliminary recoding layer(s) exist. The system stores or learns sequences on one presentation. Adjustment of a single parameter, σ, permits a degree of sequential symbol tolerance which clusters sequences into classes, somewhat like an ART network forming autoclasses under control of its parameter, ρ.

On free or unguided readout of a stored sequence, the most difficult problem is to correctly readout one of two (or more) sequences which have a common region. In our system, even if 'cdvjklp' and 'kdvjklz' are both stored and the common subsequence, 'dvjkl', is stored in the very same cells, the two sequences can be uniquely recovered because the initial occurrence of 'c', in one case, and 'k' in the other case sets up a unique trace in the chemical soup which is never totally dissipated, theoretically. The cost of increase of depth corresponding to very long common subsequences must be paid in increased numbers of activation vectors, thus, storage capacity. The free sequential readout of two sequences with identical common *initial* part, however, such as 'dvjklp' and 'dvjklz', cannot be uniquely distinguished at the 6th character(p or z), of course, since no distinction exists before the difference is encountered at the 6th character. This is analogous to retrieval of stored trees with a common root. Perhaps the most difficult task to achieve for the ReDi network is the identification or addressing of an embedded internal substring of a stored longer sequence. The difficulty arises from the fact that an internal symbol of a stored sequence is associated with its precursory context so that it is necessary to defeat, in some way, the context so as to compare the addressing string with the stored string.

The proposed defeat procedure is to widen the error tolerance for the initial symbols of the argument sequence, gradually narrowing the acceptable error as the matching proceeds along the sequence. This procedure naturally entails the possible error of finding an apparent but false match of the addressing string to some part of a stored string.

REFERENCES

Andreyev,Yu.V., Dmitriev, A.S., Chua,L.O., & Wu,C.W(1992). Associative and Random Access Memory Using One-Dimensional Maps. Int'l J. of Bifurcation and Chaos,**2**,483-504.

Bodenhausen, Ulrich and Waibel, Alex(1991). Learning the Architecture of Neural Networks for Speech Recognition. IEEE Proceedings of the ICASSP,**1**, 117-124.

Bohn,H. (1971) Interkalare Regeneration und segmentale Gradienten bei den Extremitaten von Leucophaea-Larven(Blattaria). III. Die Herkunft des interkalaren Regenerats. Wilhelm Roux Arch. **167**,209-221.

Cleeremans,Axel(1993). Mechanisms of Implicit Learning. MIT Press, Cambridge, MA.

Cummins, F.(1993). Representation of Temporal Patterns in Recurrent Neural Networks. Proc. of the 15th Annual Conference of the Cognitive Science Society, pp. 377-382, 1993.

deVries, B., & Principe,J.C.(1992). The gamma model – A new neural net model for temporal processing. Neural Networks, **5**, 565-576.

Elman, Jeffrey(1990). Finding Structure in Time. Cog.Sci. 14 (179-211).

Fahlmann,S.E.(1990). The recurrent cascade-correlation architecture. In*Advances in Neural Information Processing Systgems, pp.380-387.*

French, V. (1976). Leg regeneration in the cockroach, Blattella germanica I. Regeneration from a congruent tibial graft/host junction. J. Embryol. exp. Morph. **179**,57-76.

Hodgkin, L. and Huxley, A.F.(1952). A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve. J.Physiol.,**117**,pp. 500-544.

Katz,B.(1966). Nerve,Muscle, and Synapse. McGraw-Hill, New York.

Meinhardt,Hans (1982). Models of Biological Pattern Formation. Academic Press, Inc.

Mozer, Michael C.(1993). Neural Net Architectures for Temporal Sequence Processing. In: A. Weigend & N. Gershenfeld(Eds.), *Predicting the Future and Understanding* *the Past*, Addison-Wesley Publ., Redwood City, CA.

Turing, Alan(1952). The Chemical Basis of Morphogenesis. Phil. Trans. of the Royal Soc. of London, Ser. B, **237**,37-72.

Wiegend, A. and Gershenfeld, N.(1993). Time Series Prediction. Addison-Wesley Publ. Co., Reading, MA.