A Self-Organizing Autonomous Prediction System for Controlling Mobile Robots

Josh R. de Leeuw[1] and Kenneth R. Livingston[2]
[1]*Interdisciplinary Robotics Research Lab, Vassar College, Poughkeepsie, NY*
[2]*Department of Psychology and Program in Cognitive Science, Vassar College, Poughkeepsie, NY*
[1]*josh.deleeuw@gmail.com* [2]*livingst@vassar.edu*

Abstract

No single innovation is likely to precipitate the sudden emergence of a truly general and robust artificial intelligence. Instead, a careful study of the history of efforts to build AI systems and consideration of factors responsible for evolved general intelligence suggest point toward several promising lines of research. In the research reported here we explore the hypothesis that the ability to learn autonomously from patterns of success and failure at making predictions about what will happen next in a perception-action loop may be a necessary condition for robust AI, even if it is not sufficient. We present a novel learning architecture, the Self-Organizing Autonomous Prediction System, which learns to predict the consequences of actions within a perception-action loop. Two simulation studies demonstrate the success of the approach and the value of prediction-based learning as an important feature of highly adaptive intelligence. Future elaborations of this approach are also discussed.

1. Introduction

A growing number of researchers and theoreticians are of the view that efforts to build artificial intelligence have become too exclusively focused on narrow, domain-specific capabilities at the expense of the goal of constructing systems with the ability to be smart in a broad range of contexts and across a wide spectrum of problems [1, 2, 3]. The successes of the more modular approach to building AI are many and important (see [4]), but there is renewed interest in returning to the goal, established in the early days of AI research, to build general purpose intelligent systems. Proposals for how to accomplish that goal are many and varied. Connell & Livingston [5] have offered a taxonomy of the various proposals on offer for how to approach the robust intelligence challenge and suggest that they fall into four broad categories (with several subcategories): (1) Adding a missing mechanism, (2) introducing new core organizing processes, (3) faithfully copying properties of known intellects, and (4) adding more of, or different versions of, existing technologies in the expectation of an emergent explosion of intelligence. In a more prescriptive effort Joscha Bach [6] reviewed what he sees as the traps into which we have fallen in the effort to build artificial general intelligence and offers a set of principles that would help to avoid those missteps. Central to his prescription is the advice to build autonomous systems while making sure that representational systems are grounded. The points of overlap between these perspectives provide an interesting map of highly promising directions for work toward robust, autonomous intelligent systems.

1.1. Definitions and some desiderata for first-order robust intelligence

It is important to acknowledge at the outset that no single experiment or project can be sufficient to establish the prerequisites for robust, general purpose learning procedure (see Bach's principle number 3). Our goal in this research was to try to identify features of robustly intelligent systems that need to be developed in the long run and then to build simple versions of these features as proof of concept. In order to proceed we need a clear definition of intelligence, which we take to be the capacity for goal-directed, adaptive behavior by means of structures specialized for this purpose (a requirement that excludes plants and the tendency to over-generalize the concept of adaptation). Given this definition one can then begin to identify the desiderata for the construction of an artificial intelligence.

(1) By definition an intelligent system has goals, however simple. A system's most basic goals are those associated with maintaining system integrity (e.g., damage avoidance, attainment of resources to maintain existence, etc.). This implies the existence of evaluation functions for assessing the degree to which goals are being achieved.

(2) Goal attainment requires action, and effective action requires sensory/perceptual access to information about the world in which that action will occur. This requires an agent in an ongoing *perception-action loop* with its environment. This is not to say that the agent must be a robot wandering around in the real world, only

that it must be situated (see [5] on this distinction). A virtual intelligence operating on a distributed network could be a perception-action (PA) system where perception is of the data available on the network and action involves manipulation of those data.

(3) The capacity to adapt implies the capability to modify internal operations of the system to accommodate changes in the environment (or in the self) that could not be predicted by the processes that built (evolved) the system. In other words, an intelligent system must be capable of learning from its encounters with its environment without prior knowledge of that environment and without the constant necessity for programmer (teacher) intervention. Robust intelligence thus requires not only the capacity to learn, but also the ability to learn, at least some of the time, autonomously.

(4) Even simple PA systems operating in simple environments can result in huge sets of data on which learning procedures must operate. General intelligence requires finding goal-relevant patterns so as to compress these data into computationally tractable form [2]. This means that the learning procedures must result in *abstraction*. Ideally this abstraction process would be a procedure sufficiently general to operate across many types of inputs, including previously developed *abstract representations*. In other words, the most powerful learning systems are capable of recursive abstraction.

(5) Goals are time-sensitive, or even time-limited, so *faster learning is better* than slower learning, sometimes even at the cost of precision of output or high resolution in sensory inputs [7].

(6) General intelligence is *predictive and not just reactive* [8]. The best solution for most problems is to avoid them in the first place, suggesting that the most effective learning will incorporate procedures for improving the ability to predict events in the perception action loop.

(7) General intelligence adapts to features of its own instantiation, within some limits, and not just to the world. The range of variation in human physiology is enormous, yet the basic features of the human nervous system and the learning procedures it implements are the same. Contrast this with the specificity of software for running the typical factory robot system. General intelligence should therefore be relatively robust across some variation in its physical implementation.

Taken together, these desiderata point to the development of an architecture that incorporates agent-centered goals, the ability to learn, relatively quickly, to satisfy those goals without prior knowledge of the environments in which it will operate using feedback about predictions as well as information about action consequenes, and to do so in a way that allows adaptation not just to variations in environment but also to variations in embodiment. With these goals in mind we have begun the development of the Self Organizing Autonomous Prediction System (SOAPS) as one approach to satisfying these desiderata. SOAPS is designed to learn quickly those actions that maximize goal satisfaction by evaluating the quality of its predictions about what should happen next. At present the goals of the system are simple (navigate without collision, moving as much as possible in the process), as are the environments in which it operates, but the procedures have been developed to produce learning regardless of the perception-action capabilities of the robot itself. Here we describe two recent experiments in the implementation of this architecture in simulation of two different robot bodies, each operating in two different environments.

2. The Self-Organizing Autonomous Prediction System

The Self-Organizing Predictive System (SOAPS) is a heterogeneous neural network architecture. It consists of four distinct layers: the input, reservoir, memory, and action layers. All of the layers are founded in the basic principles of distributed neural computation, yet each has its own distinct pattern of computation (Figure 1).

Information enters the system through the input layer. Here we attempt to make as few assumptions as possible about the nature of the input. There is no built in assumption about the type of information that the nodes represent, the number of input nodes, or what sort of activation function is used. Input nodes encode sensor-specific patterns that may include values derived in any number of ways so long as the relationship of transducer to environment is causally consistent. A given sensor may thus encode any number of qualities (intensity values, frequency of detected events, information relative to location of events, etc.). As long as the method of representing the input remains consistent, the input can be represented in any number of ways.

The input layer feeds directly into the reservoir layer. This layer is based on the principles of Echo State Networks [9] and Liquid State Machines [10]. Because the computations implemented by these two architectures are functionally similar, both are sometimes referred to as reservoir computing. The general function of a reservoir network is to transform an input data stream into a higher dimensional, non-linear representation of the original data. It has been demonstrated that these networks can learn to predict patterns in non-linear dynamic systems when coupled with a supervised training mechanism [11, 12]. SOAPS uses the network exclusively for its transformational power. The benefits of this transformation are two-fold: (1) Increasing the dimensionality of the input space allows learning mechanisms that require linearly separable data to find patterns that may not have been linearly separable in the original input stream [13] and (2) the variably recursive internal dynamics of the reservoir layer maintain temporal relationships in the data stream. The system is thus sensitive to patterns that emerge only over time.

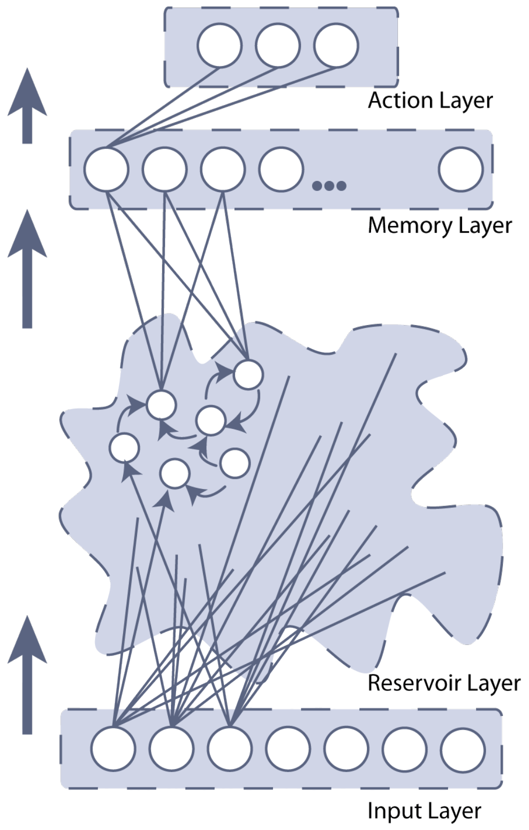

Figure 1. The basic structure of the SOAPS architecture. Activity propagates from a domain-neutral input layer to a neural reservoir. The memory layer stores predictive representations that match activity in the reservoir. The memory layer activates appropriate actions based on predicted consequence.

The implementation of the reservoir layer is as follows. The reservoir consists of *n* nodes, each of which is randomly connected to some percentage of the other nodes in the network. (Specific details on the parameters used in our experiments are given in the methods section.) The connections can be inhibitory and/or excitatory, and the degree of connectivity can vary greatly without affecting performance of the network [14]. The connections into the reservoir layer from the input are also random and sparse. The connection weights from input units into the reservoir are randomly seeded and then fixed, which ensures that the transformation function is consistent over the life of the network. The final principle that the reservoir follows is that its internal activity will decay to a null state in a finite period of time in the absence of any input data. Because of this constraint, the activity of the reservoir network will always represent a "fading memory" of the input data stream [9]. In the SOAPS model, the reservoir layer can be realized in a variety of ways. The number of nodes, connectivity patterns both within the reservoir and to the reservoir from the input layer, and the type of activation function used for the nodes within the reservoir can all vary. The tuning of reservoir networks to optimally represent input data is a topic of a research within this newly developing field (see [`4] for an overview).

The predictive function of the system is encapsulated within the memory layer. This layer consists of a set of units coding for particular patterns of activity within the reservoir. The activation within the layer is competitive such that only the unit that most closely codes for the activity in the reservoir becomes active. Each unit also contains a set of predictions. For each possible output in the action layer, a feedback value is encoded. When the winning unit becomes active, the network can then make a prediction about the feedback it will receive for taking a particular action.

The learning takes place as follows: When the network is initialized, there are no units in the memory layer. The first pattern that is presented will spawn a new memory unit, and that unit will be tuned to be sensitive to the pattern it immediately detects in the reservoir. The prediction set in the new unit will also be empty at this point. The agent will then choose an action randomly (as all actions have unknown consequences at this point), receive feedback for taking the given action, and update its prediction accordingly. At some point, the agent will encounter a situation in which the predictions it has learned are disconfirmed. The failure of a prediction triggers the spawning of a new unit. This new unit, like the initial unit, will be tuned to the pattern of activity that the agent is currently detecting in the reservoir and will have an empty prediction set. This unit-spawning mechanism is at the heart of the autonomous learning process in SOAPS. By memorizing the situations in which the system's predictions fail, SOAPS only expands its knowledge base in a goal directed fashion. This creates a set of representations (in the memory layer) which are specifically relevant for the embodiment, goals, and environment of the agent. However, because the learning *process* is very general, SOAPS is a highly adaptable approach.

Finally, the action layer is a series of nodes, each coding for a particular action. As was true for the input layer, we make as few assumptions as possible about the type of action that a particular node codes for. It could be a simple action (spin a wheel) or something more complex such as a chain of connected actions. There could be only two actions, or hundreds. However, in the current implementation of our system, adding more actions will decrease the rate at which the model learns. This is because the memory layer must learn individual predictions for each action and at present we have not implemented a system for representing patterns of similarity among actions.

3. Experiments

We used Microsoft Robotics Developer Studio (MRDS) to implement a simulation environment for our experiments. The MRDS package includes a 3D simulator with Ageia PhysX technology. This simulator allows the developer to design three-dimensional embodied simulations of robots, with a good approximation of real world physics.

3.1. Experiment One

For our first experiment, we used a simulation of the Pioneer 3DX robot that is included with MRDS. The simulated Pioneer 3DX is a two-wheel differential drive robot with a 180-degree laser range finder (LRF) and front and rear bump sensors. We constructed two environments for the simulated robot. The first environment is maximally simple, consisting of an open space surrounded by four walls. The second environment (Figure 2) adds a set of walls to the first environment, the end result of which is a maze-like environment that is comparable to a set of hallways in a building.

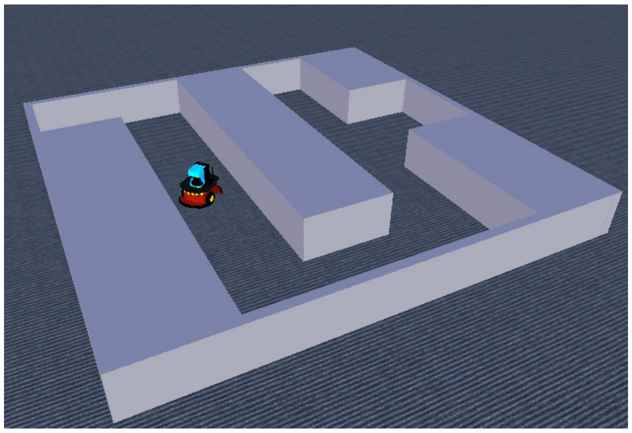

Figure 2. The simulated Pioneer 3DX robot in the maze or hallway environment in experiment one.

The robot's task was straightforward: learn to move without collisions. For this implementation of the SOAPS architecture the input layer consisted of 18 nodes. Each node represented a 10-degree segment of the laser range finder. The activity of a node corresponded to the location of the closest object (detectable up to 4 meters away) in that section of the field of view. The activation function was linear; values were scaled to the range of 0.0 to 1.0.

The reservoir layer consisted of 150 nodes. Connectivity within the reservoir was 1% with 40% inhibitory connections. Connectivity into the reservoir from the input layer was 10%. Connectivity within and to the reservoir layer was randomly generated for each new instance of the simulation. Therefore, each robot had a distinctly different network structure and thus a unique way of representing the input space at the reservoir layer.

We assigned numerical values to different outcomes so that task success and failure could be established and used to guide learning in the system. These values were used as the predictions for the memory layer. The feedback for a collision was -0.9. If the robot moved without causing a collision feedback was based on the previous action of the robot. Successful forward movement created a feedback value of -0.05; successful backwards movement -0.1; successful turning in place -0.2; and stopping -1.0. The feedback is negative so that in the case where the robot has no prediction (represented numerically as a 0), the robot would explore randomly chosen novel actions. The numbers were chosen to encourage movement, but not punish too heavily necessary turning maneuvers.

We ran ten simulated robots in each environment for a total of twenty simulations. Each simulation run lasted for 1000 iterations of the learning algorithm, one iteration every 333ms, for a total of just over 5 minutes and 30 seconds of learning (the MRDS environment simulates real time).

3.2. Experiment One Results

Each simulation run generated 1000 feedback values, one for each iteration of the learning algorithm. We grouped these data into ten time blocks with 100 values in each block. We calculated an average of all values in each block to generate 10 total data points for each robot (Figure 3). A repeated measures analysis of variance (ANOVA) on level of feedback (less negative feedback equals better performance) with environment as a between groups factor and trial block as a repeated measures factor revealed a main effect of time block on feedback $F(9,162) = 4.848$, $MSE = 0.022$, $p < 0.001$, indicating that performance improved over time. There was also a significant interaction effect of time block and environment $F(9,162) = 3.745$, $MSE = 0.017$, $p < 0.001$. From our own observation of the behavior of the robots in the open environment, it was clear that the robots are learning to avoid collisions. They learn this behavior so quickly that it is essentially undetectable in the statistical analysis. In the hallway environment, it takes the robots longer to find a reliable solution. The learning curve shows that performance levels off at around the fifth time

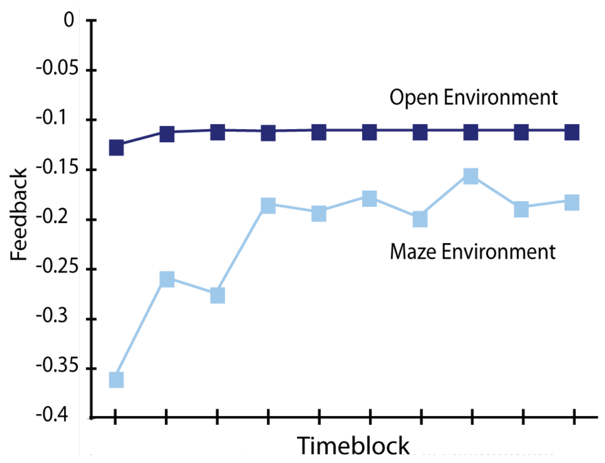

Figure 3. Average learning curves over 1000 training interactions in Experiment 1. Each timeblock represents the average of 100 feedback values. Learning is extremely rapid in the simple environment and more protracted in the maze.

block, indicating that, on average, the robot took less than 500 iterations of the algorithm to reach its peak performance. Peak performance is lower in the maze environment because successful navigation requires more turning in the maze.

3.3 Experiment Two

In our second experiment, we sought to replicate the results of the first experiment with a different robot. Using MRDS, we designed a smaller simulated robot based upon a real robot in our lab. This robot was also a two-wheel differential drive robot with front and rear bump sensors. Instead of a 180-degree LRF, we simulated three separate analog IR sensors. Each sensor had a field of view of 4 degrees and a maximum range of 80cm, comparable to a standard commercially available IR sensor. These sensors were oriented at 0, -45, and +45 degrees relative to the robot's centerline. Compared to the simulated Pioneer used in our first experiment the available sensory information was relatively impoverished.

Our experiment again consisted of using two different environments, one open and one with various walls (Figure 4). Critically for our purposes here, the parameters of network implementing SOAPS were identical to those of experiment one with the exception of the number of nodes in the input layer where there are only three units providing information instead of the 18 available to the robot in the first experiment. All other network parameters were identical to those in the first experiment. The same procedures are thus operating in a different perception-action platform.

We ran 20 simulations in each environment. Each simulation ran for 1000 iterations of the learning algorithm at the rate of one iteration every 333ms. As in experiment one, the reservoir layer for each robot is uniquely and randomly generated, allowing treatment of each robot as an individual in the data analysis.

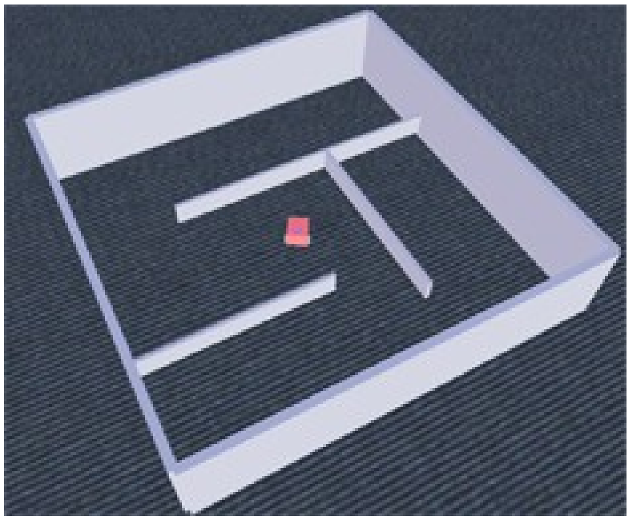

Figure 4. The simulated robot in the the maze environment in experiment two.

3.4. Experiment Two Results

We analyzed the data in the same way as our first experiment: feedback values were averaged over time block and analyzed using a repeated measures ANOVA. There was a main effect of time block, F(9,342) = 14.380, MSE = 0.128, p < 0.001. Robots in the open environment performed better than those in the maze environment, overall F(1,38) = 4.512, MSE = 0.170 p = 0.040, but both groups converged on the same level of performance by the end of the training period (Figure 5).

4. Discussion

The learning occurring in the two experiments described here is, on the surface, remarkable for how unremarkable it seems. A robot starts out moving around in an environment and colliding with the obstacles it encounters, and then it gradually gets better at avoiding them. But the results of these experiments provide ample reason to pursue the further development of this approach to building robust artificial intelligence. In particular, these experiments show progress toward several of the desiderata for general intelligence outlined in the introduction.

First, robots running the self-organizing prediction system were truly autonomous. No feedback is given to the system by an external trainer at any point in the learning process. Nor was there any prior information

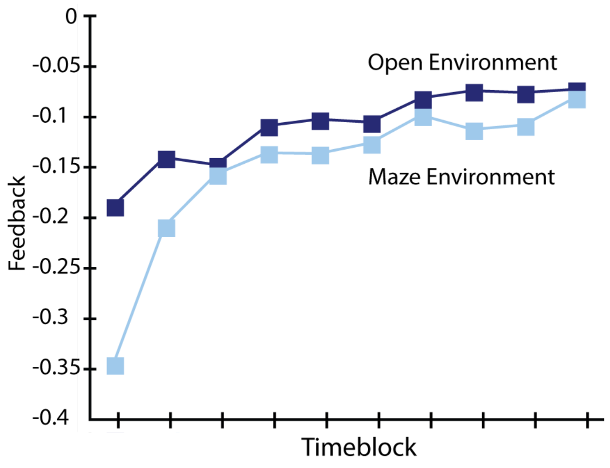

Figure 5. Average learning curves over 1000 training interactions in Experiment 2. Each timeblock represents the average of 100 feedback values.

provided about the nature or meaning of the inputs it would receive, and the environment itself was in no way pre-specified for the robot. Instead the robots are instantiated with simple goals that are closely linked to valuations placed on patterns generated during the robot's engagement in a dynamic and fluid perception action loop. Learning is remarkably fast; the system achieves a very high level of performance within minutes, even in a relatively tight hallway environment. It accomplishes this by developing the ability to predict successfully when its actions are tending toward collision; upon predicting a collision it changes its behavior to avoid that collision. It thus comes to know more than its environment. It comes to know its relationship, *in action*, to its environment, and it does so by being predictive rather than reactive.

Critically, the learning and development process of SOAPS is so rapid because of the way in which new knowledge is acquired by the system. SOAPS relies on its own predictions and their success and failure to guide its development. By using predictions as a form of feedback, SOAPS is able to learn in a goal-directed yet unsupervised manner. This is different from unsupervised clustering algorithms like those described by Adaptive Resonance Theory (ART) [15]. The assumption in ART and other unsupervised systems is that clusters can be formed by attending to patterns of similarity in the space of sensory inputs. While recognizing patterns of similarity in sensory inputs is critical for generalizing learned responses, such patterns are not intrinsically connected to any particular goal of the agent. An agent can always find a new way to partition the sensory space, and without a goal-directed learning process the agent cannot *autonomously* determine which way to partition its inputs. SOAPS solves this problem by building a goal oriented feedback mechanism into the development process.

Other characteristics of SOAPS are also very promising. With almost no modification (only that which is necessary for connecting different numbers or types of sensors) it produces learning in two different robot designs. The patterns of learning were similar, but varied in expected ways. When SOAPS operated in trivial environments with rich sensory inputs, it learned almost instantaneously. In more complex environments and in cases where sensory information was relatively impoverished, learning was slower as SOAPS had to deal with a greater level of ambiguity.

Furthermore, it shares with evolved intelligent systems the fact that the details of its implementation are variable and thus unique for each individual. This variability provides a necessary ingredient for the operation of selection processes and thus evolutionary change, which makes it an excellent platform for combining the study of ontogenetic and phylogenetic processes operating across generations. We are currently taking advantage of this feature of the system to explore the use of genetic algorithms to optimize the structure of the architecture.

SOAPS compares favorably with alternative approaches to solving the problem of learning, especially with respect to its autonomy and generality. Reinforcement learning procedures, like Q-learning, for example, require prior knowledge of the state space and then seek to optimize a response given the current state of the system. SOAPS ignores optimization and simply relies on segmenting internal patterns generated by the input stream in accordance with the success and failure of its predictions about how those patterns should unfold. However, because SOAPS segments the perceptual state space in a goal directed manner, it learns very rapidly in spite of ignoring any sort of optimization. It is thus truly self-organizing.

4.1. Next steps

In spite of the considerable success of SOAPS in these simulations there is a sizable gap between the intelligence demonstrated in the studies reported here and truly general and robust intelligence. Several extensions of this work are already under way in an effort to further develop the system's capabilities.

First, we have begun to take advantage of the natural variability in system structure by trying to evolve various parameters over generations (e.g., number of reservoir units, density and level of recursion of connections, etc.).

Early indications are that we can improve system design in this way. Second, we are in the process of loading the system architecture onto real robots operating in real world environments in order to establish that the successes demonstrated are not limited to simulated environments, which are by their nature free of many sources of noise on both the perception and action sides of the PA loop. The ability to find patterns while ignoring such noise would constitute further evidence of the power of this approach. Finally, we will shortly begin experiments to establish how well the ability to predict in one environment generalizes to a new setting. This is one of the most basic indications of non-rigid, generalized intelligence and evidence for such generalization is an important next step.

Beyond these developments there lie several features of the architecture that invite further exploration and development. For example, it is not possible at present to modify the groupings that SOAPS forms at the memory level, once they are established. The system relies on making more groupings to capture new patterns rather than on refining groups it already has. There is abundant evidence for the importance for general intelligence of the ability to change conceptual structures. Indeed, one might take that to be one of the hallmarks of a robust, adaptive intelligence. Certainly it would make for more efficient intelligence if one could revise the network in this way.

In the longer term it will be necessary to discover how the same system can master multiple predictions for multiple goals operating simultaneously, and to do so flexibly in the face of shifting goal priorities. This will include examining both new types of goals, such as predicting complex sensory patterns, and new types of input, including the output of other pattern finders. We believe that this architecture is extremely promising in this regard.

5. Acknowledgments

We would like to thank Jackie Kory for her assistance in running simulations and collecting data, John Long for comments on a draft of this paper, the Undergraduate Research Summer Institute at Vassar College for funding potions of this work. The research also benefits from the funding provided by NSF grant BCS-0320764.

6. References

[1] D. Friedlander, and S. Franklin, "LIDA and a Theory of Mind", In P. Wang, B. Goertzel, and S. Franklin (Eds.) *Artificial General Intelligence 2008,* IOS Press, Amsterdam, 2008, pp. 137-148.

[2] M. Looks, B. Goertzel, and C. Pannachin. (2004). "Novamente: An integrative architecture for general intelligence", *Proceedings of AAAI Symposium on Achieving Human-Level Intelligence Through Integrated Systems and Research,* Washington, D.C., August 2004.

[3] M. Pickett, D. Miner, and T. Oates. "Essential phenomena of general intelligence", In P. Wang, B. Goertzel, and S. Franklin (Eds.) *Artificial General Intelligence 2008,* IOS Press, Amsterdam, 2008, pp. 268-274.

[4] T. Menzies. 2003. "21st Century AI: Proud Not Smug", *IEEE Intelligent Systems*, May/June 2003, pp. 18-24.

[5] J. Connell, and K. Livingston, "Four Paths to AI", In P. Wang, B. Goertzel, and S. Franklin (Eds.) *Artificial Gen. Intelligence 2008,* IOS Press, Amsterdam, 2008, pp. 394-398.

[6] J. Bach, "Seven principles of synthetic intelligence", In P. Wang, B. Goertzel, and S. Franklin (Eds.) *Artificial General Intelligence 2008,* IOS Press, Amsterdam, 2008 pp. 63-74.

[7] J. de Leeuw, and K. Livingston, "When less is more: Sensor resolution and learning", In L. Berthouze, C. G. Prince, M. Littman, H. Kozima, and C. Balkenius (Eds.) *Proceedings of the Seventh International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems.* Lund University Cognitive Studies, *135*, LUCS, Lund, 2007.

[8] Hawkins, J. *On Intelligence*. Owl books, New York, 2004.

[9] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks", GMD Report No. 148, German National Institute for Computer Science, 2001.

[10] W. Maass, T. Natschlager, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations", *Neural Computation*, *14* (11), 2531-2560, 2002.

[11] H. Jaeger, "Adaptive nonlinear system identification with echo state networks", In S. Becker, S. Thrun, and K. Obermayer (Eds.) *Advances in Neural Information Processing Systems 15*, MIT Press, Cambridge, MA, 2003, pp. 593-600.

[12] H. Jaeger, and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication", *Science, 204*, 2004, pp. 78-80.

[13] A. Morse, and M. Aktius, "Dynamic liquid association: Complex learning without implausible guidance", *Neural Networks*, In Press, 2008.

[14] M. Lukosevicius, and H. Jaeger, "Overview of reservoir recipes", Tech. Rep. No. 11, School of Engineering and Science, Jacobs University Bremen, 2007.

[15] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance", *Cognitive Science, 11*, 23-63, 1987.