An Emergent Construction of Adaptive Neural Architectures

Jari Vaario Setsuo Ohsuga
The University of Tokyo
Research Center for Advanced Science and Technology
Komaba 4-6-1, Meguro-ku, 153 Tokyo, Japan
E-mail: jari@ohsuga.rcast.u-tokyo.ac.jp

Abstract

In this paper a modeling method for an *emergent construction* of neural network architectures is proposed. The structure and behavior of neural networks are an emergent of small construction and behavior rules, that in cooperation with extrinsic signals, define the gradual growth of the neural network structure and its adaptation capability (learning behavior). The inspiration for the work is taken directly from biological systems, even though the simulation itself is not an exact biological simulation. The example used to demonstrate the modeling method is also from biological context: the sea hare *Aplysia* (a mollusk).

1 Introduction

In the research work thus far we have introduced an idea of using Lindenmayer systems for connectionist modeling (Vaario, Ohsuga, & Hori, 1991). In this paper we describe how the idea can be used for modeling biological neural networks. As a whole, the research work aims to model the whole life cycle of nervous system as shown in Figure 1.

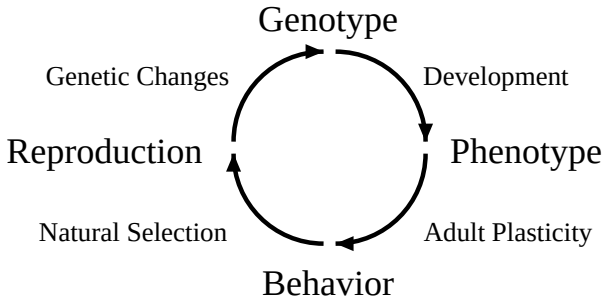

Figure 1: The Life Cycle of Nervous System Illustrating Four Forms of Adaptation: *Development*, *Neural Plasticity*, *Natural Selection*, and *Genetic Changes*. The nervous system is simulated in an environment model that gives the necessary parametric values for each adaptation function.

Modeled neural networks currently ignore the development of the network. Instead, they are designed through a "trial-and-error" method to satisfy the global requirements. However, we propose that the two processes found in biological nervous

systems, the neural network development and the information processing mechanism of the mature networks themselves, should not be separated from each other. This paper introduces a modeling method for both of these two processes. The reasons the development process is important in the neural network models can be summarized as follows.

- *Explicit design of complex systems is difficult.*
  "The complexity of an object can be measured by the length of the shortest algorithm that generates it" (Kolmogorov, 1968). We need an "intrinsic design" method to create complex systems such as neural networks.

- *The development is an adaptation form by itself.*
  The adaptation through the development fills the cap between the "fast", synaptical plasticity based adaptation and the "slow", through genetic algorithms achieved adaptation.

- *Genetic code requires the development process to describe a complex structure.*
  In many models, where genetic algorithms are used, the development process is ignored. However, the development process is necessary to genetatically code a complex structure.

- *Development implements the anatomical plasticity.*
  The long-term memory may be easier to implement through anatomical plasticity than using only synaptical plasticity.

The method uses the idea of emergent behavior, where the complexity is reached without any global definition, but rather by using several local behavior rules that together reveal the global behavior. The neural network is constructed through a set of production rules, and gradually the network grows to its mature stage. In the mature stage, global behavior is reached through several simple local behavior rules, represented by means of production rules, which change the stage of the network model. The created system resembles many knowledge base systems, and in fact is implemented through a multi-layer logic system (KAUS, discussed later) developed for knowledge base applications.

Development itself has been already modeled a long time ago. The great success in modeling plants development through Lindenmayer systems (L-systems) (Lindenmayer & Prusinkiewicz, 1989; Prusinkiewicz & Lindenmayer, 1990), and a general genotype interpretation for L-systems by Gille *et al.* (Wegrzyn, Gille, & Vidal, 1989; Gille, Wegrzyn, & Vidal, 1989) have strongly encourage us to find a similar method for neural systems. However, some extensions for the "standard" L-systems have been necessary in order to capture the convergence phenoma of nervous systems (Vaario et al., 1991).

In this paper we will concentrate on describing the development process of neural networks, especially the intrinsic part of it. Only limited behavior patterns related to the most basic learning phenoma of the mature network are covered; discussion of higher order behavior will be left for future treatment.

2 Related works

In our approach the neural networks are automatically constructed based on a simple set of emergent construction rules. Szu has introduced in his work a similar automatic construction of the growing axons *i.e. hairy neurons* (Szu, 1989, 1990). His model is based on the energy convergence theory. His research work presents a mathematical theory for growth, while our approach is to create a symbolical presentation for growth.

Emergent behavior was widely used by Brooks in his layered control systems for robots (Brooks, 1991a, 1991b, 1990; Brooks & Viola, 1990). However, the model lacks the capability for automatic construction *i.e.* , development. Emergent behavior is also a widely used approach in *Artificial Life* (Langton, 1989; Langton, Taylor, Farmer, & Rasmussen, 1992; Meyer & Wilson, 1991).

The *computational neuroethology* (Gliff, 1991) is an approach where the network is embedded within a closed-environment simulator. An example of this approach is Beer's research which modeled a complete artificial insect (Beer, 1990; Beer, Chiel, & Sterling, 1990; Beer & Chiel, 1990, 1991). However, it still lacks plasticity as well as the model of development process.

Anatomical plasticity in neural networks has been proposed in context of incremental learning (Alpaydin, 1991). In our approach we try to model also the growth of the basic system, not only the modification during the learning period.

Genetic algorithms applied to neural networks have been investigated a lot (Jefferson et.al., 1992; Belew, McInerney, & Schraudolph, 1992; Harp & Samad, 1991; Harp, Samad, & Guha, 1990; Jefferson et.al., 1990; Chalmers, 1990). They differ from our approach because the development process is not modeled. Thus, created network architectures lack one free dimension during the transcription from genetic code to the network structure.

The complete *biological models* of nervous systems (Edelman, 1987; Burnod, 1990) does not consider the computability, a necessary element from the point of view of simulation. We have stressed the capability of structure and behavior to emerge from small local rules that can be easily parameterized, modified, extended, or deleted.

3 Modeling techniques

The neural network modeling method requires basic network topologies: *divergence*, *convergence*, *recurrent networks*, and *axon-to-axon connectivity*. The capability to model cell divisions, connectivity formation, and various cell-cell interactions in each network topology are also required.

The necessary production rules for each network topology follow. Production rules are based on the L-systems, but the "standard" L-systems have had to be extended to describe also the convergence and recurrent network structures.

3.1 Lindenmayer systems

The basic elements of the L-system are an *initial word* and *production rules*. The word represents the model, and the production rules represent the modification rules

for the model. The word is a string of letters (symbols). In each *derivation step* all letters of the word are changed (rewritten) according to the production rules.

Production rules have the general form:

$$predecessor \rightarrow successor$$

with the interpretation: *the predecessor will be replaced by the successor in the given symbol string.*

Parametric L-systems have, in addition, parameters associated with the symbols and a condition to indicate if the production rule will be applied or not. The symbols : and $\rightarrow$ are used to separate the three components of a production rule: the *predecessor*, the *condition* and the *successor*. The general form of a production rule is

$$predecessor(parameters) : cond. \rightarrow successor(expression)$$

Context sensitive L-systems allow production rule selection to be based on the context of a predecessor. They are necessary to model information exchange between neighboring symbols.

Each predecessor can have *left* and/or *right context*. The general form of context-sensitive production rule is

$$left \prec predecessor \succ right \rightarrow successor,$$

where $left$ represents the left side of the predecessor in the symbol string and $right$ represents the right side. The rule reads: *the predecessor will be replaced by successor, if and only if left and right contexts are matched.*

As an extension for the L-systems a *multiple value context matching* operator has been introduced (Vaario et al., 1991). The general form is as follows.

$$left \prec (*)\ predecessor\ (*) \succ right \rightarrow successor,$$

where all possible matched left and right-context symbols are collected and their parameters are used to calculate the parameter of successor. This extension has been necessary for the convergence signal propagation, as we will see later.

3.2 Used Symbols

Used symbols and their parameters can be selected arbitrary in L-systems. Due the convergence and recurrent network structure, the "standard" L-systems' structural elements were not used. Instead an indexing method was selected. This has, however, a biological context as will be shown below.

In this paper only the following symbols are used.

$\mathcal{N}$ Neuron cell.

$\mathcal{A}$ Axon of the neuron at the left side.

$\mathcal{D}$ Dendrite of the neuron at the right side.

$\mathcal{F}$ Facilator axon to form "axon-to-axon" connections.

Each symbol may have an arbitrary number of parameters that describe the internal state of neuron, or its connections. There is no need to fix the number or type of parameters; instead, they can be created and modified by the production rules during the development process.

3.3 Network Topology

The most general neural network topology consists of divergence (from sensory neurons) and convergence (to motor neurons) networks. Between sensory and motor neurons are interneurons, that not only have forward connections, but may also have backward connections (feedback loop structure). These interneuron connections form the recurrent topology for networks.

The connections between neurons are normally formed between axon and dendrite. Axon to axon connections are also created to act as an inhibitory connection. We have included these as part of the network structure, instead of describing them as a negative weight *etc.* as is often done in mathematical neural models.

3.3.1 Divergence Network

A divergence network is a network where one neuron has multiple connections to other neurons. A symbolic representation of a divergence network may appear as in Figure 2.

$$\mathcal{N}_A(\ldots)\,\mathcal{A}(ind_1,\ldots)\ldots\mathcal{D}(ind_1,\ldots)\,\mathcal{N}_B(\ldots)\ldots\mathcal{D}(ind_1,\ldots)\,\mathcal{N}_C(\ldots)$$

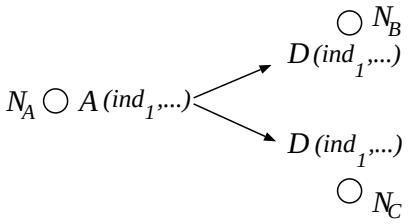

Figure 2: The Divergence Network Topology.

Axon ($\mathcal{A}(ind_1,\ldots)$) is connected to the dendrites with the same index ($\mathcal{D}(ind_1,\ldots)$).

Signal propagation from axon to dendrite through a divergence network can be implemented by the following type of production rule.

$$\mathcal{A}(ind, signal,\ldots) \prec \mathcal{D}(ind,\ldots) \rightarrow \mathcal{D}(ind, f(signal),\ldots)$$

Axon signal ($\mathcal{A}(ind_1, signal,\ldots)$) is received by the dendrites having the same index ($\mathcal{D}(ind_1,\ldots)$) and the signal effects the dendrite according to function $f(signal)$.

3.3.2 Convergence network

A convergence network is a network where there are multiple connections from other neurons to one neuron. The symbolic presentation of convergence network may appear as in Figure 3.

$$\mathcal{N}_A(\ldots)\,\mathcal{A}(ind_1,\ldots)\ldots\mathcal{N}_B(\ldots)\,\mathcal{A}(ind_1,\ldots)\ldots\mathcal{D}(ind_1,\ldots)\,\mathcal{N}_C(\ldots)$$

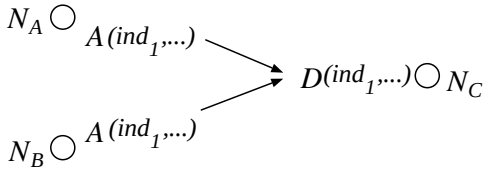

Figure 3: The Convergence Network Topology.

Dendrite ($\mathcal{D}(ind_1,\ldots)$) receives connections from the axons having the same index ($\mathcal{A}(ind_1,\ldots)$). The signal propagation from axon to dendrite through a convergence network can be implemented by the following type of production rule.

$$\mathcal{A}(ind, signal,\ldots)\ \prec(*)\ \mathcal{D}(ind,\ldots)\ \rightarrow\ \mathcal{D}(ind, f(\textstyle\sum signal),\ldots)$$

The only difference from the divergence type of network is the parameter for the predecessor operator ($\prec(*)$). This means that all matched left-context elements are collected and their parameters are usable for successor parameter calculation.

3.3.3 Recurrent network

In the recurrent network, the connections form a loop structure. The symbolical presentation of a recurrent network may appear as in Figure 4.

$$\begin{aligned}&\mathcal{N}_A(\ldots)\,\mathcal{A}(ind_1,\ldots)\ldots\\ &\qquad\mathcal{D}(ind_1,\ldots)\mathcal{D}(ind_3,\ldots)\,\mathcal{N}_B(\ldots)\,\mathcal{A}(ind_2,\ldots)\ldots\\ &\qquad\qquad\mathcal{D}(ind_2,\ldots)\,\mathcal{N}_C(\ldots)\,\mathcal{A}(ind_3,\ldots)\mathcal{A}(ind_4,\ldots)\end{aligned}$$

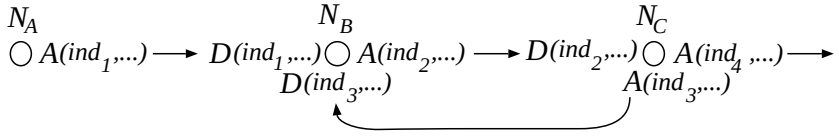

Figure 4: The recurrent network topology.

Signal propagation from axon to dendrite direction in a recurrent network topology is implemented by the following type of production rule.

$$\begin{aligned}&\mathcal{A}(ind, signal_1,\ldots)\ \prec(*)\ \mathcal{D}(ind,\ldots)\ (*)\succ\ \mathcal{A}(ind, signal_2,\ldots)\\ &\qquad\qquad\rightarrow\ \mathcal{D}(ind, f(\textstyle\sum signal_1 + \sum signal_2),\ldots)\end{aligned}$$

The dendrite, through the multiple context matching operation, must collect signals from left and right hand of the symbolical representation.

3.3.4 Axon to Axon Connectivity

In biological systems there are often connections that formed not only between axon and dendrite, but also between two axons. For these cases, for clarity, we have defined a new axon type, called *facilitator axon* and marked by $\mathcal{F}$. A symbolical presentation of a network using axon to axon connection may appear as in Figure 5.

$$\mathcal{N}_A(\ldots)\,\mathcal{A}(ind_1,\ldots)\,\mathcal{A}(ind_2,\ldots)\;\ldots$$
$$\mathcal{D}(ind_1,\ldots)\,\mathcal{N}_B(\ldots)\,\mathcal{F}(ind_1,\ldots)\,\mathcal{F}(ind_2,\ldots)\;\ldots$$
$$\mathcal{D}(ind_2,\ldots)\,\mathcal{N}_C(\ldots)\,\mathcal{A}(ind_3,\ldots)$$

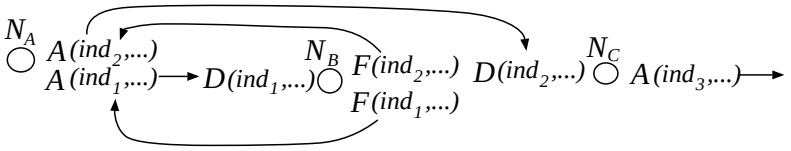

Figure 5: The axon-to-axon network topology.

Axons ($\mathcal{A}(ind_1,\ldots)\mathcal{A}(ind_2,\ldots)$) are receiving facilitator axon connection from the neuron N_B. It behaves exactly same way as the ordinary axon, but in the signal propagation rules the dendrite is replaced by the facilator axon ($\mathcal{F}$).

$$\mathcal{A}(ind,\ldots) \succ \mathcal{F}(ind, signal,\ldots) \rightarrow \mathcal{A}(ind, f(signal),\ldots)$$

The facilator axon signal is received by the axon ($\mathcal{A}(ind,\ldots)$). This rule will be applied for two axons in the above model above ($\mathcal{A}(ind_1,\ldots)\mathcal{A}(ind_2,\ldots)$).

4 Development Model

The development phase consists of a *genesis of cells*, which forms the basic neural structure; *establishing connections*, which forms the connections between neurons; *modifying connections*, which eliminates unnecessary connections; and *adult plasticity*, which reconfigurates connections, modifies existing connections and changes state of neurons and connections (Brown, Hopkins, & Keynes, 1991; Kandel, Schwartz, & Jessell, 1991; Jacobson, 1991).

To bring about L-systems' execution, one applies production rules to the current state (at beginning to the initial state). The current state represents the neural network model. During every evaluation cycle the current state will be replaced by a new one generated by the production rules. The evaluation cycle represents the increment of time. The production rules add or delete symbols, or change the parameters of symbols, *i.e.* , they represent the cell activities: division, growth, death, and interaction between other cells. The condition statement in the production rules determines if the cell activity should take place or not.

4.1 Genesis of cells

Genesis of cells creates the basic neural structure through multiple cell divisions and differentiation of the cells. It consists of two types of production rules: *Cell division* and *Cell's fate determination*.

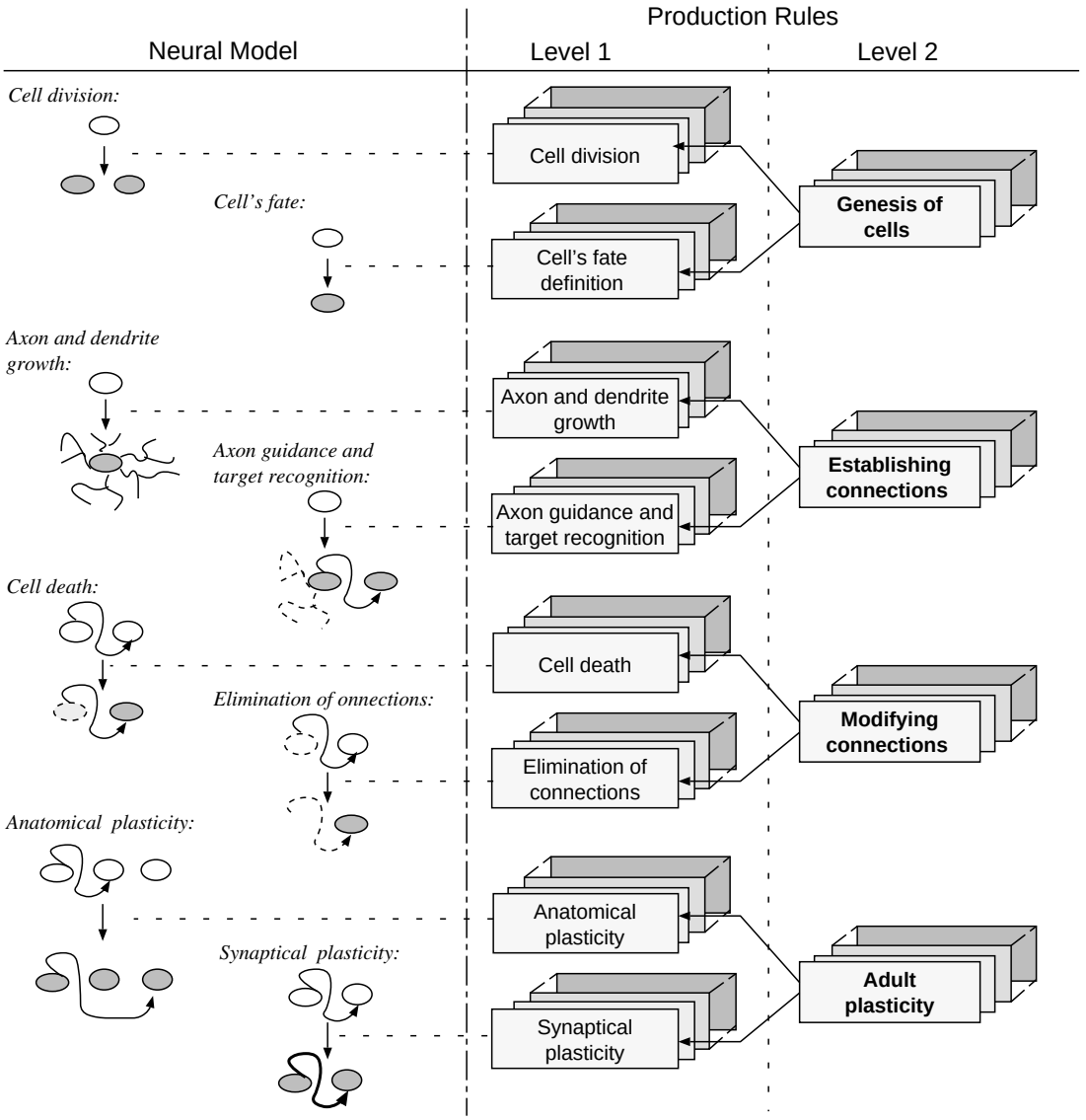

Figure 6: The Development Model of the Nervous System Consists of Production Rules that Control the Gradual Growth of the Neural Network. The upper level (meta-level) production rules may be used to control the lower level production rules. The concept is the same as used in computer aided design (CAD) to describe complex dependencies of objects (Ohsuga, 1989, 1990).

Cell division

There are basically two mechanisms for specification of phenotype: *lineage (intrinsic)* and *inductive (extrinsic)* specification (Brown et al., 1991, page 15).

The lineage specification is an asymmetric cell division producing two different types of cells. This is illustrated by the following type L-system productions.

Asymmetric division:
$\mathcal{N}(type_{12},\ldots) \rightarrow \mathcal{N}(type_1,\ldots)\mathcal{N}(type_2,\ldots)$
Symmetric division:
$\mathcal{N}(type_1,\ldots) \rightarrow \mathcal{N}(type_1,\ldots)\mathcal{N}(type_1,\ldots)\ldots$
$\mathcal{N}(type_2,\ldots) \rightarrow \mathcal{N}(type_2,\ldots)\mathcal{N}(type_2,\ldots)\ldots$

Inductive specification is determined by external matters such as position-dependent cell-cell interactions. This illustrated in the following L-system productions, where *cond1* results in context $\mathcal{N}_A\mathcal{N}(type_1,\ldots)\mathcal{N}_B$ *proliferative* division and *cond2* results in context $\mathcal{N}_{A'}\mathcal{N}(type_1,\ldots)\mathcal{N}_{B'}$ *stem cell* division and in context $\mathcal{N}_{A''}\mathcal{N}(type_1,\ldots)\mathcal{N}_{B''}$ *diversification* division.

Proliferative:
$\mathcal{N}_A \prec \mathcal{N}(type_1,\ldots) \succ \mathcal{N}_B : cond1 \rightarrow \mathcal{N}(type_1,\ldots)\mathcal{N}(type_1,\ldots)$
Stem cell:
$\mathcal{N}_{A'} \prec \mathcal{N}(type_1,\ldots) \succ \mathcal{N}_{B'} : cond2 \rightarrow \mathcal{N}(type_1,\ldots)\mathcal{N}(type_2,\ldots)$
Diversification:
$\mathcal{N}_{A''} \prec \mathcal{N}(type_1,\ldots) \succ \mathcal{N}_{B''} : cond3 \rightarrow \mathcal{N}(type_2,\ldots)\mathcal{N}(type_3,\ldots)$

After birth, most nerve will migrate to the their final position through earlier born cells. At the time being, we have simplified the model not to include neural positions, but to have a *position-free* structure. However, the modeling technique does not preclude having a direction definition in the production rules, as in the case of plant modeling through L-systems.

Cell's fate determination

In the production rules above, the cell types were determined through the division rules. The final fate of cells can be determined by a separate production rules.

Fate of Cell:
$\mathcal{N}_A \prec \mathcal{N}(type_1,\ldots) \succ \mathcal{N}_B : cond \rightarrow \mathcal{N}(type_2,\ldots)$

This production rule states that the final fate of a cell is dependent on the surrounding cells ($\mathcal{N}_A$ and $\mathcal{N}_B$) and a condition statement (*cond*).

4.2 Establishing connections

Establishing connections can be divided into: *Axon and dendrite growth*; *Axon guidance and target recognition*; and *Synapse formation*

Axon and dendrite growth

Axons and dendrites are parts of nerve cells. However, we have modeled them using separate symbols: $\mathcal{A}$for axons and $\mathcal{D}$for dendrites. In biological systems there is only one axon per neuron, but multiple terminators per axon. This has been simplified by combining these two in a one symbol. Furthermore, the axon tree, that is in basically able to create trough L-systems, has been omitted.

Growth is simulated with the following type of productions.

Axon growth:
$\mathcal{N}(label, llist, \ldots) : cond. \rightarrow \mathcal{N}(type, cdr(llist), \ldots)\mathcal{A}(car(llist), \ldots)$
Dendrite growth:
$\mathcal{N}(label, \ldots) \quad : cond. \rightarrow \mathcal{D}(label, \ldots)\mathcal{N}(label, \ldots)$

The *llist* is used to control the connectivity of network. Usage is explained in detail below.

Axon guidance and target recognition

In biological systems there are mainly two factors to control axons in finding their target: *pathway guidance* and *target recognition.* The pathway provides general guidance toward the target area. In the target area, the axon recognizes and establishes the right connection.

Many factors control the final target selection. We have selected the chemical attractors as the basis for our model. This selection does not, by any means, restrict modeling based on other factors as well. Chemical attractors are called labels in the model.

The above process has been notable simplified. Although we have a position-free structure, we cannot model pathway guidance. Axon growth has been simplified to avoid an axon tree growth model; the target recognition problem has been reduced to selecting correct labels for the axons and dendrites to facilitating formation pairs.

The labels are *inherited* through the cell lineage. Each neuron has a list of labels indicating its potential connections. The neuron grows axons, as described above, with a target cell label. Similarly, the dendrite will have a neuron label to "attract" the right axons to form connections. An axon-dendrite pair is recognized through the following production rule.

Axon-Dendrite pair:
$\mathcal{A}(label_1, \ldots) \prec \mathcal{D}(label_1, \ldots) : cond. \rightarrow \mathcal{D}(label_1, \ldots)$

A position specific target recognition can be implemented by extending the above production rule with context sensitive matching. In that case, the axon-dendrite pairs are checked not only for the label but also for the surrounding neurons.

Synapse formation

When synapse is formed, the axon changes to a growth state appropriate for releasing a transmitter.

In our model, the synapse is formed by changing the parameters of axons and dendrites. The axon-dendrite connection state is important to determine the neural death. The needed production rules are:

Axon-Dendrite synapse formation:

$$\mathcal{A}(label_1,\ldots) \quad \prec \mathcal{D}(label_1, conn, \ldots) : cond. \rightarrow \mathcal{D}(label_1, True, \ldots)$$
$$\mathcal{A}(label_1, conn, \ldots) \succ \mathcal{D}(label_1,\ldots) \quad : cond. \rightarrow \mathcal{A}(label_1, True, \ldots)$$

4.3 Modification of connections

Connection modification occurs in two basic cases: *Cell death* and *Elimination of connections*. These two cases forms the basis for anatomical plasticity in neural networks.

Cell death

Cell death is a major event in neuronal development. Survival depends on various factors, but at present a simplified rule for connectivity dependence will be used.

The production rule is a check on the axons and dendrites in order to find if they have connections or not. If there are no connections, production rule decrease in the neuron index value indicates the "liveliness" of the neuron. When there are no connections (eliminated by the production rules below) and the "liveliness" factor is below survival level, the neuron is simple removed from the model.

Cell liveliness:

$$\mathcal{D}(conn,\ldots) \prec(*) \mathcal{N}(liveliness,\ldots) (*)\succ \mathcal{A}(conn,\ldots)$$
$$: \sum conn = 0 \rightarrow \mathcal{N}(liveliness - 1,\ldots)$$

Cell death:

$$\mathcal{D}(conn,\ldots) \prec(*) \mathcal{N}(liveliness,\ldots) (*)\succ \mathcal{A}(conn,\ldots)$$
$$: \sum conn = 0 \wedge liveliness = 0 \rightarrow \emptyset$$

Elimination of connections

Connections may be removed from the model through cell death and inactivity of connection. Similar to the above check for a neural liveliness, there will be a check for the activity of connection. If the activity level falls below the survival level, the axon or dendrite is removed from the model. Activity is a function of time:

$$activity = \begin{cases} activity - 1 & \text{if not connected} \\ activity & \text{if connected} \end{cases}$$

Change in the activity can be implemented through the following production rule.

Dendrite activity:

$$\mathcal{D}(conn, activity,\ldots) \; : \; conn = 0 \; \rightarrow \; \mathcal{D}(conn, activity - 1,\ldots)$$

Axon activity:

$$\mathcal{A}(conn, activity,\ldots) \; : \; conn = 0 \; \rightarrow \; \mathcal{A}(conn, activity - 1,\ldots)$$

4.4 Adult plasticity

Adult plasticity is divided into: *Anatomical plasticity*,*Synaptical plasticity*, and *Signal propagation*. Usually in neural networks only the synaptical plasticity is modeled, but the need for anatomical plasticity is already widely accepted.

4.4.1 Anatomical Plasticity

In the previous section, the modification of connections through the early development process was described. Here the anatomical plasticity related to the learning process will be described.

Latest evidences have shown that the signal received through synaptical connection can change the developmental behavior of a neuron. However, the details of this phenomenon are yet to be uncovered (Schwartz & Kandel, 1991). There are various studies to model anatomical plasticity in neural networks on abstract level ((Szu, 1989, 1990; Alpaydin, 1991). However, it is not yet obvious how to define the growth in a meaningfull way.

For the time being, the phenomenon is defined as an additional rule to grow axons. The selection for applying the rule or not is based on the condition statement, which can, for example, check the internal state of the neuron.

Anatomical plasticity:
$\mathcal{N}(llist, \ldots) \;:\; cond. \;\rightarrow\; \mathcal{N}(cdr(llist), \ldots)\mathcal{A}(car(llist), \ldots)$

4.4.2 Synaptical Plasticity

In neural networks the most widely used synaptical plasticity is based on the Hebbian rule, according to which the connection activity between two neurons reduces the resistance of the connectivity. For example the back-propagation learning algorithm uses this principle and can be implemented through a method described in (Vaario et al., 1991). However, more detailed neural phenomena learning can be implemented in order to achieve the unsupervised learning behavior of biological systems.

In biological systems, the learning can be divided into *non-associative* and *associative* learning (Brown et al., 1991, page 109). Non-associative learning comprises *habituation*, *dishabituation* and *sensitization* (Carew, 1989). Associative learning is called Classical Conditioning. Higher-order features of conditioning can be achieved by combinations of the above mechanisms (Hawkins, 1989) as in the example the example of *Aplysia* below.

4.4.3 Signal Propagation

The propagation of input signals (sensors) toward output neurons (motor neurons) is implemented through a parametric changes in the model symbols.

Model signals move first from neuron to axon, then from action through the synaptical connection to the dendrite, and finally, from dendrite to the neuron. To model these three steps we need three production rules.

First, calculate the axon value by the function F_{out}:

$$\mathcal{N}(act, \ldots) \prec \mathcal{A}(label, out, \ldots) : cond. \rightarrow \mathcal{A}(label, F_{out}(act), \ldots)$$

Secondly, calculate the dendrite value based on axon signal by the function F_{in}:

$$\mathcal{A}(label, out, \ldots) \prec \mathcal{D}(label, in, \ldots) : cond. \rightarrow \mathcal{D}(label, F_{in}(out), \ldots)$$

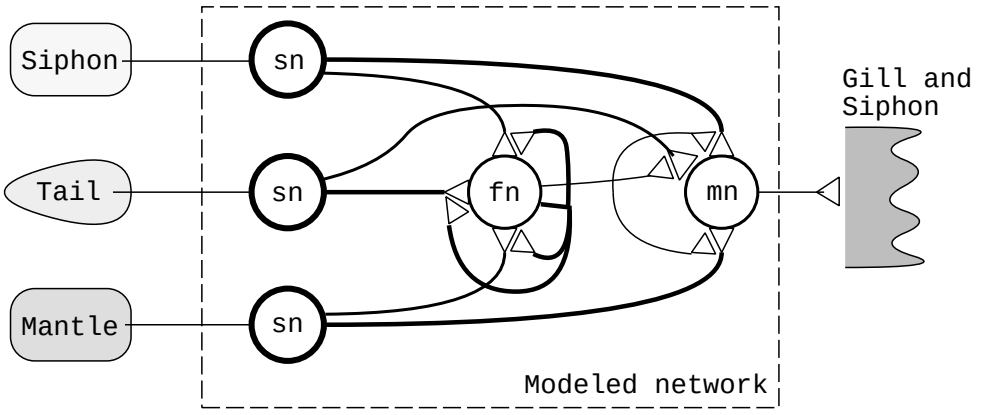

Figure 7: The Neural Network Model of Aplysia (Redrawn with modifications from (Hawkins, 1989, page 75))

Finally, calculate neuron activation value based on the dendrite values by the function F_{act}:

$$\mathcal{D}(label, in, \ldots) \prec \mathcal{N}(act, \ldots) : cond. \rightarrow \mathcal{N}(F_{act}(in), \ldots)$$

All functions ($F_{out}(act)$, $F_{in}(out)$, and $F_{act}(in)$) can be defined by the user.

5 Model of Aplysia

Aplysia is an excellent example for our model, because the animal exhibits a variety of both non-associative and associative types of learning that can exist in both short-term and long-term forms. Additionally the developmental nervous mechanism is well studied and its behavior is well established.

The model demonstrates the *Aplysia* neural network model according to Hawkins (1989). To keep the example simple, the greatly reduced aplysia model consists of sensory neurons (from siphon, mantle and tail), each of which is connected to a facilitatory neuron and a motor neuron. Each sensory neuron, while exciting a motor neuron, also excites a facilitatory neuron, which produces presynaptical facilitation at all of the terminals of the sensory neuron as in Figure 7.

5.1 Genetic growth

To create the basic neurons needed to construct the network, we use a simple initial pattern and a set of production rules specifying neural division.

The symbol with the parameters for a neuron is $\mathcal{N}(type, snList, gnList)$, where "type" defines the type of the neuron. Possible types are *sensor neuron* (*sn*), *facilitatory interneuron* (*fn*) and *motor neuron* (*mn*). The optional parameter *snList* is used to control the number of divisions. In our case it used to control the creation of three kinds of sensory neurons for each motor and facilitatory interneuron created. The word *gnList* describes the list of ancestor neurons to which the neuron should try to create connections.

With the initial neuron pattern $\mathcal{N}(mn, [tail, siphon, mantle], [\,])$ the following production rules produce the needed neurons.

$$
\begin{array}{lrcl}
(1) & \mathcal{N}(type, snList, gnList) & : & type = mn \\
 & & \rightarrow & \mathcal{N}(fn, snList, append(type, gnList)) \\
 & & & \mathcal{N}(type, gnList) \\
(2) & \mathcal{N}(type, snList, gnList) & : & type = fn \wedge snList \neq [\,] \\
 & & \rightarrow & \mathcal{N}(sn, append(type, gnList)) \\
 & & & \mathcal{N}(type, cdr(snList), gnList) \\
(3) & \mathcal{N}(type, snList, gnList) & : & type = fn \wedge snList = [\,] \\
 & & \rightarrow & \mathcal{N}(type, append(type, gnList))
\end{array}
$$

The production rule (1) models an asymmetric cell division where the motor neuron produces a facilitatory interneuron. The second production rule (2) models a similar cell division which produces three types of sensory neurons. The last production rule (3) models the preparation for the facilitation connection by appending the facilitatory interneuron type onto its own target neuron list.

5.2 Establishing connections

After the basic neurons have been created, the connections between them are created. The connections are divided into axon ($\mathcal{A}$ or, in the case of the facilitatory interneuron $\mathcal{F}$) and dendrite ($\mathcal{D}$) parts. Parameters include, for the axons of sensory neurons *calcium influx per action potential* (*Ca*) and *level of cyclic AMP* (*cAMP*); for the axon of facilitatory interneurons *number of action potentials* (*Spikes*); and for dendrites *postsynaptical potential* (*PSP*). The parameters are initialized with appropriate values, usually zero.

Neuron parameters are modified after the connections are established. New parameters are, for sensory neurons the *number of available calcium channels* (*Chs*), and for the facilitatory and motor neuron the *threshold value* (*Thresh*) and the *postsynaptical potential* (*PSP*).

Connection creation can be controlled by the following production rules:

$$
\begin{array}{lrcl}
(4)\ \mathcal{N}(type, gnList) & : & type = sn \wedge gnList \neq [\,] \\
 & \rightarrow & \mathcal{N}(type, cdr(gnList))\mathcal{A}(car(gnList), Ca_{init}, cAMP_{init}) \\
(5)\ \mathcal{N}(type, gnList) & : & type = sn \wedge gnList = [\,] \\
 & \rightarrow & \mathcal{N}(type, Chs_{max}) \\
(6)\ \mathcal{N}(type, gnList) & : & type = fn \wedge gnList \neq [\,] \\
 & \rightarrow & \mathcal{N}(type, cdr(gnList))\mathcal{F}(car(gnList), Spikes_{init}) \\
(7)\ \mathcal{N}(type, gnList) & : & (type = fn \vee type = mn) \wedge gnList = [\,] \\
 & \rightarrow & \mathcal{D}(type, 0)\mathcal{N}(type, Thresh_{init}, PSP_{init})
\end{array}
$$

Production rule (4) generates an axon with the initial values targeted to a neuron as specified in the *gnList*. After all requested axons have been generated, the states of the sensory neurons are changed by the production rule (5). Production rule (6) generates an axon for the facilitatory interneuron. Finally, the production rule (7) creates the dendrites for the motor and facilitatory neurons.

5.3 Adult plasticity

As an example of adult plasticity, we give a short example of the learning model. These production rules implement the learning of *Aplysia* as described in (Hawkins, 1989).

(8) $\mathcal{N}(type, Chs) : type = sn \rightarrow \mathcal{N}(type, f_{Chs}(Chs, input))$

Production rule (8) changes the number of available calcium channels (Chs) as a function of the old number and the input value (active or not active) of sensory neurons: $f_{Chs}(Chs, input) = Chs + C2 \times (Chs_{max} - Chs) - C1 \times Chs$.

If the sensory input is active, the number decrease a fixed percentage. On the other hand, the number recovers by another fixed percentage during each evolution step. This implements the *Habituation* learning mechanism.

(9) $\mathcal{N}(type, Chs) \prec \mathcal{A}(id, Ca, cAMP)\ (1) \succ \mathcal{F}(id, Spikes) : type = sn$
$\rightarrow \mathcal{A}(id, f_{Ca}(cAMP, Chs), f_{cAMP}(cAMP, Ca, Spikes))$

Production rule (9) changes the calcium influx per axon potential (Ca) as a function of cAMP level and the number of available calcium channels: $f_{Ca}(cAMP, Chs) = (1 + C3 \times cAMP) \times Chs/Chs_{max}$.

The cAMP level will be updated as a function of the old level, calcium influx, and the number of spikes: $f_{cAMP}(cAMP, Ca, Spikes) = C8 \times (cAMP + C6 \times Spikes \times (1 + C7 \times Ca))$.

The number of spikes is accessed through context matching; this effectively implements the facilitatory feedback. The right context match implements sensor-facilitatory synapses that resemble Hebb synapses.

(10) $\mathcal{A}(id, Ca, cAMP) <(*)\ \mathcal{D}(id, PSP) \rightarrow \mathcal{D}(id, f_{PSP}(\sum Ca))$

Production rule (10) collects the sensory neurons' calcium influx and adjust the postsynaptical potential as a function of these values: $f_{PSP}(\sum Ca) = C4 \times \sum Ca$

(11) $\mathcal{D}(id, newPSP) < \mathcal{N}(type, Thresh, PSP) : type = fn \vee type = mn$
$\rightarrow \mathcal{N}(type, f_{Thresh}(PSP), newPSP)$

Production rule (11) calculates a new value for the threshold value as a function of the postsynaptical potential: $f_{Thresh}(PSP) = C5 \times PSP$

(12) $N(type, Thresh, PSP) < \mathcal{F}(id, Spikes) : type = fn$
$\rightarrow \mathcal{F}(id, f_{Spikes}(Thres, PSP))$

Production rule (12) determines the number of spikes as a function of the threshold value and postsynaptical potential:

$$f_{Spikes}(Thresh, PSP) = \begin{cases} PSP - Thresh & \text{if } PSP > Thresh \\ 0 & \text{otherwise} \end{cases}$$

In functions above, the following constant values were used:

$C1$ =0.15
$C2$ =0.005
$C3$ =1.0
$C4$ =7.5
Chs=10.0
$C5$=0.9
$C6$=0.002
$C7$=15.0
$C8$=0.9975

All the above functions are described in detail in (Hawkins, 1989, page 80-81).

6 Implementation

L-systems are implemented through a knowledge base system, KAUS (Knowledge Acquisition and Utilization System) (Ohsuga, 1989, 1990).

For the developmental modeling, an operation was introduced to execute the L-system matching operation within the given production rules in the knowledge base.

The production rules were written in the following format:

```
rule:
    (| (match [ left pred right ] [ succ ] )
            ~(condition)
    ).
```

This is equivalent to the production rule in the L-system formalism.

$$rule : left \prec pred \succ right : condition \rightarrow succ$$

The above rules can be asserted in a world concept. The selection of worlds as well as modification of the rules can be implemented through the multi-level concept of KAUS. The latter feature is necessary in the future work to model more sophisticated management systems for the production rules.

7 Conclusion

In the paper we have shown that it is possible to construct and simulate a neural network based on the emergent of production rules. The reported results should be considered as the first step on the long way to find an efficient modeling method for artificial neural networks that not only model a well-known behavior function, but the emergent behavior found in biological nervous systems.

At least two kinds of critiques can be given for the present work. First, the example used was quite simplified and without any practical usability. Even this is true, it should be pointed out, that example was selected *only* to demonstrate the modeling method, but *not* to demonstrate the merits of created neural networks. Secondly, the problem to create the necessary production rules and the consistence of them were not discussed at all. The production rules were designed as general knowledge base rules. However, there are two points to be noticed.

1. Rules are highly re-usable. The five learning rules (8-12 apply to basic learning phenomena and may be implemented for all kind of network structures.

2. Rule generations may be automated. Being separate and well defined, the production rules are easily converted into a genetic presentation form and, furthermore, the *genetic algorithms* may be applied for them. This is to be investigate in the future.

The application for these ideas is to design a neural architecture on the computer through a simulation of evolution process, and after a neural control architecture has been found, to "copy" the solution to the application environment. The most suitable application area is autonomous control systems, where the system should adapt to the surrounding environment to some extent.

The research work conducted this far, shows that it is possible to construct a neural network based on production rules and simulate some basic behavior patterns based on similar production rules. The real exploration of the production rules to implement new type neural network applications is now ready to start.

Acknowledgment

We thank Dr. Yamauchi for his valuable help in the implementation of the production rule evaluation system through KAUS. We wish to thank Mr. Iannis Zannos for his comments and suggestions concerning the manuscript.

Reference

Alpaydin, E. (1991). Gal: Networks that grow when they learn and shrink when they forget. Tech. rep. TR-91-032, International Computer Science Institute.

Beer, R. D. (1990). *Intelligence as Adaptive Behavior*. Academic Press.

Beer, R. D., & Chiel, H. J. (1990). Neural implementation of motivated behavior: Feeding in an artificial insect. In Touretzky (Touretzky, 1990), pp. 44–51.

Beer, R. D., & Chiel, H. J. (1991). The neural basis of behavioral choice in an artificial insect.. In Meyer, & Wilson (Meyer & Wilson, 1991), pp. 247–254.

Beer, R. D., Chiel, H. J., & Sterling, L. S. (1990). A biological perspective on autonomous agent design. In Maes (Maes, 1990), pp. 169–186.

Belew, R. K., McInerney, J., & Schraudolph, N. N. (1992). Evolving networks: Using the genetic algorithm with connectionist learning. In Langton et al. (Langton et al., 1992), pp. 511–548.

Brooks, R. A., & Viola, P. A. (1990). Network based autonomous robot motor control: from hormones to learning. In Eckmiller, R. (Ed.), *Advanced Neural Computers*, chap. 6: Motor Control with Neural Networks, pp. 341–348. Elsevier Science Publisher B.V. (North-Holland).

Brooks, R. A. (1990). Elephants don't play chess. In Maes (Maes, 1990), pp. 3–15.

Brooks, R. A. (1991a). Challenges for complete creature architectures.. In Meyer, & Wilson (Meyer & Wilson, 1991), pp. 433–443.

Brooks, R. A. (1991b). Intelligence without representation. *Artificial Intelligence*, pp. 139–159.

Brown, M., Hopkins, W., & Keynes, R. (1991). *Essential of Neural Development.* Cambridge University Press.

Burnod, Y. (1990). *An Adaptive Neural Network - the corebal cortex.* Prentice Hall, London.

Byrne, J. H., & Berry, W. O. (1989). *Neural Models of Plasticity - Experimental and Theoretical Approaches.* Academic Press, Inc., San Diego.

Carew, T. J. (1989). Developmental assembly of multiple components of learning and memory in aplysia. In *Neural Models of Plasticity - Experimental and Theoretical Approaches* (Byrne & Berry, 1989), chap. 2, pp. 22–45.

Chalmers, D. J. (1990). The evolution of learning: An experiment in genetic connectionism. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., & Hinton, G. E. (Eds.), *Proceedings of the 1990 Connectionist Models Summer School* San Mateo, CA. Morgan Kaufmann.

Edelman, G. M. (1987). *Neural Darwinism - The Theory of Neuronal Group Selection.* Basic Books, Inc.

Gille, J., Wegrzyn, S., & Vidal, P. (1989). A model for developmental systems - ii. generating word with an operating system. *Automatica*, *25*(5), 707–714.

Gliff, D. (1991). Computational neuroethology: A provisional manifesto.. In Meyer, & Wilson (Meyer & Wilson, 1991), pp. 29–39.

Harp, S. A., & Samad, T. (1991). Genetic synthesis neural network architecture. In *Handbook of Genetic Algorithms.* Van Nostrand Reinhold.

Harp, S. A., Samad, T., & Guha, A. (1990). Designing application-specific neural networks using the genetic algorithm. In Touretzky (Touretzky, 1990), pp. 447–454.

Hawkins, R. D. (1989). A simple circuit model for higher-order features of classical conditioning. In *Neural Models of Plasticity - Experimental and Theoretical Approaches* (Byrne & Berry, 1989), chap. 5, pp. 73–93.

Jacobson, M. (1991). *Developmental Neurobiology* (Third Edition edition). Plenum Press.

Jefferson et.al., D. (1990). Evolution as a thema in artificial life: The genesys/tracker system. Technical Report UCLA-AI-90-09, Artificial Intelligence Laboratory, Computer Science Department, University of California, Los Angeles, California 90024.

Jefferson et.al., D. (1992). Evolution as a theme in artificial life; the genesys/tracker system. In Langton et al. (Langton et al., 1992), pp. 549–576.

Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (Eds.). (1991). *Principles of Neural Sience* (Third Edition edition). Elesevier Science Publishing.

Langton, C. G. (Ed.). (1989). *Artificial Life*. Addison-Wesley Publishing Company.

Langton, C. G., Taylor, C., Farmer, J. D., & Rasmussen, S. (Eds.). (1992). *Artificial Life II*. Addison-Wesley Publishing Company.

Lindenmayer, A., & Prusinkiewicz, P. (1989). Developmental models of multicellular organism: A computer graphics perspective. In Langton (Langton, 1989), pp. 221–250.

Maes, P. (Ed.). (1990). *Desinging Autonomous Agents*. Elsevier Science Publisher.

Meyer, J.-A., & Wilson, S. W. (Eds.). (1991). *Proceedings of the First International Conference on Simulation of Adaptive Behavior: From animals to animats*. A Bradford Book, MIT Press.

Ohsuga, S. (1989). Toward intelligent CAD systems. *Computer-Aided Design*, *21*(5), 315–338.

Ohsuga, S. (1990). Framework of knowledge-based systems - multiple metlevel architecture for representating problems and problem-solving processes. *Knowledge-Based Systems*, *3*(4).

Prusinkiewicz, P., & Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag.

Schwartz, J. H., & Kandel, E. R. (1991). Synaptic transmission mediated by second messengers. In Kandel et al. (Kandel et al., 1991), pp. 173–193.

Szu, H. H. (1989). Reconfigurable neural nets by energy convergence learning principle based on extended McCulloch-Pitts neurons and synapses. In *Int. Joint Conference on Neural Networks*, pp. I–485–I–496 Washington, D.C.

Szu, H. H. (1990). Neural networks based on peano curves and hairy neurons. *Telematics and Informatics*, *7*(3/4), 403–430.

Touretzky, D. S. (Ed.). (1990). *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann Publishers.

Vaario, J., Ohsuga, S., & Hori, K. (1991). Connectionist modeling using Lindenmayer systems. In Ohsuga, S., Kangassalo, H., Jaakkola, H., Hori, K., & Yonezaki, N. (Eds.), *Information Modeling and Knowledge Bases: Foundations, Theory, and Applications*. IOS Press, Amsterdam.

Wegrzyn, S., Gille, J., & Vidal, P. (1989). A model for developmental systems - i. generating word without any operating system. *Automatica*, *25*(5), 695–706.