Design and Use of AUTONET, a Logic-Level Neural Simulator

by

R. E. Ellis

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Science
in
The Department of Computer Science

Winnipeg, Manitoba, 1981

DESIGN AND USE OF AUTONET,

A LOGIC-LEVEL NEURAL SIMULATOR

BY

R.E. ELLIS

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

R. E. Ellis

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

R. E. Ellis

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

AUTONET is a system for the specification and simulation of neural networks. It deals with the network as a directed graph, the nodes of which are automata which accept a tree of inputs and produce a single output, and the lines of which are queues of arbitrary length via which the cellular automata communicate.

The simulation system includes the WARP network construction language. The important features of WARP are its ability to manipulate not only single cells, but also arbitrary groups of cells, as distinct units. Projection of one cell group upon another is facilitated by this feature, and also by the ability to permute the interconnections between cells. This permits convergent and divergent projections in many dimensions to be handled with ease.

The experiments conducted with AUTONET have been largely with simple and moderately complex networks. Some apparently trivial networks are shown to actually possess a wide variety of possible rhythmic outputs. These output patterns are characterized as to their stability, and to the connectivity patterns which make the various patterns more or less likely to be generated in a network subject to random effects.

Considerable simulation of the gastric and pyloric subsystems of the lobster stomatogastric ganglion was also done. It is shown that all of the rhythms generated by the ganglion may be explained by decomposing the networks into groups of two or three cells, which smaller networks have already been well-characterized.

Evidence is also presented for the hypothesis that the endogenous bursting cells of the pyloric subsystem are not the generators of the pyloric rhythm. Rather, their purpose seems to be as synchronizers of the cycle.

ACKNOWLEDGEMENTS

I wish to greatly acknowledge the assistance I received from my friend and adviser, Douglas Young.

Thanks also are due to Jon Muzio for both syntactic and semantic observations, and to Patty Churchland for the many stimulating conversations which led to my interest in lobster stomachs.

This thesis was made possible in part by a scholarship from the Natural Sciences and Engineering Research Council of Canada.

Chapter I

INTRODUCTION

Among the many lesser-researched areas in artificial intelligence is the subject of simulating neuronal networks. Just as man has surpassed his body's abilities in other ways, the general feeling goes, so he might be able to do so in the processing of information. A crane does not operate exactly as the arm does, yet lifts greater weights and works for days on end; perhaps by discovering the fundamental principles of thinking (as he discovered the principles of the six simple machines) man might yet construct a device that is truly intelligent.

Viewed in this light, neural simulation seems a curiosity. It seeks to know how things <u>are</u>, instead of how they <u>ought</u> to be. This research paradigm is certainly of concern to biologists, and perhaps even to psychiatrists, but since the modern computer is quite unlike the brain, in either structure or operating principles (such as the brain's are known), there seems to be little direct relation between them.

The rejoinder to this argument is that determining the operation of a network of elements individually as complex as neurons is an extraordinarily difficult procedure. It is

a problem in the analysis of information flow, which the computer scientist is uniquely qualified to study. It is so complicated that pure analysis is largely impossible because of the paucity of accepted first principles, and only simulations with computers (especially large computers) have regularly yielded consistent, interesting results. And neural networks are highly parallel, with many feedbacks; since most formal theory of computation deals with sequential machines, any principles of operation for networks would constitute a contribution to a little-explored but increasingly active field of research in theoretical computer science.

Also, the main work to date has been done by biologists interested primarily in explaining the detailed operation of selected networks (although they of course hope for a grander understanding, they recognize that it is not immediately forthcoming and hence concentrate on whatever they can gather the most data about). There is not even a catalogue of the possible activities of simple networks of commonly-investigated cells. The main impediment here is, once again, the enormous complexity of even the smaller nets.

The AUTONET simulation system is an attempt to make such basic research more approachable. By way of introduction, it is necessary to examine the structure and function of neurons; only then does one understand why so many resources

have already been expended on neural simulation the world over. At one time or another, all of this introductory material has been incorporated into someone's simulator. This zest for all-explanatory power can produce such a plethora of conditions, and such a wealth of data, that many simple patterns were for some time effectively obscured. This neurological data is not presented solely to further an understanding of the AUTONET system (although the terminology is useful); it is presented more so that the reader can better grasp why the high level of analysis represented by the AUTONET approach has been adopted, i.e. so that one can see just how much data there is to obfuscate the research.

After the neurology will come a chapter on AUTONET itself. The primary problem in analysing networks (and thus primary expenditure of energy in designing AUTONET) has been that there is no convenient, widely-accepted language in which to express the network. The WARP network construction language thus constitutes an examination of some of the needs of high-level simulation languages, and suggests some ways of achieving these needs. There are perpetual tensions amongst the competing ideals of completeness, conciseness, clarity, convenience, and computational efficiency. How these are resolved involves techniques drawn from the disciplines of formal languages, compiler design, information theory, and man-machine engineering; there is no

common language shared by all of these, so some terminology is also presented.

The system described, one then moves to a chapter which discusses networks *per se*. This is a brief study of how networks and their component cells might be designed, and of the interactions present in even apparently trivial networks. Following this is a thorough simulation of the network found in the stomatogastric ganglion of the lobster. This network has been much studied elsewhere, but even it yields a few unexpected results.

Chapter IV also displays the value of experimental evidence, gathered via extensive computer simulation, in the analysis of biological systems. In particular, it is possible in a simulation to rapidly modify the network parameters, leading the investigator to a better comprehension of the network's operation.

In such simulations, it is traditional to implement some sort of adaptivity in the network elements. This issue is carefully considered, and the conclusion drawn is that given the enormous complexity of apparently simple systems, a better understanding of the operational principles at work is a prerequisite to proper adaptive algorithms. The emphasis is thus more on the operation of the robust peripheral nervous system than the intricate, adaptive central system.

The thesis concludes with some closing remarks on network design, and what might be expected from future simulations.

Chapter II

INTRODUCTORY NEUROLOGY

The neuron is a eukaryotic cell, found in many organisms, that is characterized chiefly by the fact that its transmembrane potential can change by tens of millivolts in milliseconds due to the presence of pores in the cellular membrane, which permit the passive flow of the ions responsible for the transmembrane potential in the first place. The potential change can be facilitated by another neuron releasing chemicals into a synapse; by a close abutment of neurons electrotonically inducing a potential change; or by the neurons breaching their cell walls to form virtually continuous membrane. A receptor cell, which transduces (and usually amplifies) energy such as light or sound, can also have selective effects. No other natural selective effects are known, although there is considerable research into non-selective effects such as hormones and drugs.

To speak of <u>the</u> neuron is somewhat misleading, for there are at least several dozen types found in primates alone. The number of types in more specific, less plastic nervous systems, e.g. the invertebrates, is truly staggering. Thus, when discussing neurons, unless otherwise specified it may

be assumed that one is dealing with the primate central nervous system (CNS) pyramidal cell, which is the primary constituent of the cerebral cortex.

2.1 MORPHOLOGY.

Neurons[1] vary in the size of their somata from 2 to 1000 microns, pyramidals from 15 to 40 microns. They are notable for having processes, extending from the soma, which are at least several times the somatic diameter (the cell innervating the adult human big toe has a process over one metre long).

The nucleus of a neuron, like that of most secretory cells,[2] is much larger than is typical in multicellular organisms, often consuming half the somatic volume (79% of a linear dimension). As is usual in eukaryotes, the nucleus is enveloped in endoplasmic reticulum, and most of the genome is contained in the nucleolus. However, neurons are unique in that it is exceedingly rare, in higher species, for an adult cell to reproduce; concomitant with this absence of mitosis is a thin and wispy chromatin, as opposed to the robustly-staining variety normally encountered.

---

[1] This and other general neurology may be found in the excellent introduction by [Shepherd,1974]. More detailed information is available in THE NEUROSCIENCES Study Program series by Rockefeller University Press.

[2] Most neurons excrete transmitter substances, in addition to having trans-membrane ion flow. These two activities require an especially active cellular support system.

Also present in the soma are the standard eukaryotic organelles: the rough endoplasmic reticulum, speckled with ribosomes that employ m-RNA in protein synthesis; the Golgi apparatus, which appears to be active in the "packaging" of substances into saccules, e.g. transmitter substances into vesicles and enzymes into lysosomes which lyse cellular wastes; the microtubules and microfibrils which, aside from constituting the structural ground substance, also actively transport material within the cell; a centriole, known to play some part in organising the microtubule network (especially during mitosis); and mitochondria, ubiquitous in eukaryotes as the seat of cellular respiration, and particularly plentiful in neurons, which consume a lot of energy in signalling.

The cellular membrane, 7.5 to 8.0 nanometres thick, is in most parts a phospholipid bilayer largely impervious to both polar and non-polar molecules. In addition to the normal transmembrane transport associated with staying alive, e.g. glucose in, carbon dioxide out, there are channels of .4 to .8 nm I.D. that allow small ionic species such as hydrated Na^+ and K^+ to pass through the membrane when the normally-closed channels open up. The channel density is, per square micron of membrane surface, 100-200 on the soma and its proximal processes, and up to 800 on parts of the axon. The maximal flow rate of the Na^+/K^+ channels seems to be 200-220 Na^+/sec., 120-130 K^+/sec. per channel.

The cell attempts to segregate various ionic species with respect to the cellular membrane. Because of the ionic charges and relative densities, this produces a potential difference between the intra-cellular fluid (ICF) and the inter-stitial fluid (ISF) surrounding the cell. By examining the potentials and respective concentrations experimentally, and comparing this with the Nernst equations for generic transmembrane potential:

$$E = RT/zF * \ln(O/I)$$

where
- E : potential difference between outside and inside
- R : Rankine gas constant
- T : absolute temperature of system
- z : valence of ionic species in question
- F : Faraday constant
- O : outside concentration of ionic species in question
- I : inside concentration of ionic species in question

then one finds that for the mammalian skeletal muscle cell:[3] which has an ICF resting potential of -90 mV, that the potential if only K^+ ions where present and so segregated is -97 mv (both values taken at 37° C.). It is thus considered sufficient to examine only K^+ flux for most cases.[4]

---

[3] Which is quite similar to neurons, but much larger and hence easier to study.

[4] One can readily calculate from the above, assuming perfect membrane capacitance, that hyperpolarization to +40 mV causes a loss of 1 in 10^7 ions if, as in large axons, each cm^2 of membrane encloses 10^5 moles of K^+. The loss goes up to 1 in 3000 for small neurons.

TABLE 1

Ionic Concentrations in micromoles/ml

SPECIES	ICF	ISF
K+	155	4
Na+	12	145
Cl-	4	120
assorted anions	155	34
assorted cations	1	5

As previously mentioned, neurons have processes extending from the soma. When stained, the dendrites exhibit so striking a resemblance to a heavily-branched tree that it is common in the literature to refer to trunks, branches, and arborization patterns. The axon, by comparison, appears to proceed axially from the soma for quite some distance before arborizing (hence the name).

The morphology of the process membrane is virtually identical with that of the soma. The exception is that the axon, if it proceeds more than a millimetre or two from the soma, is <u>myelinated</u> (dendrites rarely, if ever, are). This myelin sheath is composed of the processes of satellite cells (Schwann cells in the PNS, oligodendrocytes in the CNS). There are few, if any, ionic channels under the myelin; they are at the maximum observed density (800/sq. micron) at the nodes. This myelination makes possible a

fast, sure conductance which will be described functionally below. It is unfortunate, especially for physiologists, that only large axons are myelinated, for distinguishing fine unmyelinated axons from fine dendrites in a virtually transparent three-dimensional agglomeration has proven near impossible, and accounts in part for the paucity of data which may be employed in simulations.

The only other morphological peculiarity of note is that as the axon proceeds from the perikaryon (or from a large dendrite) there is often found a cone-shaped region called the axon hillock; the cone seems to be a funnelling of microtubules and neurofilaments into the energy-consuming axon, and may indeed play a largely metabolic support role. Between the hillock and myelin start is the initial segment, characterized by a dense undercoating of the plasma membrane.

There is some controversy surrounding this initial segment. Not only is it definitely not physically present (in an observable form) in the Purkinje cells of the cerebellar cortex, but it is far from clear that it is present in the small, unmyelinated pyramidal cells which abound in the cerebral cortex (although it is found in many large pyramidals). It seems to be becoming more fashionable to speak of the spike initiating zone (SIZ) of the axon. This is in many ways to be preferred, especially when dealing with the motorneurons of some invertebrates, which neurons may have several SIZ's.[5]

The most common junction between neurons, or between a neuron and a transducer cell, is the chemical synapse. These are inherently refractory, i.e. if cell A synapses upon cell B, then electrical activity in cell A can affect activity in cell B, but never the converse (via that same synapse). The six classical types, roughly in order of their popularity in the literature, are: axo-dendritic, axo-somatic, axo-axonal, dendro-dendritic, somato-somatic, and dendro-axonal; the latter are quite rare. Taking the axo-dendritic synapse as an example for which the morphology is particularly well established, one finds enormous deviation within the class. Still, one can look at perhaps the most common pyramidal case.

The axon of the affector neuron comes into close physical proximity to the dendrite of the effector. Supposing that the proximal sections are three-space skew, one most often finds a mushroom-shaped terminal projecting from the axon at nearly a right angle, with its flattened "cap" abutting the "cap" of a similarly-shaped spine of the effector's dendrite. In the terminal, often lined up against the optically-dense pre-synaptic membrane, are the synaptic vesicles: small saccules, 20-40 nm diameter, which hold the transmitter substance. The abutment area may vary several orders of magnitude, but the typical pyramidal-pair synapse is between 0.1 and 10 microns in diameter. The synaptic

---

[5] 'Multiple initial segments' is a rather peculiar nomenclature.

cleft which exhibits somewhat less variation from instance to instance is about 20 nanometers wide for pyramidals. The post-synaptic membrane has receptor sites which react chemically with the transmitter, producing electric potential changes across the surrounding membrane. Because of the shape, size, and lower-level morphology of the spine, it is usually found that no current flows from the dendrite to the spine, but current can flow from the spine to the dendrite: the spine is refractory.

For some time it was commonly believed that a cell uses only one transmitter substance, and that consequently all of its synapses were of the same polarity (excitatory or inhibitory). Although one can still find the occasional reference to this doctrine, the current belief seems to be that the polarity of a synapse is governed by the combination of transmitter substance and receptor binding agent (of which there may be several possible on the cell membrane). For example, the motorneurons innervating the triceps muscle and those from the vagus nerve that affect the heart are both cholinergic in nature, yet the former are definitely excitatory while the latter are definitely inhibitory. Furthermore, in the mollusc Aplysia, there is an interneuron which is excitatory to one affector, and inhibitory to another.[6] It has long since been concluded that any logical combination of polarities is possible in

---

[6] See [Tauc & Gerschenfeld,1961].

neural networks.

The other significant juncture is the electrotonic, or gap, junction. These, too, range from 0.1-10 microns dia., but have a gap of ony 2-4 nm. There may be considerable membrane thickening, increase in channel density, and even a sharing of microtubules present. It is typical for these areas to have a low resistance, to be non-refractory, and for a potential change in one neuron to induce a proportional potential change in the other.[7] Speaking very generally, these electrotonic junctions are faster but less efficacious per unit area than the typical chemical synapse.

One can see, then, that the morphology of a neuron is not all that different from other phylogenetically-advanced eukaryotes. Like other secretory cells, it possesses an enlarged nucleus and Golgi apparatus, and "packages" the emitted chemicals in saccules (much as the liver cell does). Due to the large amount of energy thus and otherwise consumed, it has proportionately more mitochondria (though not as many as muscle); It also maintains a resting potential of about -90 mV by means of the well-known K+/Na+/ATPase pump. Where it differs significantly is in its ability to integrate potential changes in its membrane, as opposed to other cells' dampening of such variations, and to as a consequence variably produce potential changes in

---

[7] Counter-instances to all three of these cases can readily be found in the lobster stomatogastric ganglion. See [Selverston et al, 1976].

other cells (neurons, muscles, etc.). It is this functional aspect of the cell which must now be examined.

2.2 FUNCTION.

For the purpose of simplicity it is traditional to consider the neuron *qua* information processor to exhibit a large degree of I/O segregation, the dendrites/soma performing a non-linear, electrotonic summation of post-synaptic potentials (PSP's), which may be excitatory (EPSP's) or inhibitory (IPSP's) in nature. If this sum exceeds a threshold membrane potential, then action potentials, or *spikes* are produced by the SIZ, whence they travel with little or no attenuation along the axon. The spike *frequency* is a non-decreasing function of the total PSP (if over threshold).

It is instructive to consider an idealized example of, say, one neuron, whose entire membrane is at resting potential, which undergoes a few PSP's along parts of its dendritic arborization; further suppose that this first cell synapses axo-dendritically upon a second, similar cell. The dendrite, being roughly cylindrical and having a capacitative exterior, will act as a cable or waveguide, and indeed appears to obey the laws of such.[8] In particular, it exhibits exponential decay: at any point x, the potential difference is given by

---

[8] [Noble et al, 1974]

$$V = V0 \exp(-x/L)$$

where V0 is the potential difference at x=0, and L is the characteristic length. Since V, at a distance L, will be V0/e, the characteristic length gives a measure of the electrotonic efficiency amongst different neuronal processes (just as decay half-life characterizes radioisotope samples). This analysis is complicated by several physical constraints: the dendrite is of finite length, as opposed to the infinite length in simple cable theory; it has a diameter that varies significantly; and it branches.

The length entails that one must solve the differential cable equations employing boundary conditions, said conditions to be determined physiologically. Following [Rall,1959], one can observe that the two most significant conditions are an open-circuit terminus (as in the confluence of several dendritic branches) in which the terminal potential must be 0 -- rapid decay; and a closed-circuit terminus (as in the dendro-somatic boundary) in which case the terminal potential will be higher than for a segment of unbounded cable -- slow decay. For an intermediate segment of a process which is much longer than it is wide, the infinite-cable analysis appears to be quantitatively adequate.

The second complication is neural process diameter. The electrical resistance of the process interior will decrease with the square of the increased diameter (in accordance

with Kirchoff's law), while the constant-thickness membrane's resistance decreases only linearly. Electrotonic spread, which mentioned above is mediated by ionic gradient and flux, is thus more efficient in the larger processes, where current flow in concentrated in the interior. In fact, because the characteristic length is directly proportional to resistance, the characteristic length varies with the square root of the diameter. To establish the characteristic length quantitatively, it is necessary to measure the membrane resistance and internal resistance independently; while non-trivial, it has proven to at least be possible.

Branching is the last complication to be dealt with here. The general procedure is to form an n-ary brachiation, determine the equivalent varying-diameter cylinder, and thus find the characteristic-length curve. If this curve is not continuous, then the potential gradient will be attenuated as it crosses the branch, and there will be gradient reflections back along the processes (antidromic signalling). It can be shown[9] that, for main branch diameter S, auxiliary branch diameters T(i), that if

S**3/2 = SUM(i=1 to n) T(i)**3/2

then the characteristic length function is continuous. This turns out to be a reasonable first approximation to many dendritic trees in mammalian brains.

---

[9] [Rall, 1959]

As the potential difference travels from the dendrite to the soma, it continues to attenuate in the above manners. The somatic organelles do not significantly alter the signal, so the soma may be dealt with as a short, fat dendrite. Thus the next interesting event is at the axon hillock, where the soma narrows conically to become an axon. Because of its small size and hence greater resistance, the electrotonus does not easily pass down the axon and would soon decay to the level of the background noise.

If the SIZ is not depleted of certain ions, has normal pathology, and has been sufficiently depolarized (to -60 mV in pyramidals), then the region begins acting in a manner quite unlike any other in the cell: it rapidly depolarizes from threshold, commonly going from -60 mV to +40 mV in .25 ms. The cause of this is a positive feedback between the membrane depolarization and the membrane conductance of Na+ ions. This positive feedback is ultimately limited by the electrochemical equilibrium potential of Na+, usually +40 mV (depending on the concentration of other ions). Deactivation of the channels, plus a dramatic increase in K+ flux, returns the membrane potential to -70 mV within 1 ms. This sharp peak is termed the action potential; the 2-3 ms. afterwards, the afterpotential. This latter period is absolutely refractory, in the sense that the ionic balance has so changed that nothing of note can happen electrically until K+ and Na+ are actively pumped across the membrane.

When the balance has been restored, there may be another prepotential, and if the SIZ potential is still sufficiently depolarized another action potential can occur. The potential (voltage) is thus converted into action potential frequency: the information coding domain has been changed.

These action potentials travel down unmyelinated axons by inducing the membrane just ahead of them into a similar explosive depolarization. Since unmyelinated axons are typically small, this entails both slow transmission (as low as 1 m/s in 0.2 micron dia. axons), and a relatively low spike frequency, e.g. after only 5 spikes in 150 ms it may take .5-1 sec. for the axon to recover. This latter aspect is due to the high surface-to-volume ratio for a small cylinder -- there is less space to hold the ATP molecules that drive the pump, which restores the ionic gradients.[10]

Myelinated axons conduct spikes in quite a different manner. Recalling that there are no channels beneath the myelin, the spike must somehow "jump" from node to node in this saltatory conduction. The mechanism for such a flow is that at a node, the normal Na+ occurs as usual, depolarizing the interior. This ionic current then flows exclusively along the interior, driven by both its gradient and the interior's potential difference due to the ions' presence. If the current remains above threshold when it reaches the

---

[10] Such limitations can by no means be dismissed as esoteric, for short unmyelinated axons are taken to comprise a large portion of the cerebral cortex.

next node, another spike is generated, providing a fresh supply of Na+ to continue the process. This internal current flow varies with the size of the axon, the larger and less resistive ones having the greater conduction rates (up to 100 m/s in the giant squid axon). The net effect of myelination, then, is to provide a fast, high frequency, unattenuating spike conductor.

The spike train eventually reaches a synapse. If it is a chemical one, then a rather complex sequence of events is initiated. The action potential causes Ca++ channels to open on or near the presynaptic membrane.[11] This intracellular Ca++, which is usually pumped out, eventually causes the vesicles to burst, spewing their contents into the cleft. The transmitter chemical thus courses through the membrane, across the 300 A. cleft almost immediately, and binds to receptor sites on the postsynaptic membrane. The exothermic binding releases energy, which causes one of several types of possible channel to open, and will thus hyperpolarize the membrane for an IPSP or depolarize it for an EPSP.[12] Due to the slowness of the Ca++ efflux pump, some Ca++ may remain in the terminal when the next spike hits, causing more vesicles to burst than previously (frequency facilitation). Until the system limits are reached, the log

---

[11] There must be a high concentration of extracellular Ca++, or the following chain of events will not be initiated.

[12] A given transmitter-binder pair induces one or the other. There is no evidence, at the time of writing, that a synapse can ever change from inhibitory to excitatory.

of the spine PSP increases linearly with the spike train frequency.

An electrical synapse conducts the spike train, albeit attenuated in amplitude and/or frequency, to the postsynaptic membrane. Unlike the .5 ms timelapse associated with chemical synapses, the electrical coupling yields immediate transmission. The spike train may be carried by the dendrite (which differs little from an unmyelinated axon), or it may be integrated into a PSP, which consequently will have very different rise/fall characteristics (as well as little wee spikes on top).

2.3 THE INADEQUACY OF THE ABOVE DESCRIPTION.

The above is a gross oversimplification of neuronal function, and would never suffice for a serious model of this level of operation. A mere smattering of deviations from this ideal might include:

1. Lack of I/O segregation. In many invertebrates, e.g. the spiny lobster, the soma projects a single tree in which the presynaptic and postsynaptic functions both occur. This can have quite an effect on electrotonic and spike coduction, as well as affecting the Ca++ binding by changing the local potentials and pumping activity.
2. Multiple spike initiating zones. Also found in the lobster, a graded PSP travelling along the process

may suddenly induce spikes. Of course, both the PSP's and spikes travel antidromically as well as orthodromically. The consequent spikes can either replace the graded response that initiated them, or be superimposed linearly or non-linearly, depending upon the ions causing the potentials.

3. Other chemical synapses. An axo-somatic synapse has a more powerful, more immediate effect on the PSP than does its axo-dendritic cousin. An axo-axonal synapse changes membrane activity enormously, and can cause failure of spikes to travel past the synaptic area, or even allow them to pass, but because of the modified base potential the spikes may fail to stimulate vesicle release. The converse, in which vesicle release is enhanced, is also possible.

4. Reciprocal synapsing. It is not uncommon for two dendrites to synapse reciprocally, so that activity in one induces activity in the other, in a tight feedback loop (usually negative, but possibly positive). These can also be serial, e.g. A->B, B->C, B->A, C->A. The dendrites may be responding to graded potentials, spikes, or both.

5. Synaptic adaptation. After a period of firing, one frequently finds facilitation, e.g. the same number of axonal spikes may produce a larger PSP at the end of the period than at the beginning (after accounting

for fatigue). The converse, defacilitation, can also occur. The timecourse for this adaptation can be anywhere from milliseconds to hours, as may be the retention period. What causes adaptation is not well known, and there is yet debate as to its location (presynaptic, postsynaptic, or both). This effect is thought by many to be responsible for memory, but little definitive is to be found.

6. Endogenous bursting. In some cells, e.g. the pyloric dilator of the lobster stomatogastric ganglion or certain motor neurons in the leech heart, the resting potential is not constant but is cyclic,[13] and at times exceeds the threshold (with all that that entails). It thus generates burst of spikes with quiet periods in This natural cycling is thought by many researchers, e.g. [Maynard & Selverston,1976] or [Thompson & Stent,1976], to be responsible for some of the rhythms generated by neural networks.

7. "Spikeless" chemical synaptic activity. Axonal spike trains are not strictly necessary for the release of vesicles; a graded potential will also suffice.[14] This has a considerable effect on the neuron as an information processor, for the graded potential

---

[13] While the underlying mechanisms are still, in many cases, unclear, some excellent work has been done in [Berridge & Rapp,1979].

[14] [Graubard & Calvin,1979]

decays as it proceeds along the axon or dendrite. The synapses will thus no longer receive a uniform signal from the soma.

2.4 MODELS.

Aside from the high-level, rather vague models (or more appropriately metaphors) proposed by some neuropsychologists,[15] there are two common levels at which the neuron per se is modelled, and one at which medium-sized nets of them are modelled. The less prevalent neuronal level may be termed the charge level, for the characteristic of such models is that they examine the actual ions flow through the membrane. A popular method for doing such studies is compartment modelling,[16] in which the neuronal membrane is considered to consist of a set of topologically connected compartments. The activity of a given compartment in the next time interval is determined by the current activity of it and its adjacent compartments. A given potential difference in one compartment will thus decay, and also induce ionic flows in some or all of its neighbours. Thus, using heterogeneous compartments, any membrane activity whatsoever, including saltatory conduction in myelinated fibres, may be modelled. It is also possible, of course, to parameterize fatique and facilitation.

---

[15] E.g. Karl Pribram and his holographic metaphor of memory in [Pribram, 1971].

[16] [Rall, 1964]

The most prevalent modelling level, though, may be termed the signal level. The object of such modelling is to represent the neuron as a graph, and examine the potential change at various nodes of the graph. This most approximates closely the above level of functional description. Without question the greatest advantages to be derived from modelling at this level are that one has ready access to physiological measurements (easily interpreted in the model's terms), and the fact that small networks (which are, after all, the only kind about which any details are known) can be readily and accurately simulated [Hartline,1979].

The other level mentioned is not so much one of neurons, but of networks of little automata of one sort or another; this may be termed the logical level. The first efforts are likely the well-known results of McCulloch and Pitts.[17] These have been followed in more recent years by Uttley's informon,[18] the perceptron popularized by Minsky and Papert,[19] and more recently by Aleksander's RAM nets,[20] which are actually more of an artifice than a model.

---

[17] [McCulloch & Pitts,1943]

[18] [Uttley,1976]

[19] [Minsky & Papert,1969]

[20] [Aleksander,1975]

The advantage of this level is that medium-sized nets (10 to 1000 cells) could be readily simulated, and reconfigured. The disadvantage of this level is that one is farther removed from the neuron, and thus the physiologists' data become harder to interpret in the model's terms, which in turn leads to a lack of confidence in the model, e.g. while a given simulation may produce output similar to physiological readings, there is no longer surety that the similarity is not due to the programmer juggling parameters, as opposed to an actual correspondence between the model and the event.

A shortcoming that is shared by most logical-level systems is homogeneity of logic elements. In all but the very simplest biological instances, the elements are not all of the same type, but vary significantly. This means that nice, theoretical results are much harder to come by; one must either simulate to exhaustion or hope to notice a provable regularity. The problem thus becomes two-fold -- first, the tools (a fast, flexible, convenient simulator) must be built, and then comes the task of learning how to use the tool.

AUTONET is a first approach to the problem of building the tool, and a description of it and the brief experience with its use are described below.

Chapter III

THE AUTONET SIMULATION SYSTEM

Although it has long been known that a Turing Machine with a one-way infinite tape can perform the calculations of one with a finite but unbounded number of two-way infinite tapes,[21] such a machine is not necessarily *convenient* for the purposes at hand. When dealing with a logical-level neural simulation, it is much more convenient to have a three-tape machine: one head for input only, one head for output only, and the third for memory and assorted internal calculations. A simple neural simulator, then, might have N tape symbols for N synaptic inputs, and the binary information of each input is coded as the presence or absence of the respective tape symbol.

A more elaborate model would be to consider a *tree* of inputs to the automaton (hereafter, the *cell*). If the cell can determine the location in the tree of a given input, then the tree can more easily represent the dendritic arborization of a neuron. While the axonal arborization could also be a tree, this is a bit more elaborate than is strictly needed.[22] Thus a *set* of outputs, each being a

---

[21] See [Hopcroft & Ullman, 1969]

[22] It is assumed that low-level effects such as conduction blocking are not required in the model.

unidirectional queue,[23] connected to a distinct leaf of the input tree of a cell, is logically complete, sufficient, and reasonably convenient. In AUTONET, a significant assumption has been made, solely for the purposes of simplification: a cell produces a single output value, which is passed down each output line. This assumption makes it difficult (but not impossible) to implement pre-synaptic adaptation; but on the other hand to require an ordered list of output values adds tremendously to simulation cost (which is a significant factor at some institutions). Since time, distance, and charge are continuous variables at the neurophysiological level, it will be necessary to quantize them for the automaton milieu. One can consider distance to be irrelevant, and examine only the time it takes for the signal to be propagated from one point to another. This has been done, for it also provides a normalization for varying process diameters and other factors which affect the conduction velocity of a signal. The simulation may also be conducted in arbitrary time units, so a line may be measured ultimately by the time it takes for a value to pass from one end to the other.

As was noted above, neurons map from the voltage (spatial) domain to the frequency domain via the spike initiating zone, and back to the spatial domain via the chemical (and to some extent, electrical) synapse. These

---

[23] Antidromic signalling can be accomplished in tidier ways at this simulation level.

seem to occur in the life system for pragmatic rather than logical reasons, so from a logical point of view one of the domains is eliminable.[24] Rather than working with the complete, but somewhat restrictive, binary encoding so often used, the information domain is encoded in small integers (in AUTONET, from -128 to 127).

Since intercellular communication paths vary in effectiveness, some accomodation must be made in the simulator. Considering there to be weights[25] associated with the inputs to provide differentiation of effectiveness, it will be necessary for the cell to have access to this or some other weighting mechanism, in order to more easily model synaptic adaptation. This, in AUTONET, is a function entirely of the cell (hence the third tape head). How the cell weights and sums the inputs is idiosyncratic, and no concern of any higher-level programming.

The last introductory detail to be dealt with is an observation that electrical synapses allow positive and negative values (depolarization and hyperpolarization) to be communicated, whereas chemical synapses react only to positive values. As a consequence, there is a provision in the network specification language for pipes, which pass along any value, as opposed to lines, which pass along only

---

[24] Which domain is used is entirely up to the modeller's imagination, since the simulator just provides him with a bunch of numbers to play with.

[25] Not unlike those for threshold elements, as in [Dertouzos,1965].

positive values. Thus if a cell produces a negative output, this value passes only via pipes; lines pass a zero value. One would most likely represent a non-refractory electrical synapse, then, by a pair of reciprocal pipes.

The principle of AUTONET is thus that a cell has a tree of inputs, and a set of finite- (but possibly differing-) length output queues, which are in turn connected to the input trees of other cells. The respective weighting and summing of the inputs are the properties of the cells alone. There is a strong separation between the topology and geometry of the network, and the functions of the network elements.

3.1 THE SIMULATION ALGORITHM.

Once the appropriate data elements are determined, the simulation is quite straightforward. The requirements for the above-mentioned items are:

CELL :

automaton algorithm
input tree
output line list
distinct memory area for each cell

INPUT TREE NODE :

brother pointer
son pointer

OUTPUT LINE :

indicator of line/pipe status

pointer to object leaf

id of cell that owns object leaf

length of queue

queue values

Given these, the following algorithm will perform the required simulation:

```
At step i,
    For each queue,
       POP the queue
       Add the value & leaf pointer to
          the input list for the object cell
    For each cell,
       Determine the output V from its input list
          using its algorithm
       Push V onto each output pipe of the cell
       PUSH MAX(0,V) onto each output line of the cell.
```

This can be sped up significantly if one examines only those lines which have at least one non-zero value queued, and by passing to each cell calculation only the non-zero input values. The auxiliary memory for the cell calculations can also be managed in an efficient manner.

The simplicity of the above algorithm is somewhat deceptive. A simulator is of little use unless it has input, and it is desirable to have the input language

express the common constructs of the user as clearly, yet briefly, as possible. What is desired here is a language for manipulating groups of neurons, expressing the relations and connectivities between them. The facility for this expressiveness is the WARP network construction language.

3.2 THE WARP LANGUAGE.

In constructing a network of medium or large size, the greatest desire on the part of the investigator is likely to be to avoid duplication of specifications for similar, or identical, elements. This requires a facility for declaring that a group of items is to be handled as a single unit, i.e. the same operation is to be performed upon all the items (in parallel).

3.2.1 Creation of Cell Groups.

There are several facilities in WARP for creating the groups of cells which are to be operated upon later. A net may be specified as an array of cells.[26] A FORMAT gives the number of dimensions, their ranges, and the cell type of an array, but unlike a net does not cause the pertinent data areas for the cells to be created. Thus, one might declare the FORMAT name FLIPFLOP to be a vector pair of NAND cells, and later declare FF1 and FF2 to be FLIPFLOP's by coding:

```
FLIPFLOP = <2> CELL:NAND;
```

[26] There is a restriction to 5 dimensions in the current implementation.

```
FF1 := FLIPFLOP;
FF2 := FLIPFLOP;
```

Both FF1 and FF2 would now be nets of one dimension, size 2, composed entirely of NAND cells; but there would be 4 such cells, two for each net.

A powerful feature of WARP is the LAMINATE command. In this, two nets are concatenated along a given dimension. This is similar to catenation ',' in APL, except that there is no DOMAIN ERROR: if the extents of a dimension do not match, then the smaller of the arrays is filled out in that dimension by adding hyperplanes of null cells until a match occurs. The concatenation is then performed, and a new net is created. This new net is an array of the appropriate dimensions and size, and consists of the identifiers of its cells (as usual). Finally, if the dimensionality of the two operands does not match, the smaller is padded, at the high end, with dimension extents of 1 (not 0) until they do match. (This dimension coercion obviously has no effect on the number of cells in the array, unlike the extent coercion described previously).

E.G. Suppose that one has declared two nets A and B as

```
A := <2,3> CELL:NAND;
B := <2>   CELL:EXOR;
```

then A will be an array like

```
1 2 3
4 5 6
```

and B will be

```
7

8
```

so laminating them along the second dimension

```
C :=  A &2 B;
```

creates C as

```
1 2 3 7
4 5 6 8
```

since it specifies that the concatenation be along dimension 2; thus dimension 2 of the result is the sum of the dimension 1 extents of the operands. Also, B was considered to be an array of size <2,1> rather than merely <2>. Hence, the size of the result will be <2,4>, as it should be.

Supposing that one had coded

```
D :=  A &1 B;
```

the result D would be

```
1 2 3
4 5 6
7 0 0
8 0 0
```

Here, B was coerced into a <2,1> and then into a <2,3>. Only then could lamination take place.

One could also code an expression[27] such as

```
E := A &2 B    &1 A;
```

which would create E as

```
1 2 3 7
4 5 6 8
1 2 3 0
4 5 6 0
```

for the reasons stated above.

In general, the dimension of a single lamination will be the SUM of that of the operands in the lamination dimension, and the MAX of that of the operands in any other dimension (after coercion).

The last group of operations which deal with cellular arrays is REDEFINITION. This consists in giving a new name to a subset of an existing net. If the net OLD already exists, then

```
NEW = OLD;
```

will merely create a synonym for OLD. Alternatively, one could code

```
NEW = <2,3> OLD;
```

---

[27] Evaluated from left to right.

which would create NEW as a 2 by 3 array of null cells, and then copy as many cells from the upper left corner of OLD as would fit.

E.G. if OLD is a 3 by 2 array such as

```
1 2
3 4
5 6
```

then NEW = <2,3> OLD; would set NEW to

```
1 2 0
3 4 0
```

As is usual in AUTCNET, non-existent dimensions are treated as having extent 1. This operation may thus be viewed as truncation in those dimensions where the NEW extent is less than the OLD extent, and as null padding in those extents for which the NEW extent exceeds the OLD.

A close relative of this simple redefinition is RESHAPE. This is like the RESHAPE operation in APL, and causes the row major order vector of NEW to be assigned the row major order vector of OLD. If the total number of elements in NEW is less than that of OLD, there will be truncation; if greater, then there will be null padding.

E.G. with OLD as defined above,

NEW = <2,3> RESHAPE OLD; sets NEW to

```
1 2 3
4 5 6
```

as opposed to the previous result.

The final array subset operation is one in which permutation of the elements takes place. As before, the numbers in the angled brackets determine the size of the NEW array. Immediately after, however, is a list of PERMUTATION function names, enclosed in bars and separated by commas; last comes the net name.

A permutation definition is a fairly ordinary left-to-right, parenthesized arithmetic expression employing the infix operators +, -, *, and /. The only operands permitted, however, are integer constants and two "variables" which may be called ISUB and ISIZE.

The elements of NEW in a permutation redefinition are filled in, one by one, and for each element the permutation functions are evaluated. The results give the index of the OLD element which is to be assigned to the NEW element. At each step, the ISUB vector contains the indices of the current element of NEW; the ISIZE vector contains the vector extents of NEW (and thus is constant during the redefinition). If any calculated index is less than 1, or exceeds the corresponding dimension extent of OLD, then the null cell is assigned to the NEW element. If no permutation function is specified, then the identity permutation is used for that index, i.e. the Nth index of NEW is used as the Nth index of OLD.

E.G. the permutation functions TRANS1, TRANS2, SHIFTL1, and FLIP1 may be defined by:

```
TRANS1  ::  SUB:2;  uses only the ISUB variable 2
TRANS2  ::  SUB:1;  uses only the ISUB variable 1
SHIFTL1 ::  SUB:1-1;  subtracts 1 from the ISUB
                      variable 1
FLIP1   ::  SIZE:1 - SUB:1 + 1;
```

and if OLD was a 3 by 2 of

```
1 2
3 4
5 6
```

then NEW = <2,2> |TRANS1,TRANS2| OLD;

would give NEW the transposed upper square of OLD

```
1 3
2 4
```

because the indices would be calculated as

```
1,1.....1,1
1,2.....2,1
2,1.....1,2
2,2.....2,2
```

and hence the transposition.

```
NEW = <2,2> |SHIFTL1,| OLD;
```

would yield a NEW of

```
0 0
1 2
```

because it shifted OLD along the first index, while

```
NEW = <3,2> |FLIP1,| OLD;
```

flips OLD along its first dimension, giving

```
5 6
3 4
1 2
```

and NEW = <3,2> |,| OLD;

is equivalent to NEW = <3,2> OLD;

so NEW just becomes

```
1 2
3 4
5 6
```

Finally,

```
NEW = <3> |TRANS2,TRANS2| OLD;
```

creates NEW as

```
1 4 0
```

since it extracts the major diagonal of OLD.

It is apparent that this notation is quite flexible, allowing many simple permutations to be expressed with ease. In fact, the notation is <u>complete</u>, in that any permutation whatsoever may be expressed. If a function F(x) <u>expresses</u> a permutation means that evaluating F at x yields the new

position of the x'th object, then for a finite domain one can always find a polynomial in x which evaluates, at the integers, to all of the required values. This follows trivially from the results of constructing a Lagrangian polynomial through all of the required points, and is adequate because evaluation takes place only at the integers in the specified domain.[28] Completeness, of course, is not to be conflated with utility, since in the general case the polynomial passing through N points is N'th order, and will no doubt be long and somewhat opaque to ready understanding. None the less, it seems that this notation is better than some possible alternatives, especially the tabular notation (which requires a change every time N changes, whereas an expression may remain the same, e.g. SHIFTL1). The proposed notation certainly suffices for the simple, common permutations used above, and does not have the inherent nonextensibility of a handful of predefined permutations (such as are supplied in APL).

3.2.2 Connecting Nets Together.

Once a structure has been defined using the above techniques, it is necessary to have a mechanism for interconnecting cells within their respective structures. In AUTONET, this process is called projection of one net onto another. A projection is straightforward if the sizes

---

[28] There is no possibility of interpolation, and its concomitant difficulties.

of the two nets are the same, for there is a one-to-one correspondence between the respective cells. The algorithm is more complicated if convergence (many onto few) or divergence (few onto many) is indicated.

Any such algorithm must meet several criteria:

1. The number of dimensions should be irrelevant.
2. It should be invariant with respect to dimensions, i.e. projection in the i'th dimension must not be in any way dependent upon the extents of other dimensions. This is so that convenient "slices" or subsets of the two nets will have the same projection, regardless of which dimension the slice is taken in.
3. It should be invertible, i.e. for two extent values i and j, the algorithm performed upon i & j A(i,j) should, if graphed, be the mirror image of A(j,i). This ensures that convergence and divergence are duals of each other.
4. All cells of both nets must participate in the mapping.
5. The mapping must be non-trivial, i.e. merely projecting every cell of one net onto every cell of the other is inadequate.
6. The mapping should be "even". By this is meant that if one cell projects upon N others, no other cell should project upon many more, or fewer, than N.

7. It should be "order-preserving", so that the greatest cell in the range of cell i of the mapping be less than the least element of the range of cell j, if i<j.

Now, suppose that the projection involves two unidimensional vectors of length m and n respectively, with m >= n. One may then partition the first vector into n distinct sets, the i'th one being P(i). Consider now the numbers L=FLOOR(m/n) and U=CEILING(m/n). If R = m (mod n), then m = L*(n-R) + U*R, that is, with only two sizes of partitions I and U it is possible to break the first vector up into n pieces. One can thus declare that P(i) is of size L if i<R, and U otherwise. By associating partition P(i) with the i'th element V(i) of the second vector, all of the criteria but the first have clearly been satisfied.

However, it was nowhere stated that the partition elements had to be cells; they could as well be sets of cells. One can then conduct the partitioning along one dimension, and then when projecting each element P(i) onto V(i), one is projecting nets of one lower dimension than before; this may be continued until one of the sets has only one element, in which case that cell is projected onto all the cells of the other set. Since this procedure is independent of the dimension, i.e. for two dimensions i and j the results of partitioning i then j and of partitioning j then i are the same, the first criterion is also satisfied.

E.G. 6 elements A-F onto elements 1-16.

```
FLOOR(16/6) = 2
CEILING(16/6) = 3
16 (mod 6)    = 4
```

A: 1,2
B: 3,4
C: 5,6
D: 8,9,10
E: 11,12,13
F: 14,15,16

E.G. A & B onto 1-5.

A: 1,2
B: 3,4,5

E.G. A-H onto 1-5.

A: 1
B: 2
C: 3
D: 3
E: 4
F: 4
G: 5
H: 5

E.G. the 2 by 8 array A-P onto the 5 by 5 array 1-25.
The first dimension is a 2 onto 5,
the second an 8 onto 5.

First array:

```
A B C D E F G H
I J K L M N O P
```

Second array:

```
 1  2  3  4  5
 6  7  8  9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

Result:

```
A: 1,6
B: 2,7
C: 3,8
D: 3,8
E: 4,9
F: 4,9
G: 5,10
H: 5,10
I: 11,16,21
J: 12,17,22
K: 13,18,23
L: 13,18,23
M: 14,19,24
N: 14,19,24
O: 15,20,25
P: 15,20,25
```

In order to provide a concise, powerful notation for some common operations, it is possible to specify a <u>permutation</u> to be performed along with the projection. Rather than

projecting the original antecedent net, the antecedent cells are considered after permutation (just as in net redefinition) and the projection is from the resultant permuted net onto the object net. This eliminates redefinition of a net where all that is really desired is a little "twist" to the connection pattern.

E.G. the program segment

```
SHIFTR :: SUB:1 + 1;
VEC1 := <6> CELL:OR;
VEC2 := <6> CELL:AND;
VEC1 -> |SHIFTR| VEC2;
```

will yield the assignment

```
1: 8
2: 9
3: 10
4: 11
5: 12
```

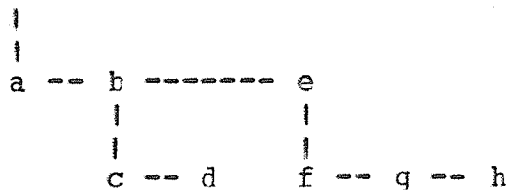
Note that the last cell of VEC1 (6) and the first cell of VEC2 (7) do not participate in the projection. In general, when a null cell is either an antecedent or object no connection whatsoever is established with its counterpart.

If it is desired to have a connection of length other than 1 established, all that need be done is to precede the '->' symbol with the integer length of the line. A pipe, as opposed to a line, is specified by the '*->' symbol.

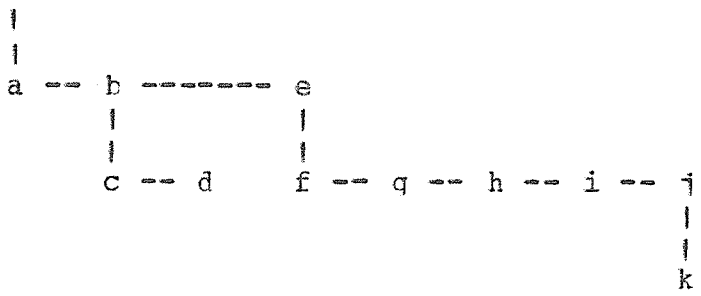
3.2.3 Input Trees.

All that now remains of the network specification problem is to identify the input tree nodes of the object cells. This is accomplished by labelling the tree in a left-to-right, brother-son manner, i.e. each tree node points to its rightmost brother (same level) and its immediate son (next level), either of which pointers may of course be null. The notation 'NET.i.j.k' thus refers to the entity NET, and for each of its cells, beginning at the first level, one counts to the i'th node; proceeding from its son, one counts to the j'th node (the j-1'th brother); proceeding from its son, one counts over to the k'th node; this is the node specified.

If a node specified does not exist then the WARP compiler will create the smallest tree structure needed for that node to exist. For example, if the tree for the single cell in NET was



then projecting onto node NET.3.5.1 would leave the tree as



with 'k' being the specified node. A node is either a LEAF, a FORK, or EMPTY; all nodes created while creating a path to a leaf are initially EMPTY. They become LEAFs by being projected upon, and FORKs by having a non-null son pointer.

As a matter of convenience, it is possible for the level count of a node to be 0. This specifies that one does not care which node at the specified level is to be chosen. Thus if the level count is for other than the leaf (last) level, the first FORK will be taken; If there is no FORK, it will make one out of the first EMPTY node; otherwise, it will add a FORK to the end of the level chain. Similarly, if the level count is for the leaf level, then it will attempt to render an EMPTY node into a LEAF; if there are no EMPTY nodes, it will add a LEAF to the end of the chain. Using the immediately preceding tree for NET, then, projecting onto NET.0 would create a LEAF to the right of 'e'; NET.0.0 would create a leaf below 'a'; and NET.2.0 would 'i' into a LEAF (assuming that 'f', 'q', and 'h' were all LEAFs by previous operations).

3.2.4 Input, Output, and Erasure of Nets.

Although the above describes the basics of network construction in WARP, it is desirable for a network to have inputs and outputs.

An input to a net is a positive integer, which will be queued and eventually be presented at a node of the input tree. Thus one may input the value 2, via a line of length 3, to a net NET by coding

```
2 3-> NET;
```

or one could define a symbol to be an integer, and then project the symbolic value upon the net, as in

```
INITIAL_VALUE_2 = 2;
INITIAL_VALUE_2  5-> NET;
```

When the simulation begins, these will be placed in lines in the ordinary fashion. One may also specify a data file which is to be read, the values of which are passed in row major order to the cells of the net (one value per cell). Permutation of initial values is not supported.

Output may be accomplished by projecting an unpermuted net onto a character string. When a cell with such a projection produces a positive non-zero output, the character string is written to the standard output device. In this way, the user may view the operation of the network without having to trace its entire operation (although the latter course of action is also available).

Finally, there is the matter of erasure of objects within a network. Standard programming languages make no allowance for the removal of objects from the symbol table, except for those defined by scoping and visibility rules. None the less, it must be allowed that a network is unlike a program, and an interactive session in network construction will almost certainly result in constructs which are to be removed.

The problem here is that WARP is a very free-form language, and there is no analogy to strong typing that would limit the extensiveness of the removal of a net from the network. Consider, in particular, a net which is the laminate of two others, both of which are redefinitions of some sort of yet other nets. There is no semantic relationship borne between the original and the redefined nets, and still less between them and their laminates. So for the sequence

```
   .
   .
   .
A := <...> CELL:TYPE1;
B := <...> CELL:TYPE2;
A1 = <...> RESHAPE A;
B1 = <...> |....| B;
C := A1 &... B1;
C -> D;
E -> C;
   .
   .
   .
```

where C and D are elsewhere defined, if one desired to erase C there are several options open:

1. Remove only the name 'C'. This is unacceptable because the operations performed with C are still extant, and there effects will still be present in the network.
2. Remove the name, and the cells contained in C. This entails that any operations done with the names A, B, A1, and B1 become invalid; yet these names precede the definition of C, and in at least some sense have priority over C.
3. Remove the name C, and all of the projections to and from C. This is at least less objectionable than the previous two alternatives, but is impossible in an environment where a network may be stored externally, retrieved, and then operated upon. This would require that all networks be recompiled anew each time, which is not only very costly but is in conflict with the relaxed interactive environment desired in network construction. Also, there is no logical reason to stop with C's connections, since the cells it connects to are connected to others; the ramifications of a single element are usually throughout the entire net.

For these reasons, then, there is no facility for erasure of objects provided in WARP.

3.2.5 Complete WARP Syntax.

The above subsections have described the semantics of the language. It has also included an introduction to the language syntax. Presented here is a more concise description of both, the syntax being expressed in BNF.[29]

```
PROGRAM : SYNTAX - <PROGRAM> ::= <PERMDEF>     ";" <PROGRAM>
                               | <SYMDEF>      ";" <PROGRAM>
                               | <NEWNET>      ";" <PROGRAM>
                               | <PROJECTION>  ";" <PROGRAM>
                               | "END" ";"
```

SEMANTICS - A sequence of statements which are executed in the order of entry, each having been delimited by a semicolon.

```
PERMDEF : SYNTAX - <PERMDEF> ::= <NAME> "::" <EXP>
          where
                   <EXP>    ::= <TERM>
                              | <EXP>  "+" <TERM>
                              | <EXP>  "-" <TERM>

                   <TERM>   ::= <FACTOR>
                              | <TERM> "*" <FACTOR>
                              | <TERM> "/" <FACTOR>

                   <FACTOR> ::= <INTEGER>
                              | "SUB:"  <INTEGER>
                              | "SIZE:" <INTEGER>
                              | "(" <EXP> ")"
```

---

[29] Backus Normal Form. See [Backus,1959]

SEMANTICS - The name is entered in the symbol table as a permutation function definition. Evaluation of the function at x, within the context of an array operation, will yield the new (permuted) position of x. The ISUB and ISIZE values allow references to the complete context of the x'th element during evaluation.

SYMDEF : SYNTAX - <SYMDEF> ::= <NAME> "=" <INTEGER>
| <NAME> "=" <FILE>
| <NAME> "=" <QUOTE>
| <NAME> "=" <NET>

SEMANTICS - The name is entered in the symbol table as an integer, file, or quote string with the appropriate value for the first three alternatives. For the last, it is a net of the status and values of the net specified, i.e. it will be REAL iff the specified net is real.

NEWNET : SYNTAX - <NEWNET> ::= <NAME> ":=" <NET>
| <NAME> ":=" <LAMINATE>

SEMANTICS - The name is entered in the symbol table as a REAL net.

PROJECTION : SYNTAX -

```
<PROJECTION> ::= <INTEGER> <ONTO> <OBJECT>
               | <FILE>    <ONTO> <OBJECT>
               | <NAME>    <ONTO> <FILE>
               | <NAME>    <ONTO> <OBJECT>

where
<ONTO>::=  "->"            |       "*->"
         | <INT> "->"      | <INT> "*->"
         |       "->" <PERMUTE> |       "*->" <PERMUTE>
         | <INT> "->" <PERMUTE> | <INT> "*->" <PERMUTE>

<OBJECT> ::= <QUOTE> | <NAME> | <NAME> "." <TREE>

<TREE>   ::= <INTEGER> | <INTEGER> "." <TREE>
```

SEMANTICS - This creates a connection between the primitive elements of the antecedent (identifier preceding <ONTO>) and those of the consequent (that which follows <ONTO>). All nets referred to must be REAL. If INTEGERs, FILEs, or QUOTEs are projected, explicitly or via a defined symbol, they may not also be permuted (<PERMUTE> must be absent). The INTEGER in <ONTO> is the length of the queue that constitutes each intercellular projection. Employing "*->" in <ONTO> creates a PIPE, which allows negative as well as positive values to be queued; employing "->" creates a LINE, in which all negative values are converted to 0. The TREE of an OBJECT specifies the leaf node of the cellular input trees onto which

the queue projects (where the POPped queue value is placed), the nodes being numbered in a left-right, brother-son manner. If a permutation (see below in <NET>) is specified by employing a <PERMUTE>, the antecedent cells are considered as if they have been permuted by the standard rules.

NET : SYNTAX -

```
<NET> ::= <NAME>
        | <CELL>
        | "<" <ARRAY> ">" <CELL>
        | "<" <ARRAY> ">" <NAME>
        | "<" <ARRAY> ">" "RESHAPE" <NAME>
        | "<" <ARRAY> ">" <PERMUTE> <NAME>
where
  <ARRAY> ::= <INTEGER> | <INTEGER> "," <ARRAY>
```

SEMANTICS - The first case specifies a previously created name, which must be either a FORMAT or a REAL net, in its entirety. The second is equivalent to ' "<1>" <CELL> '. The third is a format for an array, which is of size and shape <ARRAY>, composed of cells all of type <CELL>. The fourth specifies that a slice of the REAL net <NAME>, of size and shape <ARRAY>, is to be taken (in index lexicographical order). The fifth specifies that a new net is to be created of size and

shape <ARRAY>, and that its row major order is to correspond to the row major order of the REAL net <NAME>. The sixth case specifies that the REAL net <NAME> is to be shaped into a new net of size and shape <ARRAY>, and that the indices of each new element are to be calculated using the <PERMUTE> permutation functions, these calculated indices now determining which <NAME> element is to be selected.

<LAMINATE> : SYNTAX - <LAMINATE> ::= <NAME>

```
<LAMINATE> ::= <NAME>
             | <LAMINATE> "&" <NAME>
             | <LAMINATE> "&" <INTEGER> <NAME>
             | <NAME> "&"              "(" <LAMINATE> ")"
             | <NAME> "&" <INTEGER> "(" <LAMINATE> ")"
```

SEMANTICS - The given nets are concatenated along dimension <INTEGER>, or along 1 if <INTEGER> is omitted. Evaluation is from left to right, with parentheses altering the order in the standard manner.

<PERMUTE> : SYNTAX - <PERMUTE> ::= "|" <PERMLIST> "|"
where
<PERMLIST> ::= <PERM> | <PDELIM> <PERM>
<PERM> ::= <NAME>
| <NAME> <PDELIM>
| <NAME> <PDELIM> <PERM>

<PDELIM> ::= "," | "," PDELIM

SEMANTICS - The <NAME> refers to a previously defined permutation function which is to be used in calculating a new index in a net. The commas separating the permutation functions determine which index is operated upon, a function permuting the i'th index if it is preceded by a total of i-1 commas. An index for which no function is specified (either by double comma, or list end) will be permuted by the identity function F(x)=x. All permutation calculations are done in integer arithmetic.

There are, in addition to the above, the following primitive elements of the language:

<NAME> : a string of 1 to 16 alphameric characters, beginning with an alphabetic character.

<INTEGER> : a string of numeric characters.

<CELL> ::= "CELL:" <NAME> | "C:" <NAME>

<FILE> ::= "FILE:" <NAME> | "F:" <NAME>

<QUOTE> : a character string (possibly including blanks) which is enclosed in single quotes. The quotes do not become part of the string.

The implementation of this grammar is quite straightforward, for since it is LL(1) a conventional, top-down parse, e.g. recursive descent, is adequate. The AUTONET implementation places a few restrictions upon the language:

1. The value of an integer may not exceed $2^{31}-1$.
2. A <CELL> or <FILE> name may not exceed 8 characters.
3. A net may have only 5 dimensions. It may not have more than 16384 elements, including null cells. No dimension extent may exceed 1024.
4. No identifier may span the 72 character input line.
5. The remainder of an input line following a double hyphen "--" is treated as comment (as in ADA). Aside from this, blanks may be employed or not, as the user wishes.

3.3 THE SIMULATION ENVIRONMENT.

The WARP compiler is not stand-alone, i.e. it produces no object file which may then be executed. Rather, it runs as a subroutine of the WEFT monitor.

The basic element manipulated in WEFT is the workspace, a section of main memory in which all work is done. There is only one active workspace at any one time; all others are externally stored. The workspace contains the symbol table, all data areas for the network, and is where any cellular context[30] is saved and maintained. Upon start of execution, WEFT allocates a small workspace and initializes the top pointers to the data areas. It also ensures that the files required for system (as opposed to network) operation have been supplied.

AUTONET was intended primarily for interactive use, and hence does not normally echo the user's input. One can compile a program either by entering the statements directly (following the WEFT command DO), or by directing WARP to an external file (WEFT command COMPILE filename). If the results of operation are at any time unsatisfactory, the CLEAR command will re-initialize the workspace. Whenever WEFT is processing commands, it is possible to have the active workspace written out to an external file with the SAVE filename command, or to replace the current active

---

[30] Information needed by the calculating routines which must be maintained between invocations, e.g. the weights applied to different input nodes.

workspace by retrieving a saved one via the FETCH filename command.

An exceedingly important point is that the symbol table is globally attached to the workspace, and is not internal within the compiler. The AUTONET system provides but a single workspace at once, and there is no "linkage editor" facility to resolve the problem of binding together a number of separately compiled networks. A network is a rather free-format object, to which the traditional programming language notions of arguments, parameters, scoping, or even variables are virtually inapplicable. When there is nothing to bind, binding obviously cannot be done.

The most popular way of extricating oneself from this dilemma seems to be a single, massive compilation; by associating a symbol table (in the workspace) with the data, AUTONET more efficiently effects this process.[31] By allowing breaks in the compilation process, however, the user can inspect various parts of the network (and even conduct partial simulation), and then simply continue compiling where he left off.

Following a compilation, then, one might want to know just what one has managed to construct. The SHOW command of WEFT allows the user to inspect the symbol table, the network configuration (including inputs and outputs), or

[31] Staging of programs can be done, but it is still impossible to take two previously-compiled networks and combine them. What must be done is to FETCH one, and compile the other one into the active workspace.

both. The sorted symbol table dump entry gives the symbol identifier, its associated type and status, and for REAL nets the array bounds and numerical identifiers of the cells contained therein (0 for the null cell, as usual). The array contents are dumped in row major order. On the other hand, the network dump is in two parts, the first being a list of the cells in numerical order, and for each cell is listed its symbolic calculation function (cell name) and the list of nodes or quoted strings onto which the cell projects, e.g. a LINE onto cell 2, node 1.1 is listed as (2).1.1, while a PIPE onto cell 3, node 4.2.5 is listed as *(3).4.2.5 . The second part of a dump is a list of initial values (not the symbols defined as initial values), and the nodes onto which these initial values project.

Having gone to the trouble of creating a net, one ought to be able to simulate it. This entails some mechanism for being able to determine when to cease the simulation. There are two ways in AUTONET for a normal cessation of a simulation, which may be termed network quiescence and timing out respectively. In the first case, simulation stops when there are no longer any queues which contain non-zero values, the assumption being that information flow between cells is largely coded by the non-zero values.[32] The second mechanism involves a simulation TIMESTEP. Each workspace contains a TIME value, which is the number of time

[32] It certainly will be, if the user wants to simulate the network with AUTONET.

units that have been simulated within that workspace.[33] If the TIMESTEP value is zero, the simulation ceases only with network quiescence; otherwise, it proceeds until either quiescence is achieved, or TIMESTEP time units have been simulated. The TIMESTEP value can be changed in WEFT with the TIME newtimestep command.

Simulation of a network is initiated by the RUN command. A network is busy if it is not quiescent, i.e. if the preceding simulation was stopped because of a non-zero TIMESTEP. All workspaces are initially not-busy. When such a not-busy network is simulated, the first step is establishment of the initial value conditions. This consists in creating queues of the appropriate length, placing the initial input values in them, and finding and opening the input files[34] required by the network. Once these steps are taken, WEFT ensures that the calculation functions are available. These must be supplied by the user, conform to the IBM PL/1 linkage conventions, and be placed in a library accessible via filename CELLS. Should a required function be unavailable, no simulation will be performed.

---

[33] Irrespective of the number or order of compilations also done.

[34] In AUTONET, there may be only 16 input files open at once. This is due to an inability of the PL/1 Optimizing Compiler to handle an indeterminate number of files.

A simulation may be monitored by the user in several ways. The network may simply be simulated, in which case the only output will be any quoted strings printed as a result of cells firing non-zero values. The *time* may be traced, in which case the simulation TIME value is printed before each simulation step. The *lines* may be traced, in which case the POPped value of each line is printed before any cell functions are evaluated. And lastly the *cells* may be traced, in which case the output value of each cell, if non-zero, is printed as it is evaluated.[35] Any or all of these traces may be applied independently.

The remaining feature available through WEFT is that the user may *reset* the workspace status. A *simple reset* closes all input files, removes all active lines, remove all cellular calculation functions, and set the network to not-busy, i.e. quiesce the network; a *full reset* will, in addition to performing a simple reset, also remove any contextual information from the workspace,[36] and sets the TIME value to 0.

In brief, then, the commands available under the AUTONET monitor WEFT are:

CLEAR : clears the workspace.

---

[35] Since the cellular functions are presumably running in parallel, these and the quoted string outputs will appear in no special order within a simulation step.

[36] Thus establishing a state equivalent to having just done all previous compilations, and nothing else, into a clear workspace.

SAVE filename : saves a copy of the active workspace in the sequential file filename.

FETCH filename : replaces the current active workspace with the workspace in the file filename.

ECHO ON : Causes WEFT and WARP to echo all inputs from the user onto the standard output device.

ECHO OFF : the default, in which no echoing of input occurs.

COMPILE filename : causes WARP to compile the program statements in file filename into the active workspace.

DO : causes WARP to begin compiling the statements the user enters via the standard input device.

SHOW : dumps both the symbol table and network configuration via the standard output device.

SHOW SYM : dumps only the symbol table.

SHOW NET : dumps only the network configuration.

TIME integer : sets the TIMESTEP value to integer, and displays the previous TIMESTEP value.

RUN : simulates the network in the active workspace.

RESET : performs a simple reset of the active workspace.

RESET ALL : performs a full reset of the active workspace.

TRACE TIME : enables the simulation time trace feature.

TRACE LINES : enables the simulation line trace feature.

TRACE CELLS : enables the simulation cell trace feature.

TRACE ALL : enables all simulation trace features.

TRACE OFF : disables all simulation trace features.

This concludes the discussion of the structure and function of the AUTONET simulation system.

Chapter IV

SOME AUTONET RESULTS

Experience with AUTONET falls roughly into the two categories of design criteria for cells and the experimental results of simulating various nets. Following a description of this experience are a few conclusions regarding the possible future of logic-level simulations.

4.1 CELLULAR DESIGN

If one is going to examine properties of networks, then one is going to have to consider the design of the cells that comprise the network. AUTONET was intended especially to handle logic-level neuronal models, so a few comments on the design of such elements would now be in order. The main features to be examined are those applicable to cells in general: specification of input polarity and weights; how the input sum and output values could be calculated; and how adaptivity in cells might be approached.

4.1.1 Distinguishing Input Weights.

It is necessary, in most simulations, to assign a different interpretation to different inputs, e.g. a digital logic latch may have a "data" input and a "control" input.

In general, it is not a good idea to distinguish between two inputs "geometrically", i.e. by giving a fully specified node address; one is much better off distinguishing them "topologically", i.e. by relying more upon the node level than the node address, and specifying the leaf level by coding a 0. This is because of the WARP algorithm for tree creation: in the case of a convergent projection, such as 2 onto 1, WARP will attempt to project two queues onto the same leaf (an error) if the leaf node address is fully specified, whereas it will happily construct the extra node if given the leeway to do so.[37] A topological distinction will thus require less work on the part of the network programmer, and for technical reasons is likely to be preferred.

It can also be argued that there is no firm reason for believing that, in general, a neuron is capable of distinguishing one dendrite from another: the principle of spatial isotropism applies. By far the most important factor in communicative efficacy between two neurons is the number of intercellular synapses and their proximity to each other and to the spike initiating zone; the fact that two synapses are near each other on the same dendritic branch may be relevant, but exactly which dendritic branch is not.

---

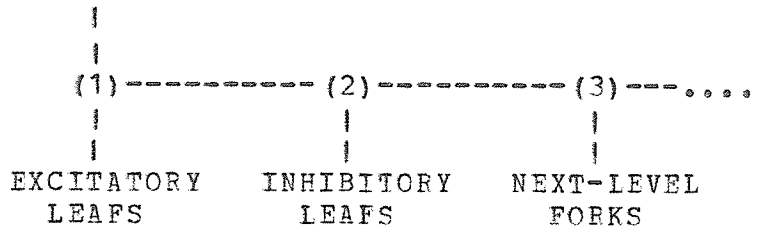
[37] There are cases where this careful checking may be used to advantage. For example, one may wish to have only one control input, but will allow many data inputs. Thus, a mixture of the two specification styles will permit both the cross-checking and flexibility, respectively.

Hence, the best model of the situation would be to leave at least the leaf level of the node address flexible by specifying it as 0.

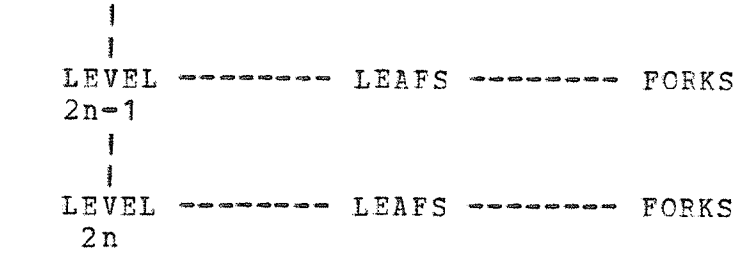
Things become slightly more complex when more than one factor is involved. For the neuron, it will be necessary to have inputs differ in both their relative weights and polarity (sign of the weights). It is possible that in a complex neuron the proximity of synapses also becomes a factor, so many simplistic approaches are out of the question.

Assuming that one is using some sort of monotonically decreasing function of distance for weighting, e.g. exponential decay (characteristic length), the problem becomes one of systematically recognizing the polarity of synapses which are at the same "distance" from the soma and are on the same dendritic branch. Once again, this may be done either geometrically or topologically.

The former course is to have, at a given level, distinguished branches of the tree on which may be LEAFs only. For example, at each level, any node below the first FORK is excitatory; any below the second FORK is inhibitory; and the next level below the third and following nodes must be FORKS only. The tree would then look, at each "distance interval", like



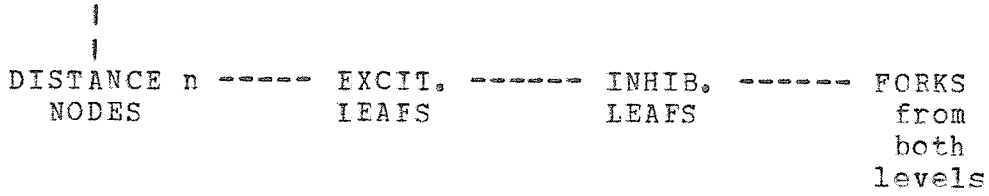
The second course might be to consider a pair of levels to be at the same distance from the soma, with say the lesser level excitatory and the greater inhibitory. This would apply only to same-level leafs, and the remainder of the tree would be merged logically to be at the same effective distance.[38] Thus, a tree of the physical form



would be logically interpreted by the cell as

---

[38] The order in which the merging occurs is rendered irrelevant by an isotropic, topological interpretation of the tree.



As was mentioned above, this topological approach has technical superiority, especially when convergent projections are employed.

4.1.2 Simple Cellular Algorithms.

Once the input protocol to a cell has been established, the next consideration is the function algorithm. At the signal level, the traditional techniques mentioned in Chapter II are used to determine the waveform at a given place and time, with this waveform decaying appropriately as it spreads. At the logic level, one can either do a painstaking modelling or use a grosser, combinational-circuit model to rapidly form the sum; which model is used will depend upon such factors as the time available for the simulation, and the detail or preciseness required. One would take the latter course if, as is the case here, the primary concern is for the general operation of the network (as opposed to an exact and detailed simulation which is, after all, where signal-level simulations excel and thus are at the preferred level).

Such a first approximation to neuronal function might be to linearly sum the inputs, after weighting them for distance and polarity. Because every neuron has an upper limit on spike frequency, one would have to set some maximum for the cellular output value. This approach constitutes a theory of information coding in neural transmission, namely that neurons respond linearly over a wide range of states. As long as the upper (and lower) limits are avoided, the assumption of linearity is actually not a bad model of a neuron with undepleted ionic stores. The greatest difficulty is in determining what output a linear sum of inputs should produce. The spike frequency is limited by the minimum time duration of the refractory afterpotential, and will be 0 for subthreshold activity. Contrarily, subthreshold activity is critical for the correct operation of electrical synapses. To accommodate both systems, the cell output should vary monotonically from the lower limit (maximum hyperpolarization) to the upper limit (maximum spike frequency), with some significant discontinuity to represent initiation of superthreshold activity.[39] Some sort of branch function would satisfy these criteria. For example, if the output upper limit is U, the lower limit L, the threshold T, the resting potential R, and the weighted

---

[39] Experience with AUTONET has shown that even with very flexible parameters this function may vary significantly and not affect network operation; but this shows not so much the irrelevance of the calculation function as the resistance of the network to "faults" and the vagaries of its constituent cells.

sum of the inputs is X, then the calculation function

$$F(X) = \begin{cases} MIN(U, X+R) & \text{if } X>T \\ MAX(L, X+R) & \text{if } X<0 \\ 0 & \text{otherwise} \end{cases}$$

would suffice.[40]

Given the assumptions of reasonable linearity, threshold effects and decaying inputs, this calculation gives results which seem to match well with available data, providing that one is looking not at isolated cells (in which case one shouldn't be modelling at the logic level anyway) but rather at networks of such cells.

4.1.3 Adaptivity and the Cell Function.

The above discussion applies only to the non-adaptive case, and does not account for the previous state of the cell. In fact, the membrane potential from time t-1 may not have decayed to 0, so it will indeed have some effect on the cell's operation at time t. This potential, as is always

---

[40] A more exacting function would be

$$F(X) = \begin{cases} U*X/(X+R) & \text{if } X>T \\ L*X/(X+R) & \text{if } X<0 \\ 0 & \text{otherwise} \end{cases}$$

where R will control the non-linearity of F. This has the advantage of asymptotically approaching U and L, but in the domain of small integers it is not really worth the trouble because of the sparseness of the coding space.

the case for membranes, undergoes exponential decay, so a decay factor D may be associated with the previous summed inputs X'. This would entail the cell's outputting not F(X), but F(X+X'/D). It is possible to decay as many of the earlier states as one wishes, but not only do they become very small very soon, it is difficult to even conceive of a neuronal mechanism that is associated with other than the previous state of the cell.[41] One should note that decaying of previous states is closely related to the varying of the cell's threshold potential.

This type of adaptation is, however, only one of its two common meanings. The other type may be represented as a modification of the weight of one or more of the cell's inputs. While the above adaptation has firm experimental and theoretical bases, the basis for synaptic adaptation is far from clear. It is known that when a pathway between two cells is inactive for some time, it can actually die off; this is very common in the newborn primate, as is the converse growth of dendritic arborization and especially spines (which presumably entails synaptic proliferation). It is also known that a synapse can change its efficacy with use independent of fatigue,[42] but the main factors and even simple morphology have yet to gain widespread acceptance.

---

[41] Until there is firm physiological evidence for a mechanism, it seems that accounting for previous states is rather esoteric.

[42] Distinguished by its relatively short timecourse and predictable pathology.

It can also be argued that implementation af adaptive inputs is somewhat premature. The state of the neural network synthesis art can perhaps best be described as nascent. There are few models of neuronal information processing which are capable of adequately employing the slight differential synaptic efficacy that seems to best describe actual neural adaptation. The role of changing synaptic weights in a complex network of cells is still poorly understood. One is likely much better off exploring the network and behavioral consequences of externally changing the input weights rather than attempting to decide which of many competing dynamic weighting algorithms is in any sense the "best".

4.1.4 Consistency of Cellular Function.

The last factor which is applicable to cells in general is the variability of the cellular calculation function. It is very rare for a cell under study, either *in vivo* or *in vitro*, to display exactly the same behaviour over a one-hour period, even when temperature, nutrient availability, and similar factors are very carefully controlled. AUTONET provides a random variable should the user desire to incorporate this into a cell; it is a normal variate with excellent characteristics, and its global use prevents the interlocking that sometimes occurs when several similar pseudo-random generators are used.

The most common use of randomness is to modify either the resting potential or the threshold of the cell. The former has been intensively investigated (described below), by having the variate account for from 0 to 20 percent of the cell's resting potential (this being physiologically the most justifiable, since it is readily interpreted as random channel openings and other related phenomena).

The option of randomly affecting input weights or the inputs themselves has in essence the same drawbacks as procedurally modifying them, but is even worse due to the unpredictability of the result.

4.2 SIMULATION RESULTS

The results of simulations using AUTONET fall into the two categories of examining simple artificial networks and of simulating a life-system network that has been particularly well mapped. In all of these, the predominant cell is named MOTOR. Like many invertebrate motorneurons, it fires tonically at a low rate until inhibited (resting potential exceeds threshold). Following the arguments presented above, the algorithm is to topologically weight the linearly-summed inputs by an exponential strategy, and output a truncated threshold sum. It also decays the previous state by .5 (D=2), and a normal variate is optionally employed in determining the resting value of the cell.

4.2.1 Simple Networks.

It might be thought that only two or three cells, simply interconnected, would be so trivial a case as to be unworthy of simulation. This view will be shown to be quite incorrect. In fact, simple networks of two or three cells can produce an impressive panoply of activity. There are also many different conditions under which the networks may be operating; the network activity is invariant to some, and highly sensitive to others. It will be seen, in the subsection following, that an understanding of these simple networks is of use in understanding the activity of more complex ones.

The first of these simple networks to be considered consists of 2 cells, both MOTOR. These have no random variates in their calculation functions. They synapse inhibitorily upon one another; the only necessary condition is that any superthreshold response by one cell is sufficient to completely inhibit the other cell. Given the appropriate cell definitions, the WARP program

```
A := C:MOTOR;
B := C:MOTOR;

A 1-> B.1.0;
B 1-> A.1.0;

END;
```

will construct such a network.

When some step is taken to avoid or ignore quiescence, there are a number of patterns that can be generated by this unassuming little network. A simulation of it as specified will yield the infinite pattern

```
A _ A _ . . .
B   B
```

where 'A' indicates that cell A fired during a given simulation timestep, 'B' indicates that cell B fired, and '_' indicates that neither fired during that timestep.

The other patterns can be elicited by synchronizing the network, i.e. guaranteeing that an initial, stable pattern is entered. Letting the name SYNC have a sufficiently large integer value, the addition of exactly one of the statements

```
SYNC -> A.1.0;
SYNC -> B.1.0;
```

will cause the named cell to not fire the first time. This will produce the sequence

```
A   A   . . .
  B   B
```

Adding the statements

```
SYNC  -> A.1.0;
SYNC 2-> A.1.0;
```

to the original program will produce the sequence

$$A \_ \_ \underset{B}{\_} A \_ \_ \underset{B}{\_} \cdots$$

This results because the decayed state value is sufficient to keep a cell inactive for three timesteps.

All of the above patterns are maintained forever, once they are achieved, by "deterministic" or "reliable" cells. Addition of a normal variate to the calculation of the resting value will cause an unsynchronized network to enter one of these stable states. The perhaps surprising result is that it can, after a (usually brief) interval of fluctuation, enter a different stable state. In fact, this flitting from one pattern to the next must occur whenever a random effect is mathematically capable of altering the cell's output function, no matter how minor this randomness may appear. If the random effect is small, then the mean length of a stable pattern will be large, and conversely a large effect will produce ever shorter mean patterns of stability (until the neuron is essentially a stochastic process). But if it is at all possible for a change to occur, then a non-zero probability may be assigned to the event.

Note that this is in apparent conflict with some well-known researchers, e.g. Allen Selverston in [Selverston,1980]. He seems to claim that such reciprocal

inhibition must produce alternating outputs, perhaps comparable to the second and third examples of this subsection. Although this is certainly one of the possible output sequences and intuitively is the simplest, it is hardly the only one. Interestingly enough, any of the above output patterns may be made more probable by slightly changing some of the parameters. For example, the third pattern is immensely stabler than the previous two if either one cell's threshold is slightly lower than the other's, or if one cell inhibits the other noticeably more.[43] The ultimate effect of either case is to have one cell more likely to fire when it ought to, while the other is less likely to fire when it shouldn't; thus, an alternation is strongly favoured.

As Selverston quite properly notes,[44] this alternation cannot occur if only one of the cells is firing, and the cell is not subject to fatique: it would simply fire forever, and the other cell would never fire. Some sort of decaying of cellular efficacy is thus necessary, e.g. gradually decreasing the maximum output value until the cell no longer responds. Because the cell's previous state is available, such a fatiguing is easy to implement. The result, of course, is an increase in the stability of

---

[43] Either of these would explain the experimental results, since it is rather unlikely that two cells would always have exactly the same operating characteristics.

[44] [Selverston, 1980]

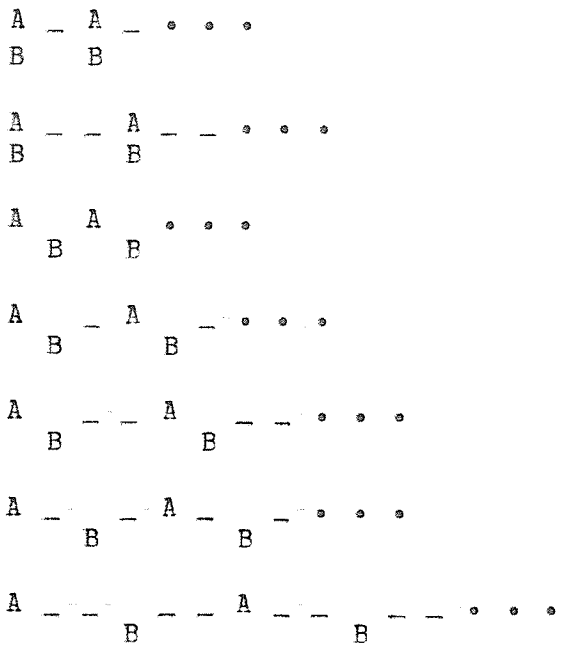
alternate firing sequences.

As also noted by Selverston, the stability of these patterns is enhanced by the introduction into the cellular function of post-inhibitory rebound. This can be implemented, in a logic-level simulation, simply by lowering the threshold of a cell if its previous state was negative (successful inhibition). Such rebounding will greatly increase the likelihood of a recently inhibited cell firing, which is just what is desired in the case of alternately-firing network. While this effect is most dramatic in a complex network, it can be seen in even this simple one: where it may be typical for the pattern to change every 15 to 20 timesteps, the patterns can be stabilized to lengths of 50 or more.[45]

Such is the level of complexity of the output pattern of two cells tightly coupled. The complexity increases for cells more distantly connected, e.g. if the lines are of length 2 instead of length 1, the seven most common and stable patterns are:

---

[45] Exact statistics are pending further analysis of the results.



In terms of information theory, what has occurred is the addition of memory to the system; assuming that each cell had no memory, each line value could either inhibit or not inhibit the cell, so there was 1 bit of information stored in each line. By doubling the line length, the information store in the system was also doubled. For a fixed calculation function, the maximum conceivable number of system states is 2**n, for n bits of information. Thus, doubling the line length increased this maximum from 4 to 16.

Such an analysis is somewhat simplistic. The actual maximum number of achievable states will be considerably less than 2**n, because for a given network configuration some states are unstable. With the cell definitions simulated, one such state[46] is one of the cells firing

constantly; this is highly unstable, and the slightest perturbation will result in the selection of another state. Introduction of cellular memory, by means of previous-state decay, adds more bits of information to the system, and thus more conceivable states.

Refractory chemical connections are not the only ones found in the life systems; electrotonic coupling also abounds. The addition of such a coupling to an inhibitory pair, e.g. with the WARP statements

```
A *-> B.0;
B *-> A.0;
```

results in A being excited when B fires, A being inhibited when B is inhibited (because of the nature of the pipe), and by symmetry B will be likewise affected by A.

Simulation of this shows that if the pipe is of a similar length and efficacy to the line, then when, say, A fires the result upon B is to be excited by A, rather than being inhibited.[47] This tendency becomes even more pronounced when the pipes are of lesser length than the lines but of similar efficacy.

---

[46] There is a duality present because of the network symmetry.

[47] This is also because the cells fire tonically, rather than being normally quiet.

The effect of the pipes is to induce the cells to act together by rapidly propagating the summed signals from one cell to the other. Instead of two distinct cells, there is one more "diffuse" cell possessing two SIZ's. As the coupling constant approaches 1, the cells may be considered to be a single unit. If they are mutually inhibitory, then this single cell is in effect inhibiting itself.

This entails that the two cells strongly tend to fire together, and then be simultaneously inhibited. The electrotonic inhibition, with its faster timecourse, becomes synchronized with the arrival of the slower chemical effect. As a result, the activity pattern

```
A _ A _ . . .
B   B
```

becomes the only one with any significant probability of occurrence, even if the cells' functions or inhibitory effects vary slightly. It is possible, by the introduction of electrotonic coupling, to cause the pattern of simultaneous firing to be preferred, just as other changes (mentioned above) cause alternate firing to be the one that is strongly preferred.

Another example from the literature is the network of three MOTOR cells, connected in a cyclic inhibitory manner. This may be coded in WARP as

```
  A := C:MOTOR;
  B := C:MOTOR;
  C := C:MOTOR;
--
  A-> B.0.0;
  B -> C.0.0;
  C -> A.0.0;

END;
```

As might by now be expected, this produces a wealth of output patterns. The six most common are:

```
A   A
B - B - . . .
C   C


    A     A
  B     B   . . .
C     C


A       A   A       A
B - B -   - B - B -   - . . .
    C   C       C   C


A   A
B - B - . . .


A   A
  -   - . . .
C   C


B - B - . . .
C   C
```

Note the difference between these results and Selverston:

> The recurrent cyclic inhibition model ... can produce as many different phases as there are neurons in the ring.[48]

The proliferation of output sequences due to the increased memory capacity of the network is rapid; for cells with 1 bit of memory, interconnected by length 1 lines, there are 2**6 system states, and allowing different calculation functions there are 2**2**6 or over 10**19 conceivable patterns generable. Of course, this number is reduced considerably by the connectivity, simple cell operation, and dependence of the memory on the cell's previous state; but there are likely hundreds of patterns this network configuration could yield, with only minor changes to the cellular functions.[49]

4.3 SIMULATION OF A LIFE-SYSTEM NETWORK

Simulations of actual neural networks are difficult to achieve because of the problem of gathering sufficient, accurate data. Neurons are small, pack together closely, and interconnect in three dimensions via twisting, branching processes that are often several orders of magnitude longer than the somatic diameter. Even if most of the synapses are found, there is no guarantee that one has accurately established their polarity, efficacy, and timing characteristics. And if the network in question varies from animal to animal, the experiment is plainly unreproducible.

---

[48] [Selverston,1980]

[49] And permitting different cells to have slightly different functions.

It is fortunate for investigators that many invertebrates, especially lower-order ones, have identifiable neurons: cells which vary little in size, location, connectivity and function from individual to individual within a species (and sometimes are constant even across species). This means that the researchers can gather an enormous amount of data by examining many individuals, gleaning a little information from each study until a complete picture has been built.

Allen Selverston and his colleagues have done this with several hundred experiments on the common spiny lobster, Panulirus Interruptus. They used the excellent approach of establishing connectivity functionally by stimulating each neuron in turn, and examining the effects on the others. Given these functional relationships, they were in most cases able to physiologically confirm them by detecting synaptic sites.

The main object of their studies was the stomatogastric ganglion. It is a little bundle of nervous tissue in the lobster's abdomen, containing about 30 cells and projecting many nerves onto muscles. Most of these cells are motorneurons, but there are also several interneurons present. The ganglion's function is to generate the rhythms that result in the stomach "chewing" the food ripped apart by the mandibles.

The stomatogastric ganglion is a particularly important preparation because it has very few afferents. It is also almost unique in that the generated rhythms not only continue when it is deafferented; the outputs remain virtually unchanged. This means that the *in vivo* and *in vitro* results may be compared, and that any conclusions reached in the laboratory are quite likely to be directly applicable to the intact lobster.

The stomach is very much an extension of the lobster's exterior. It is lined with several ossicles or "teeth", which are sharp, hard pieces of chitin to which muscle is attached. Food from the oesophagus sits in the cardiac sac. Constriction of this sac passes the food to the *gastric mill*, which possesses three teeth.[50] The outside pair are the *lateral* teeth, which move in and out. The other tooth is on the top of the mill, and is the *medial* tooth; it moves down and forward, or back and up. The gastric mill crunches up the food by repeating the sequence:

Lateral teeth in.
Medial toothdown (the power stroke).
Lateral teeth out.
Medial tooth up.

---

[50] Conceptually there are only two, since two of the teeth constitute an opposing pair.

These operations overlap very slightly, which may well induce some turbulence that also mixes the food with the gastric juices.

The food then proceeds to the pylorus. The basic operations here are alternate constriction and dilation of a roughly tubular section of tract. This action pumps food into the gut, where final digestion and absorption are conducted.

The identifiable neurons of the ganglion have been assigned mnemonics which are by now in almost universal use. Because of the large amount of muscle to be moved, some which have virtually identical connectivity and muscular innervation may be conceived of as a single logical unit. The one exception to this rule is that PD and AB, which are extremely closely coupled electrotonically, show virtually the same membrane characteristics and thus tend to be treated as a single cell. The names, mnemonics, and known functions in the table are drawn from [Selverston et al,1974].

As mentioned in Chapter II, the stomatogastric cells do not exhibit any significant degree of I/O segregation. The ganglion's physical construction is with all of the cell somata on the outside, each cell sending its single process into the ganglion interior. This interior is the entire neuropil, where all synaptic connections are made. The cell axons then proceed out of the ganglion, and innervate the various muscles via sheathed, myelinated nerves. Because

NAME		KNOWN FUNCTION
Lateral Gastric	(LG)	closes lateral teeth
Median Gastric	(MG)	opens lateral teeth
Lat. Post. Gastric	(LPG)	opens lateral teeth
Gastric Mill	(GM)	pulls medial tooth down
Dorsal Gastric	(DG)	pulls medial tooth up
Anterior Median	(AM)	constricts cardiac sac
Pyloric Dilator	(PD)	dilates pylorus
Anterior Burster	(AB)	??
Lateral Pyloric	(LP)	constricts pylorus
Pyloric	(PY)	constricts pylorus
Ventricular Dilator	(VD)	??
Inferior Cardiac	(IC)	??
Interneuron 1	(INT1)	
Interneuron 2	(INT2)	

the exact sites and relationships of the multiple spike initiating zones are not well known, it shall be assumed for the purposes of simulation that the resultant of the multiple SIZ's is that all synapses are about the same distance from the nearest SIZ. This has nearly the same effect as normal I/O segregation.

4.3.1 The Pyloric Subsystem.

The pyloric subsystem, being a much simpler neural net than the gastric, should be examined first. The network consists of 14 physical, or 5 logical, cells (see diagram IV.1). The PY (8 physical) and LP are reciprocally inhibitory, in the same manner as was described above in the first simple network. The IC and VD neurons also form an inhibitory pair.

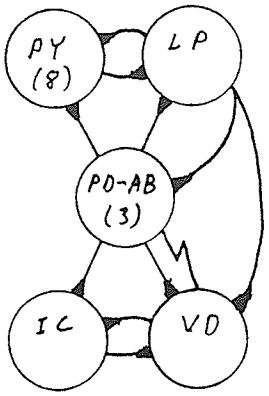


Diagram IV.1 The pyloric subsystem. Closed triangles indicate inhibitory synapses; "lightning" bars indicate electrotonic coupling. Modified from [Selverston, 1976].

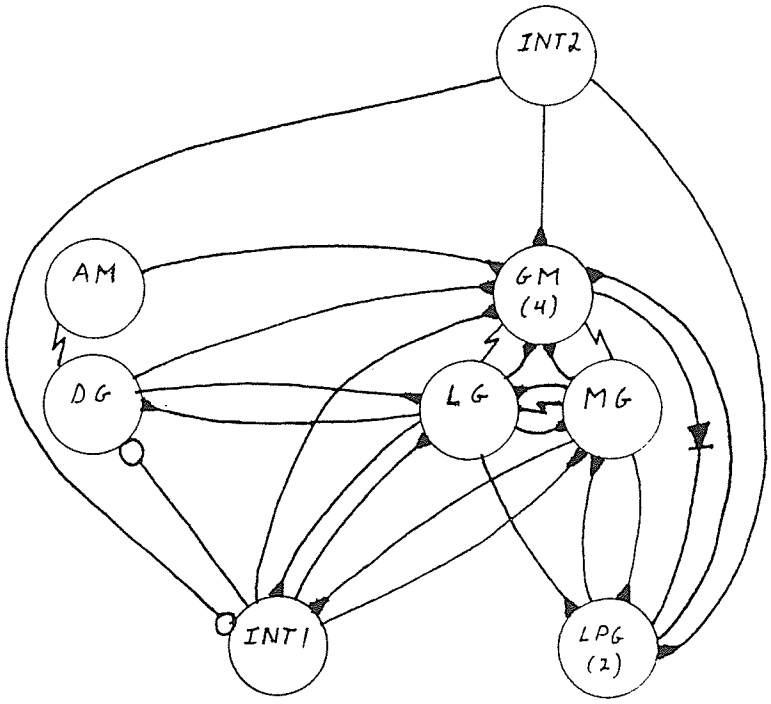


Diagram IV.2 The gastric subsystem. Symbols as above, with open circles indicating excitatory synapses and the diode indicating refractory electrotonic coupling. Modified from [Selverston, 1976].

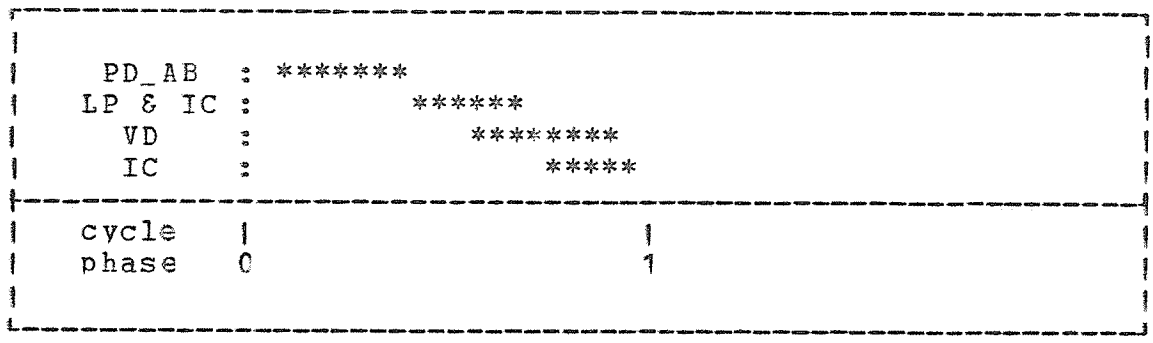
At the heart of this little network is the PD_AB group (2 physical PD, 1 AB). It inhibits all four other neurons, and is electrotcnically ccupled with VD. It is also inhibited by LP, as is VD. PD_AB differs from all others in the stomatoqastric qanqlicn by beinq an endoqenous burster. Its burstinq phase is closely coordinated with the cyclic firinq of the netwcrk, and is qenerally held to be the qenerator of the pyloric rhythm.

The WARP proqram, ignoring the relative synaptic strenqths and line lenqths, is:

```
  PY := C:MOTOR;
  LP := C:MOTOR;
  IC := C:MOTOR;
  VD := C:MOTOR;

  PD_AB := C:BURSTER;
--
  PY -> LP.0.0;
  LP -> PY.0.0;

  IC -> VD.0.0;
  VD -> IC.0.0;

  PD_AB -> PY.0.0;
  PD_AB -> LP.0.0;
  PD_AB -> IC.0.0;
  PD_AB -> VD.0.0;

  PD_AB *->    VD.0;
  VD    *-> PD_AB.0;

  LP -> PD_AB.0.0;
  LP ->    VD.0.0;

END;
```

This is quite simple, as real neural networks qo.

According to a summary of experimental results in [Selverston,1976], the relative sequencing and duration of neuronal firing in the subsystem is given by:



When the cycle phase reaches 1, the pattern will repeat (starting at phase 0, which in this case entails 1 quiet period, followed by the initiation of PD_AB firing, etc.).

There has been extensive AUTONET simulation of this network. Many scores of trials were conducted; variables included the line and pipe lengths, the relative strength of synapses (within a factor of 2), and the degree of randomness with which the resting value was established. The universal constraints for the trials were that the PY-LP pair and IC-VD pair all have the same line length for reciprocal inhibition, and that the electrotonic coupling between PD_AB and VD never exceed their chemical coupling.

The general pattern of output was easy to achieve; the exact pattern was more difficult. This difficulty is attributed to a combination of the abstract level of simulation (there is no

surety that the motorneuron is correctly parameterized) and the simplifying assumptions made -- some of them could be invalid. A typical good match, at a severely quantized level, would be the firing sequence:

```
PD  PD
      LP LP        . . .
           PY PY
```

Eventually, a general tendency in the network was observed. From an essentially random startup, the network often stabilized in the sequence:

1. The PY-LP pair and IC-VD pair would independently cycle.
2. LP inhibition of VD would cause LP and IC to phase-lock together, as did PY and VD.
3. The coupling of VD and PD_AB kept the latter just under threshold.
4. When PD_AB fired, nothing else would fire.

Seeing this pattern led to the question of just what PD_AB was doing in the network, since everything else proceeded just fine when it was removed. In order to better distinguish the network's effect on PD_AB from the endogenous bursting tendency, the statement

```
PD_AB := C:BURSTER;
```

was replaced with

PD_AB := C:MOTOR;

and another series of trials were run.

Despite this modification, the network operation was basically unchanged. The patterns and phase-locking still occurred, but a curious effect was noticed as the randomness of the resting potential was increased: the network would "jump" out of phase, and then take 8 or 10 timesteps to realign itself (2 to 3 full cycle periods). This effect was never observed when PD_AB endogenously burst.

Although the possibility of a programming artifact cannot be discounted completely, there is a rather plausible explanation of all these events. The PD_AB neurons did not seem to generate the rhythm so much as they synchronized it. The network has lots of feedback and is reasonably stable, but it is obviously desirable from a lobster's point of view that the pyloric system not inject food back into the mill or otherwise get itself in a knot. Whenever PD_AB fires, it has a lasting inhibitory effect, and provides an excellent synchronization period. An endogenous burster keeps VD from immediately rebounding, because of the significant resting value drop that is electrotonically transmitted.

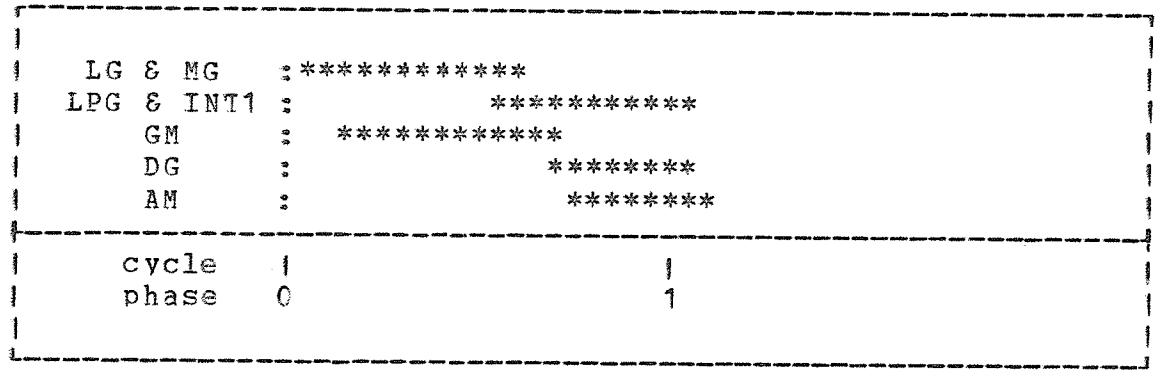
This view of PD_AB as a synchronizer -- as opposed to a generator -- differs somewhat from the conventional view.[51] It is clear that further research is needed. While it might be

[51] Although it has apparently not yet occurred to anyone to not use an endogenously-bursting PD_AB group.

physiologically possible to stabilize the PD_AB group by a cyclic counter-current, it would be easier by far to conduct a very careful signal-level simulation of the pyloric subsytem replacing the bursting PD_AB with a tonically-firing group. A firm conclusion on the matter is impossible pending independent corroboration.

4.3.2 The Gastric Subsystem.

In comparison to the pyloric subsystem, the gastric subsystem is complex. Its 12 physical cells are grouped in 8 logical units with many interconnections, as may be seen in diagram IV.2. There are 4 physical GM cells and 2 LPG cells. The firing relationships, once again from [Selverston,1976], are:



Note that when the cycle phase reaches 1, DG and AM are still firing, so LG and MG will slightly overlap with them.

At the logic level, this cycle is very near to being an alternation of LG, MG, & GM against LPG,INT1, DG, & AM. The last two cells do not have the same firing duration as LPG & INT1; such an event is interpreted, at a high level, as firing for the same duration but with a lower value.

Once again, a large number of simulations bear out the fact that while not exact, the results are consistent with the neurological findings even when many things are varied. After failing to find endogenous bursting in any of the gastric cells, most researchers astutely concluded that the pattern was a network property. Logic-level simulation points to an explanation of this "emergent property".

From the original chaos of startup, the net slowly (20 timesteps) falls into a pattern. One of the common event orders was:

1. LG and MG begin to fire together.
2. GM begins to fire with them.
3. INT1 and DG begin to alternate with LG-MG-GM.
4. LPG alternates with LG-MG-GM.
5. AM fires with DG.

Examining the connectivities reveals why these complex events occur. Note that in addition to reciprocal inhibition, LG and MG are electrotonically coupled. As was described above, this is like one cell that inhibits itself, causing a cyclic firing pattern that is very stable. GM is electrotonically coupled to both LG and MG (though not as

strongly as the two are to each other) so it will begin to adopt the same cyclic variation in total potential that they have. The LG-MG inhibition of GM enhances this effect.

Now, DG synapses reciprocally with LG; LPG, with MG; and INT1, with both LG and MG. Given any slight variations in operation they will, as mentioned above, begin to fire in alternation. AM is very tightly coupled with DG (almost as closely as PD is to AB) so they too will fire together. Extra stabilization is provided by INT1 exciting DG,[52] and AM and DG inhibit GM so that it has synchronizing feedback from this process. Finally, GM has a refractory electrotonic effect on LPG; the function of this connection, however, still remains obscure.

The cells labelled 'E' are not physically part of the stomatogastric ganglion; they actually reside in a commisural ganglion. These tonically excitatory cells <u>are</u>, however, a functional part of the network. Note that they are inhibited (strongly, as it turns out) by INT1. This means that they will provide phasic excitation to the network. It is interesting that the two E cells project only upon those motorneurons which receive heavy inhibition; it may be presumed that the excitation is needed for the cells to overcome deep, depleting hyperpolarization. Regardless, they provide yet another correcting factor should perturbation of the network occur.

---

[52] Which, along with AM, is a normally-quiet motorneuron (and thus is unusual).

The only cell not yet discussed is INT2. This enigmatic interneuron is often left out of both simulations and explanations, both because it is normally quiet and because it receives no intraganglial afferents. It was included in the AUTONET simulations of the gastric subsystem, but due to the paucity of corroborating data, its function must still be considered to be speculative.

If the gastric subsystem is inactive, as when there is nothing to be chewed, some mechanism for initiating the mill must be present. INT2 is the cell receiving most extraganglial afferents, and it is excitatory to INT1 and inhibitory to GM and LPG. A substantial amount of firing by INT2 would initiate INT1 activity, which would in turn stop the LG-MG-GM complex while starting up DG-AM. The combined muscular effect is to open up the mill teeth and squirt in some food.

If INT2 now stops firing, LG, MG, and GM are promptly subject to mild but effectual post-inhibitory rebound. The electrotonic interconnections will nearly ensure a synchronous startup. One this triad has fired, it will begin inhibiting LPG, DG, AM, and INT1. The mill is active.

It is thus hypothesized that the purpose of INT2 is somewhat dual: it not only initiates gastric mill activity, but ensures an initial synchronization of the rhythm. INT2 will act only on the orders from higher-order centres (although it may also be capable of "resetting" network operation). This hypothesis is subject to verification by either a lower-level simulation, or stimulation of a prepared ganglion.

4.3.3 The Combined Network.

Neither the gastric mill nor the pylorus fulfills its gustatory function without the other. There are, as a consequence, some direct interconnections between the two subsystems, as is shown in diagram IV.3. All of these connections have been shown experimentally to be not very strong. This is confirmed by a considerable independence of the respective cycles. Of the five connections, only the electrotonic coupling between LPG and VD has been physically verified; the rest are currently only functionally established.

Repeated simulation has shown that increasing the interconnections only wreaks havoc with one of the two rhythms, as it is forced into a resonance that is not natural. This effect appears, at least in part, to be an artifact of the simulation level, for the resonant frequencies are both discrete and well separated. No significant degree of phase-locking between the two networks was achieved; this is consonant with the majority of the experimental results.[53]

---

[53] See [Selverston et al,1976]

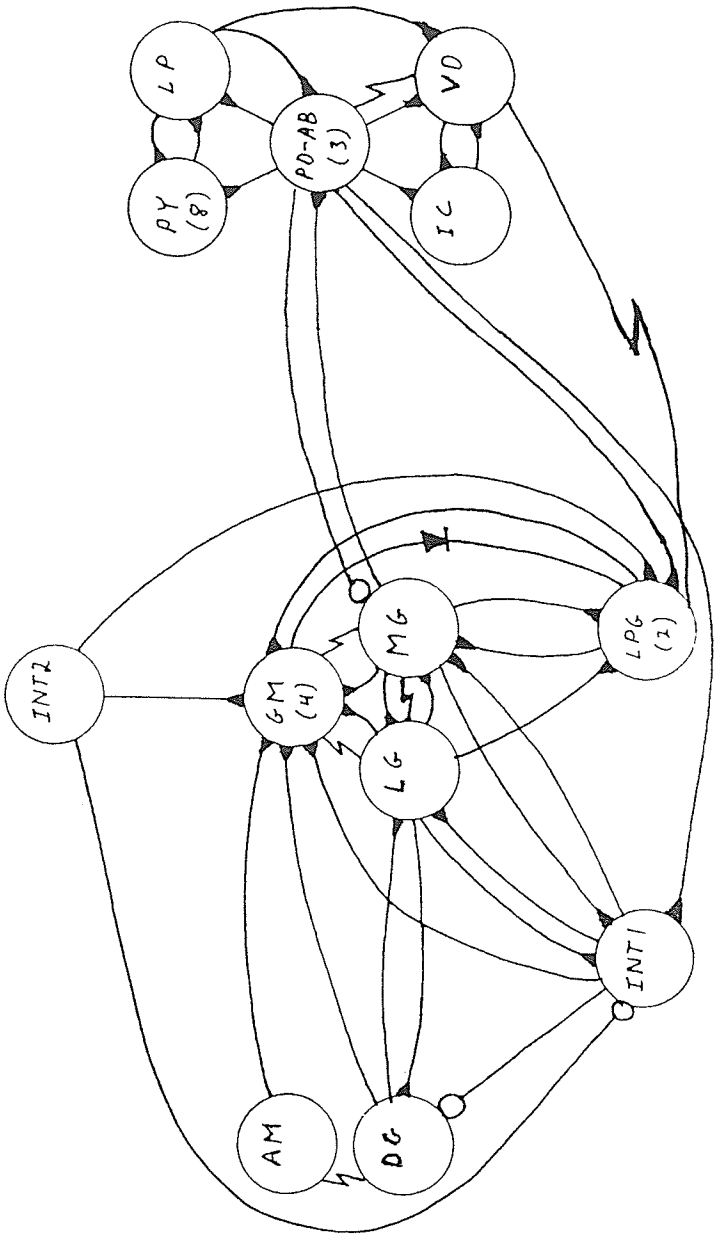


Diagram IV.3 The combined system. Symbols as in Diagrams IV.1 & IV.2 . Modified from [Selverston, 1976].

Chapter V

CONCLUSIONS

AUTONET has proven to be useful in clarifying several theoretical points. While the design of languages is in general a well-established discipline in computer science, little has been done on the specific needs of neural simulation languages. The ability to group cells together into units readily manipulated, and especially clarification of the issues of projection and the types of permutations one may encounter, are among the positive contributions arising from the design of WARP. Determination and distillation of an appropriate simulation algorithm is also expected to be of use in further simulator research.

On the experimental side, the experience with AUTONET has shown the value of logic-level simulation. The greatest advantage it possesses is the investigator being able to rapidly modify the structure of a network or the operation of its constituent elements. The greatest disadvantage is that its abstract level entails that it is sometimes difficult to accurately model a network, and until more is known about the relationship between the logic level and the simulation level any conclusions about a network's operation should be taken as tentative (if the results of a logic-level simulation are uncorroborated).

This hints that perhaps the most efficient use of AUTONET would be in conjunction with a signal-level simulation. It is easy to test networks written in WARP, and thus it has great utility in verifying one's intuition regarding the network. These intuitions are, by and large, sadly lacking; it may be that AUTONET will be valuable in assisting one to recognize the operational criteria of a network, as well as verifying one's suspicions.

AUTONET may also be useful in directing the intensity of research, for it rapidly simulates even relatively complex networks. The main difficulty with a signal-level simulator is that there are a great number of parameters to be identified and quantified. Modifying these without some global heuristic amounts to an exhaustive search of a very large solution space. Experimentation with AUTONET may point to some parameters being crucial, and others being effectively irrelevant to some point in question. This interaction will greatly assist in the multilevel understanding of a network.

An example of this process was given in Chapter IV. Some very simple networks were analysed; one of them, a reciprocally inhibitory pair of motorneurons, was found to generate two types of patterns. The conditions under which these patterns occurred and remained stable were analysed. This highlighted the factors which gave rise to one or the other type, as well as the ones under which they were

unchanged (the transformations to which they were invariant).

Given this, it was possible to analyse a more complex network by breaking it down almost entirely into reciprocal pairs. Knowing the pairwise interactions, the network could be reconceived as a set of modular elements, making the analysis much easier. This led to an explanation of the network's activity.

Such a process is an important part of the growth of any science. There are currently only two levels of explanation for a neural network's activity: either in terms of its constituent parts and their connections, or in terms of its global behavior. What is missing is some intermediate level. But to have such a level of explanation, one must have events that are interacting. Here is where a higher level of simulation plays its most important role: it allows the investigator to abstract from the particulars, and to manipulate the higher-order structures. Indeed, there is no reason (other than ignorance) why the cells of AUTONET could not be non-trivial groups of neurons; the only mitigating factor is that there is not yet any "pool" of such structures from which one can draw.

The future of neural network simulation, then, is promising. It has not proven possible to explain the actions of any but the most trivial creatures by means of "top-down" psychological theories; there may indeed be

insurmountable problems inherent in such an approach.[54] A "bottom-up" stategy, in which the structure of the organism is for once not ignored, may yet provide some answers. If it is to do so, then at least one, and likely several, intermediate levels of explanation of complicated neuronal arrangements must be found. This is to be a long and arduous task, but it is hoped that AUTONET is at least a first step in the process.

---

[54] See [Churchland, 1979]

BIBLIOGRAPHY

Aleksander, I. Action-oriented learning networks. Kybernetes 4:39-44 (1975)

Backus, J.W. The syntax and semantics of the proposed International Algorithmic Language of the Zurich ACM-GAMM conference. Proceedings, International Conference on Information Processing. Paris: UNESCO, pp. 125-132 (1959)

Berridge, M.J. & Rapp, P.E. A comparative study of the function, mechanism and control of cellular oscillators. J. Exp. Biol. 81:217-279 (1979)

Churchland, P.M. Scientific Realism and the Plasticity of Mind. Cambridge: Cambridge University Press. (1979)

Dertouzos, M.L. Threshold Logic: A Synthesis Approach. Boston: MIT Press. (1965)

Graubard, K. & Calvin, W.H. Presynaptic dendrites: implications of spikeless synaptic transmission and dendritic geometry. The Neurosciences Fourth Study Program. (ed. F.O. Schmitt & F.G. Warden) Boston: MIT Press. (1979)

Hartline, D.K. Pattern generation in the lobster (Panulirus) stomatogastric ganglion. II. Pyloric network simulation. Biol. Cyber. 33:223-236 (1979)

Hopcroft, J.E. & Ullman, J.D. Formal Languages and Their Relation to Automata. New York: Addison-Wesley. (1979)

Maynard, D.M. & Selverston, A.I. Organization of the stomatogastric ganglion of the spiny lobster. IV. The pyloric system. J. Comp. Physiol. 100:161-182 (1979)

McCulloch, W.S. & Pitts, W.H. (1943) A logical calculus of the ideas immanent in nervous activity, reprinted in Embodiments of Mind. (1965) Boston: MIT Press.

Minsky, M. & Papert, S. Perceptrons: An Introduction to Computational Geometry, Boston: MIT Press. (1969)

Noble, D., Jack, J.J.B., & Tsien, R. Electric Current Flow in Excitable Cells. Oxford: Clarendon Press. (1974)

Pribram, K.H. Languages of the Brain: Experimental Paradoxes and Principles. Prentice Hall. (1971)

Rall, W. Branching dendritic trees and motorneuron membrane resistivity. Experimental Neurology 1:491-527 (1959)

Rall, W. Theoretical significance of dendritic trees for neuronal input-output relations. In Neural Theory and Modelling. (R.F. Reiss, ed.) Stanford University Press. pp. 73-97 (1964)

Selverston, A.I., Russell, D.F., Miller, J.P. & King, D.G. The stomatogastric nervous system: structure and function of a small neural network. Prog. Neurobiol. 6:1-75 (1976)

Selverston, A.I. Neuronal mechanisms for rhythmic motor generation in a simple system. Neural Control of Locomotion 377-399 (1976)

Selverston, A.I. Are central pattern generators understandable? BBS 3:535-571 (1980)

Shepherd, G.M. The Synaptic Organization of the Brain. Oxford: Oxford University Press. (1974)

Tauc, L. & Gerschenfeld, H.M. Cholinergic transmission mechanisms for both excitation and inhibition in molluscan central synapses. Nature 192:366-367 (1961)

Thompson, W.J. & Stent, G.S. Neuronal control of heartbeat in the medicinal leech. I. Generation of the vascular constriction rhythm by heart motor neurons. J. Comp. Physiol. 111:261-279 (1976)

Uttley, A.M. A two-pathway informon theory of conditioning and adaptive pattern recognition, Brain Res. 102:23-36 (1976)