Toward Evolutionary Design of Autonomous Systems

Jari Vaario* Koichi Hori Setsuo Ohsuga
The University of Tokyo
Research Center for Advanced Science and Technology
Komaba 4-6-1, Meguro-ku, 153 Tokyo, Japan
Phone: +81-3-3481-4486, FAX: +81-3-3481-4585
E-mail: jari@ai.rcast.u-tokyo.ac.jp

Original Manuscript: October 16,1992
Revised: November 22,1992
The International Journal in Computer Simulation — Special Issue on High-Autonomy Systems

Abstract

An evolutionary method for designing autonomous systems is proposed. The research is a computer exploration on how the global behavior of autonomous systems can emerge from neural circuits. The evolutionary approach is used to increase the repertoire of behaviors.

Autonomous systems are viewed as organisms in an environment. Each organism has its own set of production rules, a genetic code, that gives birth to the neural structure. Another set of production rules describe the environmental factors. These production rules together give rise to a neural network embedded in the organism model. The neural network is the only means to direct reproduction. This gives rise to intelligence, *i.e.* organisms which have "more" intelligent methods to reproduce will have a relative advantage for survival.

Keywords: Evolutionary Design, Autonomous Systems, Emergent Computation, Artificial Life, Adaptive Behavior, Artificial Neural Networks, Genetic Algorithms.

1 Introduction

Recently it has become widely accepted that the explicit design of autonomous systems is overwhelming difficult task. The Brooks approach, where the human is included in the design loop, has it's limits: complexity of design process and giving only solutions for predefined tasks.

When the number of possible behaviors increases, the complexity of the system (parameters for behaviors, selection of a particular behavior etc.) increases exponentially. At a certain point this will exceed human capacity and the design

*To whom correspondence should be addressed.

process will slow down. Thus we should create a design system where the needed "engineering" could be as small as possible.

Another aspect is that autonomous systems should be capable of self-modifications during their life-time. During the design phase all possible situations, what the system may face, are not accountable. The predefined solutions given by a human will not be sufficient. The system should itself find the solutions through an individual and/or evolutionary adaptation to the environment.

Individual adaptation, especially learning, has been a topic of various researches. Despite of the success made, for example reinforcement learning, it has not solved all problems. We need also the evolutionary approach. Brooks himself has acknowledged this in [1].

In this paper we propose a method where behavior is not defined explicitly. Instead, it emerges from the neural network level. Furthermore the neural network, its input (sensors) and its output (effectors) will each adapt to the environment in a dynamic process. Thus the behavior is a dynamic function of these adaptation processes.

We would like to emphasise the adaptation of sensors and effectors to the environment. As Kugler [2] has shown, these are very important elements for the modification of behavior. By changing the interpretation of input and/or output values of neural network, the behavior of system will change remarkably.

We use abstract computer models, instead of real robotic simulations. The goal is to find a method to model intelligence as adaptive behavior, rather than simulate real robots. We include development process and plasticity (learning) as individual adaptation, and natural/sexual selection and genetic changes as evolutionary adaptation.

This work is related to Artificial Life [3, 4, 5] as well as to Simulated Adaptive Behavior [6], where biological systems have been used as a source of inspiration. These approaches are a response to an argument that AI has failed to model *intelligent behaviors*, but modelled only *intelligent solutions* instead.

The *Genetic Algorithms* are often used with a well-defined fitness function to implement the evolution process. Our approach is to view the evolution process as a tool for growing gradually the complexity of systems *i.e.* the repertoire of behaviors. This is an open-ended evolution, where we can not predict the final complexity of the system.

This paper does not make use of any existing learning algorithms developed for *artificial* neural networks. In our earlier work [7] we have introduced some *biologically* inspired learning models. In this paper we concentrate on the structural networks without any learning capabilities.

2 Autonomous System and Neural Network

Autonomous systems have many definitions [8]. We have adapt here the definition of Cariani [9, 10]: A highly autonomous system is viewed generally as consisting of *syntactical adaptation* (*i.e.* capable to alter its contingent upon its performance in an external environment) and *semantical adaptation* (*i.e.* capable to construct and selecting its sensing and effecting elements upon its performance). This kind of autonomous system is called *epistemically autonomous*.

The conceptual graph is shown in Figure 1.

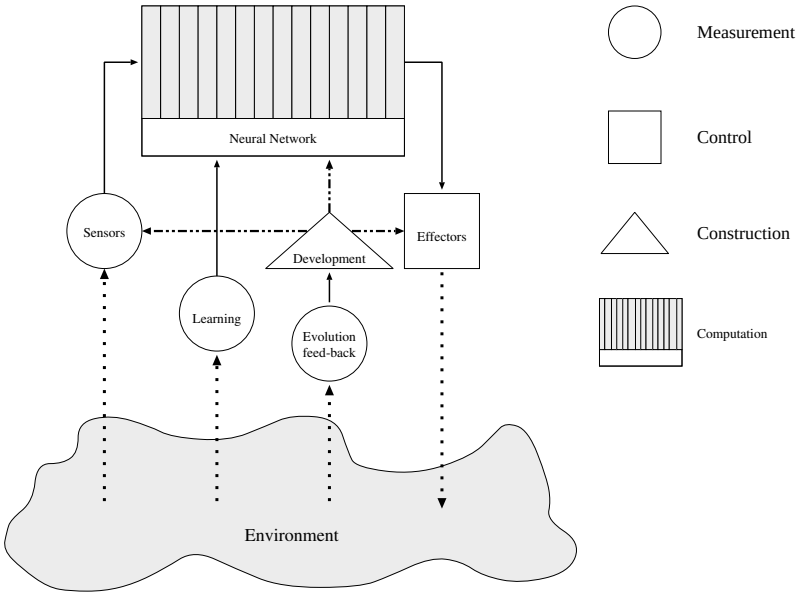

Figure 1: Autonomous Systems as a general evolutionary device with the construction part as development process. (Modified drawing after Cariani [9].)

The computation parts may be implemented by various means. We have selected the neural network as implementation tool. Most neural network research has been concentrating on *syntactically adaptive* systems, *i.e.* having no construction part at all. In our research we have concentrated in finding a method to implement the construction part, rather than developing a sophisticated learning mechanism.

The construction is divided into external construction and internal construction. The external construction part will create the measure, control, and computational parts of the system (development process) based on the evolutionary feed-back (sexual and natural selection). The internal construction is a reconfiguration of the network for a new interpretation of sensor input and effector output. Both of these are explored in this paper.

3 Modeling Method

Emergence has been described as a phenomenon where interesting global behavior *emerges* from many local interactions. For a computation of the emergent process we need the following constituents: [11]

- A collection of agents, each following explicit instructions;
- Interactions among the agents (according to the instructions), which form implicit global patterns at the macroscopic level *i.e.* epiphenomena;
- A natural interpretation of the epiphenomena as computations.

In our model the lower level agents are neurons, that have instructions for cell divisions and axon-dendrite growth, *i.e.* genetic coding. Also sensors and

effectors are part of lower level agents, although explicitly given without being result of cell divisions. These agents are grouped to belong to an organism. Furthermore each organism belongs to an environment.

The interactions between these agents are simulated as physical laws. As the "correct" simulation of physical laws would need to start from the atomic, or even lower, level, we have restricted ourselves in describing the laws more like a symbolical result of them. Thus, for example, interactions between two neurons are described as production rules, that change the attributes of neurons at symbolical level. However, it is up to the user to define the level on which the changes will occur.

The natural interpretation of epiphenomena is such behavior as following, selection, and change of target. These behaviors are the result of interactions between the lower level agents, and these are not explicitly defined. The behaviors arise from the structure of the network, that itself can change according to the evolution process.

Figure 2 gives an overview of the models as organized in the levels, and interactions between these levels.

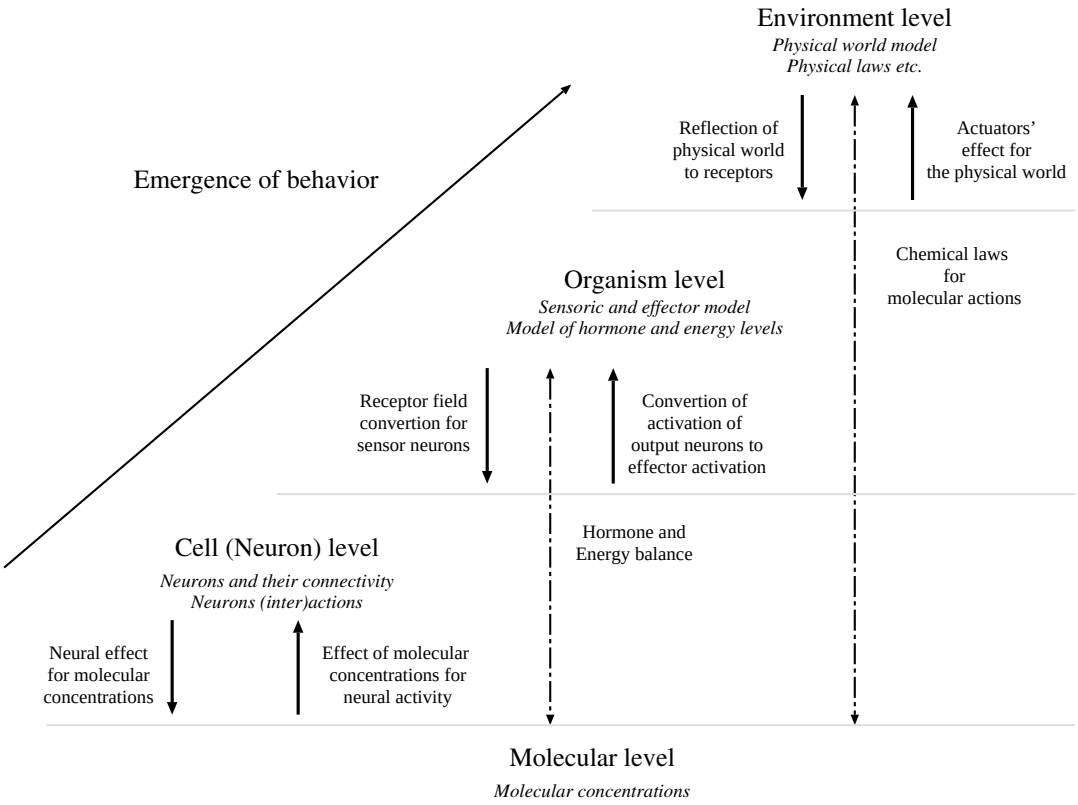

Figure 2: Organisation of modeling levels and their interactions

3.1 Production rules

The implementation is based on the idea of Lindenmayer systems [12, 13]. 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:

$$predecessor(parameters) : condition \rightarrow successor(expression),$$

where predecessor is rewritten by successor if the condition is fulfilled.

However, some modifications have been done for efficiency and to introduce *interleaved L-systems*. In interleaved L-systems, one L-system may have other L-systems inside of it, thus being able to modify them as well. For example, the L-system for environment has multiple organisms which each are defined by L-systems. The environment L-system has control over for the organisms to remove, modify, *etc.* them.

As an example let's assume that we have an environment with two organisms. This is defined as

```
Environment(Constraints, ...,
            Organism1( GeneticCode, ..., Neurons( attributes ),
            Organism2( GeneticCode, ..., Neurons( attributes), ... ),
            ... )
```

"Constraints" defines the environment depending on the production rules that are common for each organism. "GeneticCode" defines the organism depending on the production rules for cell divisions, axon-dendrite growth *etc.* "Neurons" defines the initial cell for the growth process. The execution of the "GeneticCode" production rules is done without an explicit definition for it.

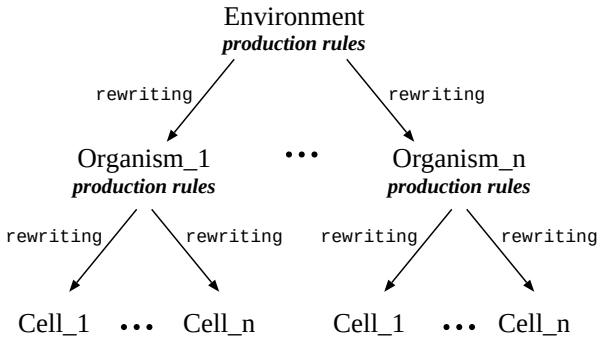

Figure 3: The interleaved L-system, where production rules are executed first at the environmental level, then at the organism level in parallel.

For example "Constraints" can include the following production rules.

```
mating(org1,org2) : intersect(org1,org2) &&
                    (org1.Sex == female) && (org2.Sex == male)
                  → org1(State=Pregnant,
                         ChildGene=merge(org1.GeneticCode,
```

```
                                                    org2.GeneticCode))
                        org2(...)
birth(org1) : (org1.Sex == female) && (org1.State == Pregnant)
          → org1(State=Normal)
             new_org(Sex=random(female,male),
                     GeneticCode=org1.ChildGene)
```

At the first evaluation step, if two organisms intersect, the female will become "pregnant" and the childs genetic information will be merged from both organisms by the *merge(org1.GeneticCode,org2.GeneticCode)* function. After the next evaluation step, the female will give birth to a new organism posessing new genetic information.

Production rules like the above should be replaced by interactions on the cell level, but for computational reasons, the neural level has been used exclusively for movement control.

3.2 Innate for Simulation

The innate consists of two parts: *production rules for "Constraints"* and *genetic code for initial organisms*. The production rules that define "Constraints" are basically same for each simulation. The initial genetic code for organisms is required only to be capable to start the reproduction process within "Constraints".

The definitions for sensors and effectors are given in this paper explicitly, but the definition for neurons is indirect through a genetic code that produces the growth of network. This growth is based on the idea of recurrent construction, where a symmetry plays an important role. The symmetric nervous system can be produced by using a simple fractal-like construction method, where a same pattern is repeatedly used with only slight changes in its parameters.

The drawback of this approach is that it is difficult to find correct production rules to produce the desired result. Advantages are that with a simple modification for these production rules we can change the result completely. The first can be avoided with a simulation of evolution process. Where as the second one is very useful for this evolution process.

In Nature the innate has developed during a long time from molecular level. We would like to skip this phase and introduce, found through a 'trial-and-error' method, an initial organism, that will give birth for other organisms. The sufficient initial organism is described hereafter.

4 Genetic coding

The genetic code expresses the genotype of the autonomous system. It includes the definition of the neural network, sensors, and effectors.

The genetic code is not coded for bit stream presentation. Instead, a symbolic presentation is used to enable genetic engineering like handling during crossover and mutation operations.

The development from genotype to phenotype should occur in close interaction within the environment. In the following the environmental effect for the transcription process of genetic code is made possible. Such developmental effects are visible for example in the growth of neural connections and in the synapse formation.

4.1 Sensors

The following sensor attributes are used for sensors: Relative position in the organism coordination, intensity of stimulus source as a function of distance and angle, type of stimulus the sensor is sensitive to, and received perception as a natural logarithm value of the stimulus.

For example sensors for an opposite sex, *i.e.* some chemical compounds, have the following genetic code:

$$\text{sensor}(Type = sex, X = 10, Y = 0, \delta = 0.35, Perception = 0)$$

Type the type of sensor, for example sensor for *opposite sex*
X x-position of sensor in the organism coordination
Y y-position of sensor in the organism coordination
$\boldsymbol{\delta}$ sensing profile, for example value 0.1 for a sharp, 0.35 for a normal, and 1.5 for a wide sensing at the heading angle
Perception the received stimulus

The function used to update the perception value is

$$Perception = \sum_{i}^{organs} (org_i.Stimulus * e^{-(\Theta/\delta)^2/2} * (1 - \frac{1}{1 + e^{(Dist_{max}/2 - Distance)/2}}))$$

where Θ is the angle between the stimulus source and the heading direction. Figure 4 demonstrates the function with two δ values.

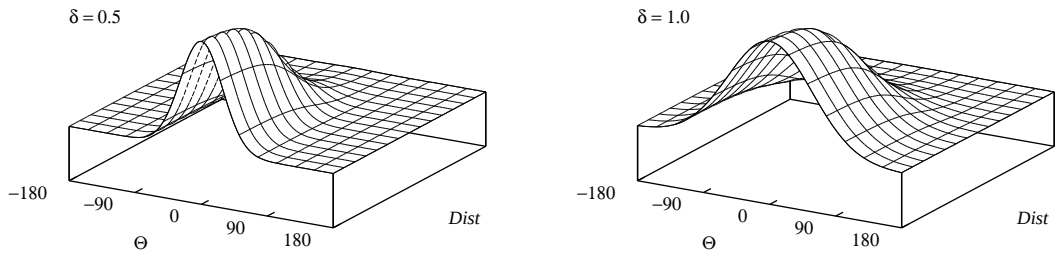

Figure 4: Perception as function of angle and distance.

This function is too simplified for simulating a real physical device. For that purpose it should be changed to correspond to a response function of some physical sensor. Also some noise should be added in the function.

4.2 Effectors

The effectors correspond to muscles of the organism. The input value causes an extensor or a reflexor force for a structure ("bone") of the organism. In this paper, for simplicity, an explicit definition for a movement is given. This includes the positions of effectors, and the force on what they effect for the organism when activated.

For example a effector for a step forward movement have the following genetic code.

effector($Type = step, X = -10, Y = 0, \Delta F_x = 1, \Delta F_y = 0$)

Type the type of effector
X x-position of effector in the coordination of organism
Y y-position of effector in the coordination of organism
$\Delta \boldsymbol{F}_x$ the force at x-direction
$\Delta \boldsymbol{F}_y$ the force at y-direction

The effectors cause the following forces for the organism.

$$F_x = \sum_i^{effectors} Activation_i * (\cos(Heading) * \Delta x_i - \sin(Heading) * \Delta y_i)$$

$$F_y = \sum_i^{effectors} Activation_i * (\sin(Heading) * \Delta x_i + \cos(Heading) * \Delta y_)$$

$$M = \sum_i^{effectors} Activation_i * (X * \Delta y_i - Y * \Delta x_i)/\sqrt{X_i^2 + Y_i^2}$$

The position and heading of organism is updated relative to these forces.

4.3 Neurons

The neuron circuit is defined by the following definition.

n($State = Embro, ConnList = [[[sex, sen], [step, eff]]]$)

This tells that the neuron will have connections for an 'opposite sex sensing' sensor ($[sex, sen]$) and a 'stepping' effector ($[step, eff]$).

```
segregation() : (State==Embro)&&(ConnList!=[])
              → n(Side=left,Connections=ConnList)
                n(Side=right,Connections=ConnList)
segregation() : (State==Embro)&&(Connections!=[])
                → n(Connections=rest(Connections))
                  n(State=Grow,ConnectList=Connections=first(Connections))
segregation() : (State==Embro)&&(Connections==[])
              → n(State=Dead)
```

The above production rules gives a network with two neurons. The neurons will have the following parameters.

n($Side = left, right, State = Grow, Connections = [[sex, sen], [step, eff]]$)

4.3.1 Growth of connections

To create connections for sensors and effectors the following production rules are used. Here we introduce a model where multiple connections are created, and the elimination of unnecessary connections is done later by other production rules. The growth of connections is controlled by a 'GrowthFactor', which itself is dynamically updated based on the connections a neuron has. Thus, when a neuron looses some connection the 'GrowthFactor' will fell, and the neuron will start to grow new connections.

```
connection() : (GrowthFactor<StopGrowing)&&(State==Grow)&&
               (Connections!=[])
             → n(Connections=rest(Connections))
               c(Connected=False,ToConnection=first(Connections))
connection() : (GrowthFactor<StopGrowing)&&(State==Grow)&&
               (Connections==[])
             → n(Connections=ConnectList)
connection() : (GrowthFactor>=StopGrowing)
             → n(State=Mature)
```

4.3.2 Creation of synaptical connections

As a result of these production rules, with appropriate growth control parameters, a number of connections with labels ($ToConnection$) to connect to sensors ($[sex, sen]$) or to effectors ($[step, eff]$). These labels will be used to create the synaptical connections by the following production rules.

```
connect(s,c) : (!c.Connected)&&(first(rest(c.ToConnection))==sen)&&
               (s.Type==first(c.ToConnection))
             → c(Connected=True,Type=dend) s(Connection++)
connect(e,c) : (!c.Connected)&&(first(rest(c.ToConnection))==eff)&&
               (e.Type==first(c.ToConnection))
             → c(Connected=True,Type=axon) e(Connection++)
```

These production rules will result the network shown in Figure 5.

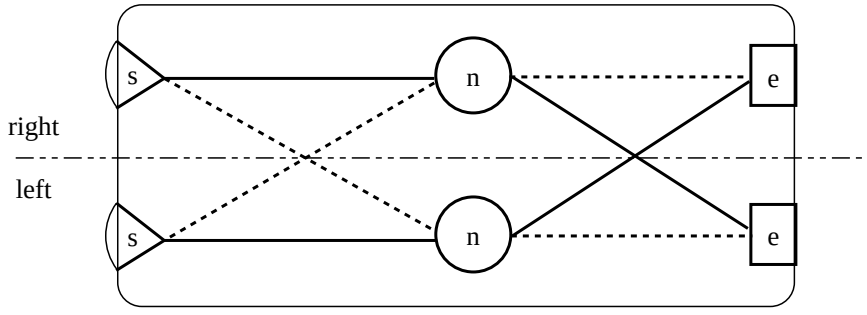

Figure 5: The network before elimination of unnecessary connections (shown by dot line).

In biological systems the creation of synaptical connections is a complicated phenomenon. The above production rules give only the basic idea that the

target recognition should be based on the labels, that are inherited through the cell division process. The second basic idea is that we should not try to create only the exactly needed connections, but instead plenty of them, and then use different rules for elimination of unnecessary ones.

4.3.3 Elimination of connections

This network includes unnecessary connections that should be eliminated before the network can function correctly. For this example the following approximation may be used: *In order to get close to something (*i.e. *attractor signals), the sensors must be connected directly to same side neurons, and the effectors connected to the opposite side.* This principle could be found easily through 'trial-and-error' during the evolution process. This can be implemented through the following production rules.

disconnect(s,c) : (c.Side!=s.Side)
→ c(Connected=False) s(Connection−−)
disconnect(e,c) : (c.Side!=e.Side)
→ c(Connected=False) e(Connection−−)

As above these production rules should be replaced by rules describing the biological phenomenon of connection elimination. In fact, these should be integrated with learning rules, that are restricted out of the scope of this paper.

4.3.4 Extension of the structure

To create a new behavior for tracking another object the above description for neurons can be changed.

$\mathrm{n}(State = Embro, ConnList = [[[sex, sen], [step, eff]], [[food, sen], [step, eff]]])$

This change will result in the network shown in Figure 6.

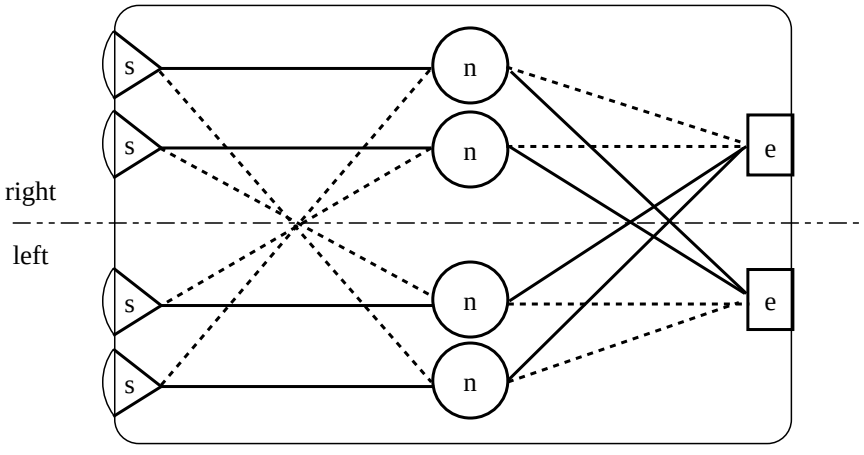

Figure 6: The network before elimination of unnecessary connections (shown by dot line).

All above production rules will remain as before. However, the problem will be how the signal from two sources will be selected, *i.e.* focus of interest. A solution for this will be discussed below.

5 Behavior

The behavior is defined as an action taking place by existence of a certain stimulus. Thus it is a high level feature that emerges from lower level actions. We do not explicitly define behavior, but we are interested in to find a organism capable of the following behavior types:

- For a stimulus of opposite sex, a movement toward the stimulus.
- For a stimulus of food, a movement toward the stimulus.
- Change from one stimulus tracking to another based on the hormone balance.

These basic behavior types can be combined to create new behaviors. The idea is that the system should automatically combine these basic behaviors in order to create new behaviors.

5.1 Stimulus tracking

The simplest form of movement is toward a stimulus. In a simplest form this can be implemented by two sensors and two effectors that are cross connected as shown in Figure 7.

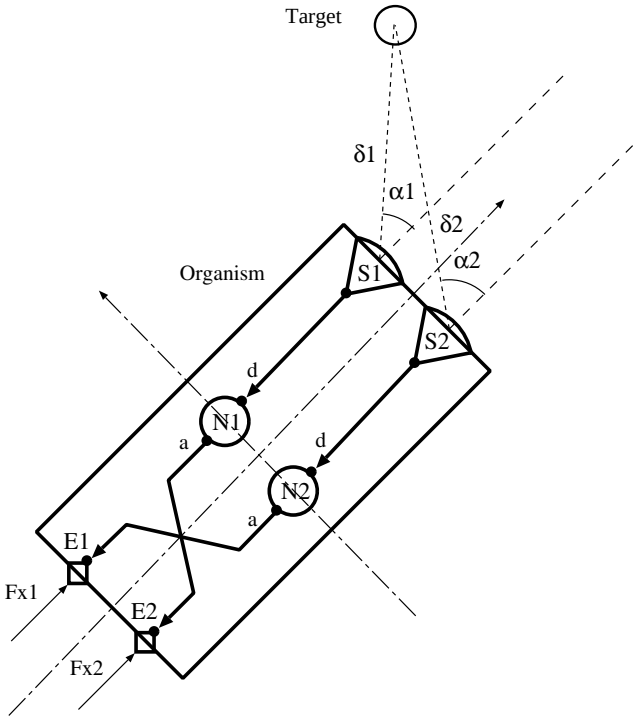

Figure 7: An example of a simple animat in the environment.

In the Figure 8 an example of this is given. The used production rules to propagate signals in the network are as follows.

propagate(s,d) → d(Activation=s.Perception)
s(Perception=0.1*Perception)

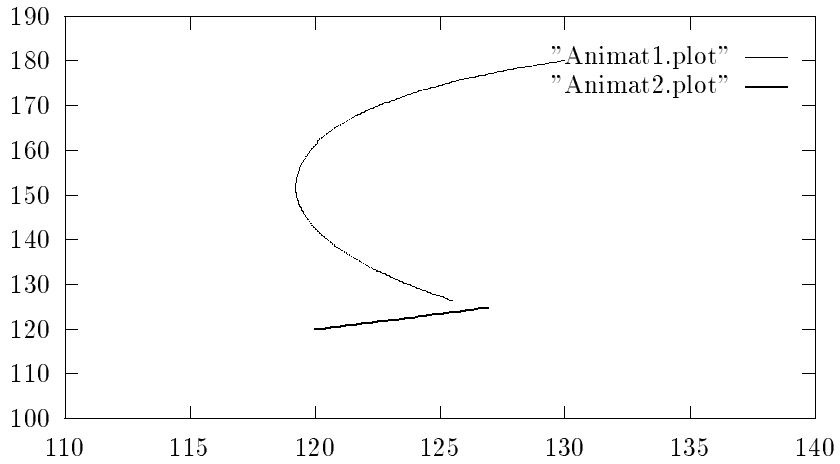

Figure 8: A male (Animat1, initial heading angle 220 degrees) is sensing a female (Animat2, heading angle 35 degrees) and moving toward stimulus.

propagate(d,n) : (n.State==Mature)
→ n(Activation=Activation + d.Activation)
d(Activation=0)
propagate(n,a) : (n.State==Mature)
→ a(Activation=n.Activation)
n(Activation=0)
propagate(a,e) → e(Activation=a.Activation)

The above trivial propagation rules will give surprisingly good results. However, these can be used only for creating a rough movement in the stimulus direction. The final approaching has been done using a separate network, as will be shown later.

The same simple network is able to select a more "stimulating" object as will shown in Figure 9.

5.2 Change of focus

Each effector is driven by several networks. A switch between these networks to be used to drive the effectors are done based on a "hormone" balance (sexual desire and energy level indicator).

For example in Figure 10 a male animat is first tracking opposite sex, and, after mating, starts to look for food. The both stimulus tracking is done by a similar network. However, an additional threshold function is introduced by the following production rule.

propagate(n,a) : (n.State==Mature)
→ a($Activation = (n.Activation > n.Threshold ? n.Activation : 0)$

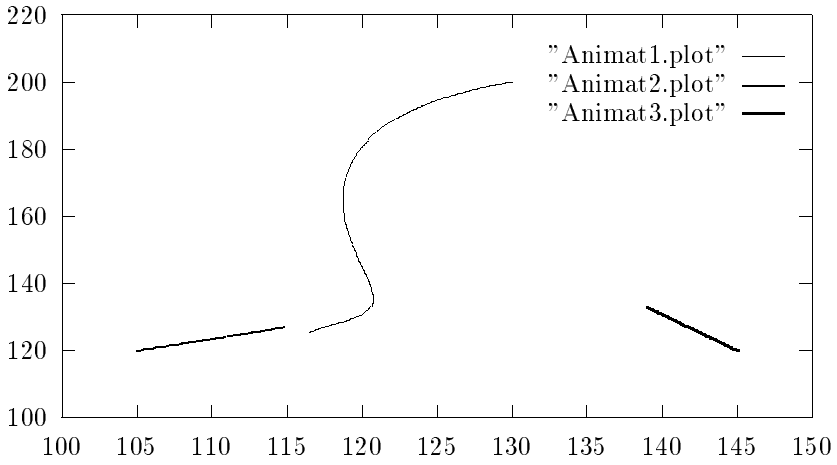

Figure 9: A male (Animat1, initial heading angle 220 degrees) is sensing two females (Animat2, heading angle 35 degrees, and Animat3, heading angle 225 degrees) and moving toward them. At a specific moment the stimulus from one of the two females becomes larger and the male will direct for her.

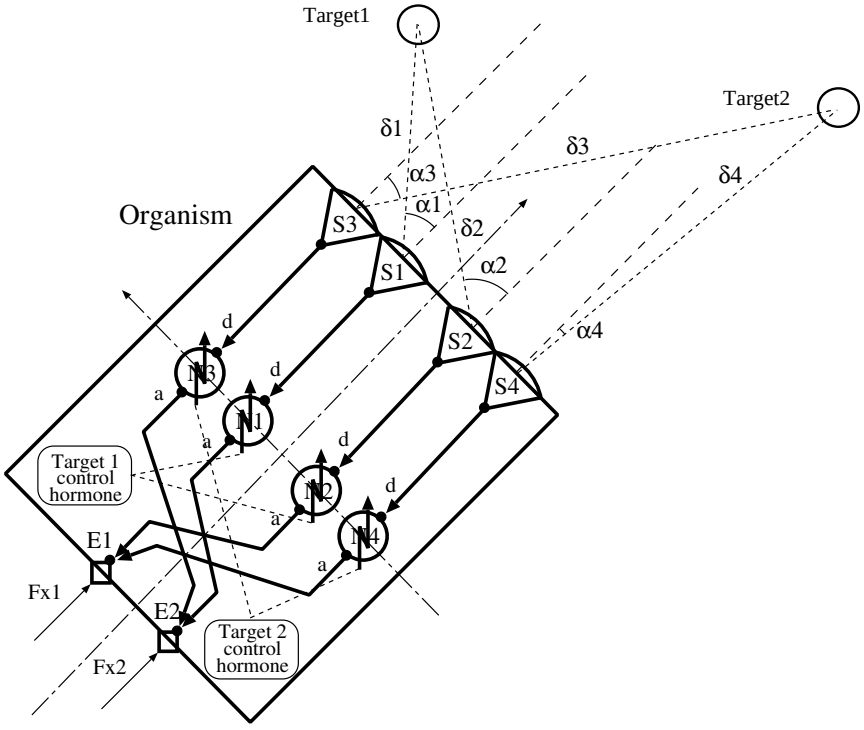

Figure 10: Two sensor-neuron pairs to drive the same effectors.

n(Activation=0))

The threshold value is a function of time and is determined by the following production rules where *SexHormone* and *EnergyLevel* are updated by other production rules.

hormone(org,n) : (n.Type==Sex)
$\rightarrow$ n($Threshold = 1/(1 + e^{(org.SexHormone_{Max}/2 - org.SexHormone)/2})$)
hormone(org,n) : (n.Type==Food)
$\rightarrow$ n($Threshold = 1/(1 + e^{(org.EnergyLevel_{Max}/2 - org.EnergyLevel)/2})$)

The resulting behavior is shown in Figure 11, where "Animat1" is a male, that is first after food ("Animat3") and then after female ("Animat2").

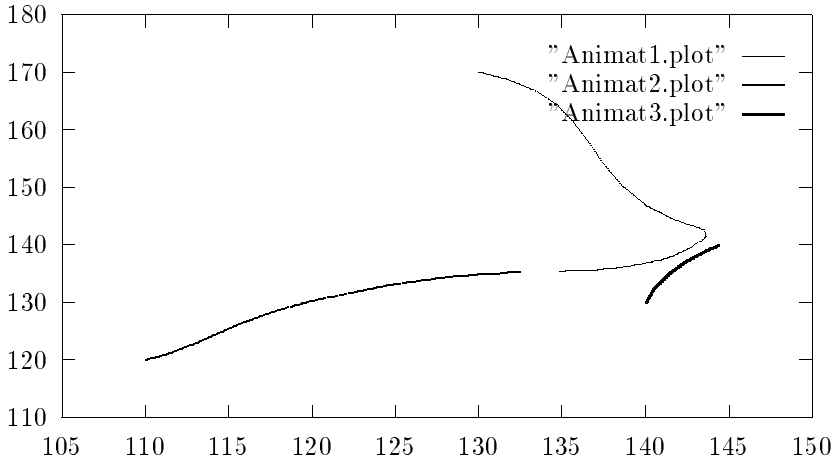

Figure 11: A male (Animat1, initial heading angle 325 degrees) is sensing female (Animat2, initial heading angle 35 degrees, sensitive for male signal trying to find it) and food (Animat3, initial heading angle 80 degrees, sensitive for male signal trying to avoid it). Because the focus is on the food, the male will direct toward food, and afterwards change the focus on the female.

6 Evolution of Organisms

This far we have described how the behavior of an individual autonomous system emerges from simple rules. This gives us the basic elements for an evolution process. In the following we will describe how to make use of these elements to evolve the organisms.

The key component of evolution is selectionism, *i.e.* what organisms are selected to reproduce new individuals. This is also the seed for "intelligence".

Each organism effects for its possibility to be selected through its behavior. The capability to observe its own best is directing the evolution. In the words

of Erwin Schrödinger "Individuals which — by change or intelligence — change their behaviour accordingly will be more favoured, and thus selected" [14]. This together with Piaget's work of "Adaptation and Intelligence" [15] emphasizes the active role of individuals for directing their evolution against Darwin's passive selection principle [16].

6.1 Sexual Selection

Bisexuality is needed for crossover operation. This has a remarkable effect for the evolution of complexity. Sexual selectionism is not part of the environmental description, but part of each individual's behavior. In other words, which particular opposite sex attracts one is based on the structure of the nervous system.

In our simulation the attraction is modeled through a simulation of odor. Each animat has a value for the "odor", that can be updated dynamically. This odor is used to calculate the "attraction" of another individual and thus directly effects for the selection.

6.2 Natural Selection

An energy modeling is used to determine liveliness of an organism. Each organism is given some initial amount of energy, that will decrease as function of time and action. If the energy gets below some given level, the organism "dies".

This is implemented through the following production rule.

survival(env,org) : (org.Energy $<$ env.SurvivalLevel)
$\rightarrow$ org(State=Dead)

Another possibility for an organism to "die" is to become eaten by another organism. Thus the fitness function is depending on how well the individuals can find food and how well they succeed avoiding prediators. It is notable that this function is a relative function of the environment, rather than explicitly defined as in the case of Genetic Algorithms.

7 Genetic Changes

Genetic changes consist of two operations: Crossover and Mutations. Crossover is produced when two genetic information are merged together to form a new one. Mutations are changes in the genetic code.

7.1 Crossover

When two animats of opposite sex get close enough they "mate". That means, the female receives male's genetic information and a cross-over takes place.

The following production rule defines the cross-over function.

crossover(gene,gene1,gene2) :
→ gene(StructureGene=merge(gene1.StructureGene,
gene2.StructureGene),
ConstructGene=merge(gene1.ConstructGene,
gene2.ConstructGene))

The "merge" function takes two lists as parameters and randomly merges them. For example, two lists:

StructureGene=
[sen1($Type = sex, X = 5, Y = 0, Shape = 0.35, Dist_{Max} = 100$),
sen2($Type = sex, X = 5, Y = 0, Shape = 0.35, Dist_{Max} = 100$),
n($State = Embryo, Hormones = [Sex]$,
$Connections = [[[sex, sen], [step, eff]]]$),
eff1($X = -10, Y = 5, \Delta x = 1, \Delta y = 0$)
eff2($X = -10, Y = -5, \Delta x = 1, \Delta y = 0$)]
StructureGene=
[sen3($Type = food, X = 10, Y = 10, Shape = 0.5, Dist_{Max} = 100$),
sen4($Type = food, X = 10, Y = -10, Shape = 0.5, Dist_{Max} = 100$),
n($State = Embryo, Hormones = [Food]$,
$Connections = [[[food, sen], [step, eff]]]$),
eff1($X = -10, Y = 10, \Delta x = 1, \Delta y = 0$)
eff2($X = -10, Y = -10, \Delta x = 1, \Delta y = 0$)]

may be merged as follows:

StructureGene=
[sen1($Type = sex, X = 5, Y = 0, Shape = 0.35, Dist_{Max} = 100$),
sen2($Type = sex, X = 5, Y = 0, Shape = 0.35, Dist_{Max} = 100$),
sen3($Type = food, X = 10, Y = 10, Shape = 0.5, Dist_{Max} = 100$),
sen4($Type = food, X = 10, Y = -10, Shape = 0.5, Dist_{Max} = 100$),
n($State = Embryo, Hormones = [sex, food]$,
$Connections = [[[sex, sen], [step, eff]], [[food, sen], [step, eff]]]$),
eff1($X = -10, Y = 10, \Delta x = 1, \Delta y = 0$)
eff2($X = -10, Y = -10, \Delta x = 1, \Delta y = 0$)]

In order to reduce the necessary evolution iterations some "heuristics" have been included in the "merge" function.

7.2 Mutations

After the crossover has taken place, some mutations are introduced for the new created gene. This is done by the following production rule.

mutation(gene) → gene(StructureGene=randomize(StructureGene),
ConstructGene=randomize(ConstructGene))

The "randomize" function takes a list as parameter and makes random changes for it.

8 Discussion

Our research work should not viewed as a simulation of real world robots. It is merely a computer exploration of behavior creation based on the development and the evolution process of neural structures. The realization of neural development or evolution on hardware is very difficult and our view is that the first we should be able to create them on computer in abstract form, and after that we can enter into the realization problems.

The transcription from genotype to phenotype, *i.e.* the development process, is a very important part of the research. As it has widely recognized, in biological systems the transcription process is a non-linear process. This means that all phenotype features are not explicitly defined in genotype, but they are a sum of different parts of genotype.

Another important aspect of the development process is the adaptation during it. As Piaget [15] has argued, the development process is greatly affected by the environment. We have not been able to model this so far. Our model gives means for modeling the adaptation process. Thus some environmental factors can give in addition to the learning schema individual characteristics to the modeled organisms. The demonstrated connection growth is an important feature in recreating broken connections between neurons. Of course in a simple neural network not so much variation can be created while persisting the original functionality.

The strongest advantages of the presented system is, however, the capability to create new behaviors without explicit definition. Usually a biological environment is viewed as a well designed system, where each animal has its own well defined task. Still the environment is an adaptive system to be able to adjust itself to changes without any explicit modifications. This is something what we are after: *self-modification for environmental changes*. The need for changes can be divided into fast or slow changes. The fast changes are implemented through a learning mechanism, the slow changes through mutations, and between them are the development process and natural and sexual selection.

9 Conclusion

In the paper we showed some results of how simple production rules can emerge sensor-motoric behaviors. All behavior was not explicitly defined, but instead arose out of the network and its structure. This structure is in order to change through the evolution process giving birth to a richer repertoire of behaviors. Due to lack of computational power the current implementation of the production rule interpreter is not able to generate a population large enough to demonstrate this explicitly.

At the current level of research we have shown

- The global behavior of an autonomous system can emerge from a simple set of production rules.
- The modification for the production rules enables the evolution process to develop autonomous system without explicit definitions.

The research has concentrated this far on stimulus based "intelligent" actions. For a higher intelligence recurrent networks with a learning mechanism are necessary. Biological learning rules described by production rules has been shown in [7]. Ongoing research combines these rules together with behavior simulation creating a little bit "more" intelligent behavior.

An open question remains how the evolution process can be directed toward some desired goal as it is required in engineering applications.

References

[1] R. Brooks, "Artificial life and real robots," in Varela and Bourgine [5], pp. 3–10.

[2] P. Kugler, "Informational field and intentional action." The International Summer School of Comparative Approaches on Cognitive Science (CACS'92), July 6-17 1992.

[3] C. G. Langton, ed., *Artificial Life*. Addison-Wesley Publishing Company, 1989.

[4] C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, eds., *Artificial Life II*. Addison-Wesley Publishing Company, 1992.

[5] F. J. Varela and P. Bourgine, eds., *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life (1991)*, A Bradford Book, MIT Press, 1992.

[6] J.-A. Meyer and S. W. Wilson, eds., *Proceedings of the First International Conference on Simulation of Adaptive Behavior: From animals to animats*, The MIT Press, Paris, September 24-28 1991.

[7] J. Vaario and S. Ohsuga, "An emergent construction of adaptive neural architectures," *Heuristics - The Journal of Knowledge Engineering*, vol. 5, no. 2, 1992.

[8] P. Borgine and F. J. Varela, "Towards a practice of autonomous systems," in Varela and Bourgine [5], pp. xi–xvii.

[9] P. Cariani, "Emergence and artificial life," in Langton *et al.* [4], pp. 775–798.

[10] P. Cariani, "Some epistemological implications of devices which construct their own sensors and effectors," in Varela and Bourgine [5], pp. 484–493.

[11] S. Forrest, *Emergent Computation*. MIT/North Holland, 1991.

[12] A. Lindenmayer and P. Prusinkiewicz, "Developmental models of multicellular organism: A computer graphics perspective," in Langton [3], pp. 221–250.

[13] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants.* Springer-Verlag, 1990.

[14] E. Schrödinger, *What is life?* Cambridge University Press, 1944. Combined reprint with Mind and Matter 1989.

[15] J. Piaget, *Adaptation and Intelligence — Organic Selection and Phenocopy.* University of Chicago Press, 1980. The original French edition *Adaptation vitale et psychologie de l'intelligence : Sélection organique et phénocopie* in 1974.

[16] C. Darwin, *The Origin of Species by means of natural selection.* The Modern Library, 1859.