ASL/NSL: A Multi-level Approach to Neural-based Architectures[1]

Alfredo Weitzenfeld
Departmento Académico de Computación
Instituto Tecnológico Autónomo de México (ITAM)
Río Hondo #1, San Angel Tizapán, CP 01000
México DF, MEXICO
email: alfredo@lamport.rhon.itam.mx
tel: (525) 6284060
fax: (525) 6162211

Keywords: Modules, Neural Networks, Hierarchical, Simulation, Architecture

Abstract

Neural-based systems are quite common in solving different technological applications. For example, in autonomous robot agents, agents vary in their sophistication, from those eliciting simple behaviors to those trying to imitate nature as close as possible both in their simulated behavior as well as in their neural structure. As the level of complexity increases, the corresponding computational models become more sophisticated involving multi-level approaches to enable top-down and bottom-up designs. The ASL/NSL architecture was designed with such purpose in mind. At the highest level, animal-like behaviors such as prey acquisition and predator avoidance are decomposed into lower level ones such as moving forwards or orienting. At the structural level, depending on available data, behaviors are mapped into specialized neural modules or if unavailable, they are implemented through other AI techniques. In this paper we describe the basic ASL/NSL computational model enabling the integration of neural network implementations as part of more complex AI systems.

[1] We thank the NSF-CONACyT collaboration grant (#IRI-9522999 in the US and #546500-5-C018-A in Mexico), the CONACyT REDII grant in Mexico, as well as the "Asociación Mexicana de Cultura, A.C.".

Introduction

In order to integrate neural-based subsystems in complex AI systems, we have developed a multi-level software development approach separating between an application's emerging behavior and its corresponding underlying structure. Our approach, known as the ASL/NSL computational model, is based at the behavior level on ASL (*Abstract Schema Language*) [11] - the term *schema* represents a behavior module or agent [2] and at the structure level, it becomes implemented (not exclusively) with neural networks on NSL (Neural Simulation Language) [12]. The ASL/NSL computational model has been used in many applications domains such as vision, visuomotor coordination, motor control (see [13]). In particular it has been used to model animal behaviors such as the praying mantis search for a habitat [4] and the frog's learning to detour behavior [7]. In general, most of the related modeling work as either concentrated at the higher-level behavior level or at the lower structure level due to the involved complexity and processing requirements. The present work tries to fill this gap by integrating complex neural underlying structures [3] into full sets of behaviors in a distributed manner.

Behavior Modules (*Schemas*)

The basic ASL/NSL computational model defines a tree-like module hierarchy as shown in Figure 1. Starting by the root module (known as the model), modules are further decomposed into additional *submodules*, having no limit on how many levels this may reach. At the same abstraction level, modules are interconnected (solid arrows), while at different levels, modules have their task delegated (dashed arrows). Networks of submodules – *module assemblages* – are seen in their entirety in terms of a single higher-level module and may be implemented independently from each other in both top-down and bottom-up fashion, an important benefit of modular design. At the higher abstraction levels, the detailed module implementation is left unspecified, only specifying the module's interface and what is to be achieved. At the lowest level, behavior is implemented in the form of structure modules, such as neural networks.

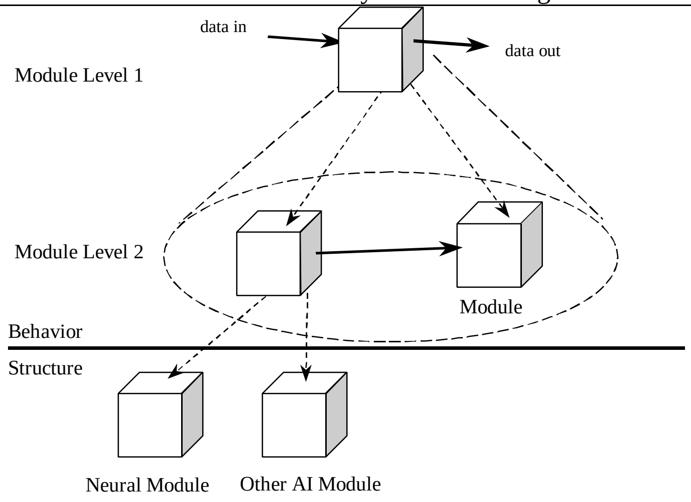

Figure 1. The ASL/NSL computational model is based on hierarchical interconnected modules. A behavior module at a higher level (level 1) is decomposed (dashed lines) into additional interconnected (solid arrow) behavior submodules (level 2). At the lowest level, structure modules are implemented by neural networks or other AI techniques.

As a computational unit, every module incorporates its own local structure and control mechanisms. Every module defines an external interface made of a set of unidirectional input and output *ports* supporting data passing between modules, as shown in Figure 2, together with a set of public methods that can be externally invoked from other modules.

Communication between modules is in the form of asynchronous message passing for both data and methods. Internally, communication is hierarchically managed through anonymous data port reading and writing. Externally, communication is managed through dynamic port *connections* (solid arrows) - links between output ports in one module to input ports in another module - and *relabelings* (dashed arrows) - module ports at one level in the hierarchy are linked to similar input or output ports at a different level.

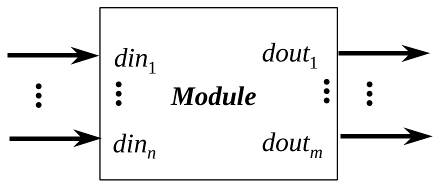

Figure 2. Each module may contain multiple input, $din_1,...,din_n$, and output, $dout_1,...,dout_m$, ports for unidirectional communication.

The hierarchical port management approach enables the development of distributed architectures where modules may be designed and implemented independently and without prior knowledge of the complete model or their final execution environment, encouraging component reusability.

For example, let's consider the *depth perception* problem where a three dimensional scene is presented to the two eyes. In general a point in space results in a left eye projection that differs from that in the right eye where this difference determines the point's actual depth. A whole ray of points results in a single point projection on each retina corresponding to different depths in space, but points on the two retinas determine a single depth point in space, the intersection of the corresponding projectors. The depth perception model developed by House [9] uses two systems to build a depth map, one driven by *disparity* cues - difference in retina projection - while the other is driven by *accommodation* cues - receiving information about focal length. The accommodation driven-field - left to its own devices - sharpens up the information to yield depth estimates. The disparity driven-field receives difference in retina projection to suppress ghost targets. The corresponding ASL/NSL model consists of a *Stereo s* root module and three interconnected submodules: *Retina r, Depth m* (accommodation) and *s* (disparity), as shown in Figure 3.

The following Java-like ASL/NSL programming language structure stores the internal program representation for the *Stereo* module definition:

```
nslModule Stereo (int sizeX,
   int sizeY, int sizeR)
{
   private Retina
      r(sizeX,sizeY,sizeR);
   private Depth m(sizeX,sizeY);
   private Depth s(sizeX,sizeY);
   public NslDinFloat2
      in(sizeX,sizeY);
   public NslDoutFloat2
      out(sizeX,sizeY);
   public void makeConn () {
      nslConnect(r.a,m.s);
      nslConnect(r.d,s.s);
      nslConnect(m.mf,s.tf);
      nslConnect(s.mf,m.tf);
      nslRelabel(in,r.in);
      nslRelabel(m.mf,out);
   }
   public void simRun () { }
}
```

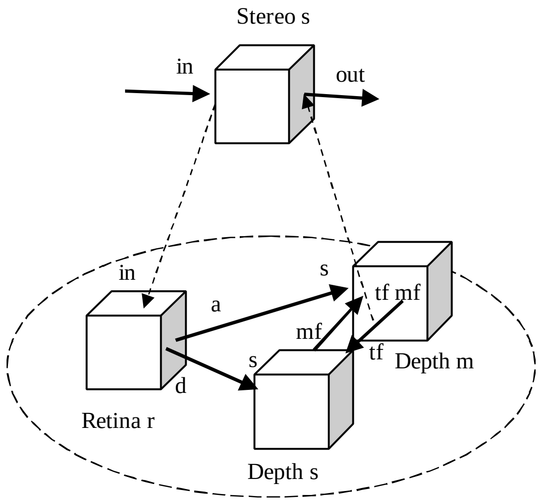

Figure 3. The *Stereo* module contains an *in* input port and an *out* output port. It is further decomposed into a *Retina* module containing an input port *in* and two output ports, *d* and *a*, for disparity and accommodation, respectively. The *Depth* module consists of an input port *s*, receiving data from the *Retina*, a second input port *tf*, receiving input from the other *Depth* module, and an output port *mf*.

In the above program definition, a *Stereo* module is defined having three instantiation parameters: *sizeX*, *sizeY* and *sizeR*. The next two lines specify the *Retina* and two *Depth* submodule *private* instantiations. The external module interface consists of a *public* input port, *in*, receiving 3D visual input and a *public* output port, *out*, communicating the *Stereo* processing results. We then define a *makeConn* method where iconnections and relabels are made between the three submodule ports in correspondence to Figure 3. Finally the *simRun* processing method is left empty since the actual behavior is implemented by the three neural submodules.

Structure Modules (Neural Networks)

The basic ASL/NSL module is extended to implement structural modules as well. In particular, we emphasize *neural modules* - neural network implementations consisting of a set of interconnected neurons, as shown in Figure 4. In our previous depth perception example each *Depth* module becomes implemented by a neural accommodation or disparity depth map, respectively. The *Depth* neural network is described by the following set of equations, where *m* corresponds to the excitatory field, *u* is the inhibitory field and *s* receives input from the retina:

$$t_m \frac{\partial m_{ij}}{\partial t} = -m_{ij} + w_m * f(m_{ij}) + w_t * f(t_{ij}) - w_u * g(u_j) - h_m + s_{ij}\,,\; f(m_{ij}) = sigma(m_{ij})$$

$$t_u \frac{\partial u_j}{\partial t} = -u_j + \sum_i f(m_{ij}) - h_u, \; g(u_j) = ramp(u_j)$$

The two depth systems are intercoupled so that a point in the accommodation field excites the corresponding point in the disparity field, and viceversa. This intercoupling among the two *Depth* modules is shown in Figure 5.

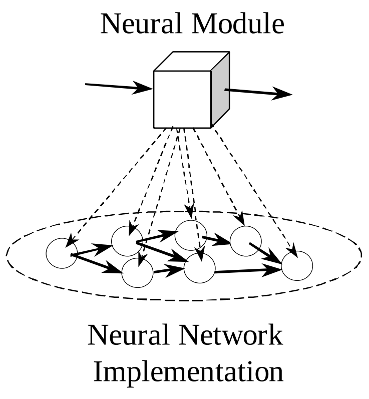

Figure 4. Every neural module is implemented by a neural network. Although neurons could be treated themselves as modules for further refinement, we treat them as separate entities, thus drawing them as spheres instead of cubes.

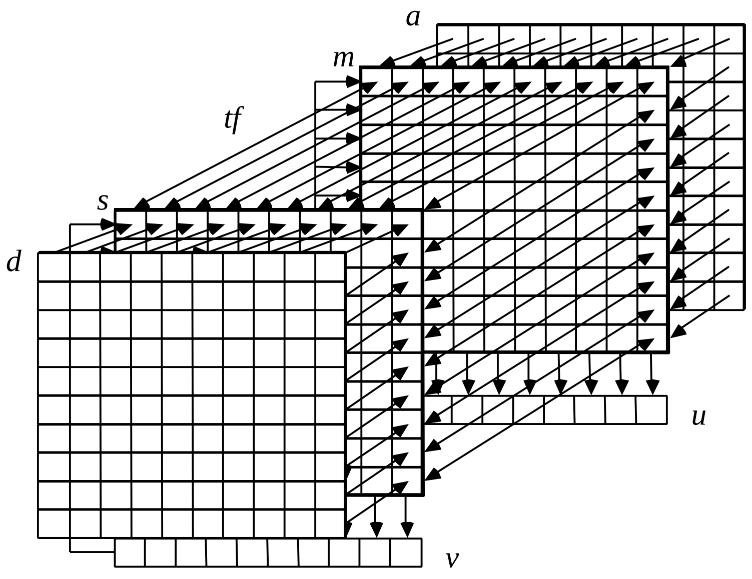

Figure 5. The two interconnected *Depth* modules can be seen at the neural level at two interconnected neural networks, the *d*-*s*-*v* disparity network and the *a*-*m*-*u* accommodation network.

The *Depth* neural module is programmed as follows (note the similarity with the *Stereo* module template):

```
nslModule Depth (int sizeX, int sizeY)
{
   public NslDinFloat2 s(sizeX,sizeY);
   public NslDinFloat2 tf(sizeX,sizeY);
   public NslDoutFloat2 mf(sizeX,sizeY);
   private NslFloat2 mp(sizeX,sizeY);
   private NslFloat1 up(sizeY);
   private NslFloat1 uf(sizeY);
   public void simRun () {
      mp = nslDiff(mp,-mp+wm@mf+wt@tf-wu*nslExpandRows( uf,mp.getRows())-hm+s);
      mf = nslSigmoid(mp);
      up = nslDiff(up,-up+nslReduceRows(mf)-hu);
      uf = nslRamp(up);
   }
}
```

While we have omitted part of the declarations in the above definition, we want to highlight the *simRun* method implementing the neural dynamics. Since we use the *leaky integrator* neuron model [1], a neuron *m* is described by two values, its membrane potential *mp* depending on the neuron's previous history and external input, and the firing rate *mf* depending only on its membrane potential and computed according to some *threshold* function.

A More Complex Example

In Figure 6 we show a simplified diagram of a more complex model involving a full set of behaviors implemented at the structure level by a number of biologically inspired neural networks. The modeled behaviors correspond to the toad's prey acquisitions and prey avoidance model [6]. The highest level, schema level 1, describes the different behaviors being modeled, *prey approach* and *predator avoid*, together with perceptual and motor schemas. In this example, we include a visual input and four types of motor action: forward, orient, snap and duck. Some of the schemas at this level are delegated to the next level down, schema level 2, where schemas perform more refined tasks. In this model, both prey approach and predator avoid, delegate their tasks to a schema assemblage composed of a prey/predator recognizer, a prey/predator selector, depth and heading, translators and maps. Next level down, the different neural networks implement the neural tasks through neural processing, known as neural schemas. In particular neural schemas in this model correspond to *Retina* [10], *Tectum* [5], *Maximum Selector* [8], and *Cue Interaction*) [9] (*Depth* only modules from the depth perception neural model).

Discussion

The work presented here has been motivated by ethological, physiological and anatomical studies of vertebrate animals. This work has resulted in multiple specialized neural models serving as the underlying structural implementations in higher level behaviors. We are currently experimenting with distributed simulation of such models [14] as well as its application in robotics [4]. Since modules have been developed independently from each other actual input and output dynamics may not be easily matched among them. Current challenges to solve these problems involve the incorporation of different simulation frequencies in different modules to match varying temporal dynamics as well as matching the type of data being transmitted among modules.

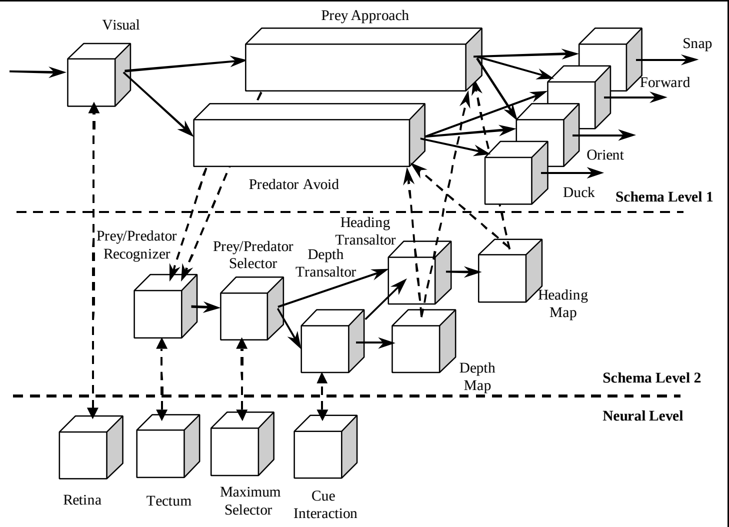

Figure 6. Module hierarchy for prey acquisition and predator avoidance behaviors. The top two levels correspond to behavioral levels (schema levels 1 and 2) and the lower level corresponds to structural neural level.

References

[1] Arbib, M.A., *The Metaphorical Brain 2: Neural Networks and Beyond*, pp. 124-126. Wiley Interscience, 1989.

[2] Arbib, M.A., Schema Theory, in the *Encyclopedia of Artificial Intelligence*, 2nd Edition, Editor Stuart Shapiro, 2:1427-1443, Wiley, 1992.

[3] Arbib, M.A., Erdi, P. and Szentagothai, J., *Neural Organization: Structure, Function and Dynamics*, MIT Press, 1998.

[4] Arkin, R.C., Ali, K., Weitzenfeld, A., Cervates-Perez, F., "Behavioral Models of the Praying Mantis as a Basis for Robotic Behavior", en *Journal of Robotics and Autonomous Systems*, 2000 (to be published).

[5] Cervantes-Perez, F., Lara, R., and Arbib, M.A., A neural model of interactions subserving prey-predator discrimination and size preference in anuran amphibia, Journal of Theoretical Biology, 113, 117-152, 1985.

[6] Cobas, A., and Arbib, M.A., Prey-catching and Predator-avoidance in Frog and Toad: Defining the Schemas, J. Theor. Biol 157, 271-304, 1992.

[7] Corbacho, F., and Arbib M. Learning to Detour, *Adaptive Behavior*, Volume 3, Number 4, pp 419-468, 1995.

[8] Didday, R.L., A model of visuomotor mechanisms in the frog optic tectum, *Math. Biosci.* 30:169-180, 1976.

[9] House, D., Depth perception in frogs and toads: A study of in neural computing, *in Lecture notes in Biomathematics*, 80, Springer-Verlag, 1989.

[10] Teeters, J.L., and Arbib, M.A., A model of the anuran retina relating interneurons to ganglion cell responses, *Biological Cybernetics*, 64, 197-207, 1991.

[11] Weitzenfeld, A., ASL: Hierarchy, Composition, Heterogeneity, and Multi-Granularity in Concurrent Object-Oriented Programming, *Proceedings of the Workshop on Neural Architectures and Distributed AI: From Schema Assemblages to Neural Networks*, USC, October 19-20, 1993.

[12] Weitzenfeld, A., Arbib, M.A., NSL, Neural Simulation Language, in *Neural Networks Simulation Environments*, Editor J. Skrzypek, Kluwer, 1994.

[13] Weitzenfeld, A., Arbib, M., Alexander, A., *NSL - Neural Simulation Language: System and Applications*, MIT Press, 2000 (to be published).

[14] Weitzenfeld, A., Peguero, O., Gutierrez, S., NSL/ASL: Distributed Simulation of Modular Neural Networks, in *Proc of MICAI 2000*, Acapulco, Mexico (to be published).