

# **PART II**

## **RESEARCH PAPERS**

### **6. LEARNING: TOWARD SELF-EVOLVING ARCHITECTURES**

- 6.1 Evolution of Intelligent Systems Architectures: What Should Be Measured?  
A. Meystel, Drexel University, USA
- 6.2 Machine IQ with Stable Cybernetic Learning With and Without a Teacher  
H. Szu, Digital Media Lab, Wash DC, USA
- 6.3 Properties of Learning Knowledge Based Controllers  
E. Grant, G. Lee, NC State University, USA
- 6.4 Examining the Resource Requirements of Artificial Intelligence Architectures  
S. A. Wallace, J. E. Laird, K. J. Coulter, University of Michigan, USA
- 6.5 A Metric for Monitoring and Retaining Flight Software Performance  
A. Peterson, Computer Sciences Corp., USA
- 6.6 Flexible Robotic Assembly  
D. P. Gravel, W. S. Newman, Ford, Case Western Reserve University, USA

# Evolution of Intelligent Systems Architectures: What Should Be Measured?

A. Meystel

Drexel University, Philadelphia, PA 19104

## Abstract

Various degrees of intelligence evolve in the intelligent systems as a result of their development by the virtue of external design and self-organization. The increase in degree of intelligence is achieved via evolution of its architecture. This paper is intended to establish a conceptual and methodological background required for design and evaluation of performance and the degree of intelligence of intelligent systems. The paradoxical ability to increase redundancy while reducing complexity is described as a hallmark of intelligence. The naturally evolved architectures of intelligence are constructed in such a manner that the tools of complexity reduction do not curb the combinatorial capabilities of the system.

## 1. Intuitive Approaches to the Concept of Intelligence

An attempt is made to approach the concept of intelligence constructively and from the scratch. This analysis is motivated by the need for using the results for *constructed* (primarily, engineering) *intelligent systems* and *agents*. For the author, the Descartes' problem (of the Mind existing separately from the Body) simply doesn't exist, because the Mind of the constructed Machine is undoubtedly produced by its Body. Nevertheless, the author doesn't adhere to the technological paradigm alone. Both the examples of intelligence and its architectures will be discussed for all domains shown in Figure 1.

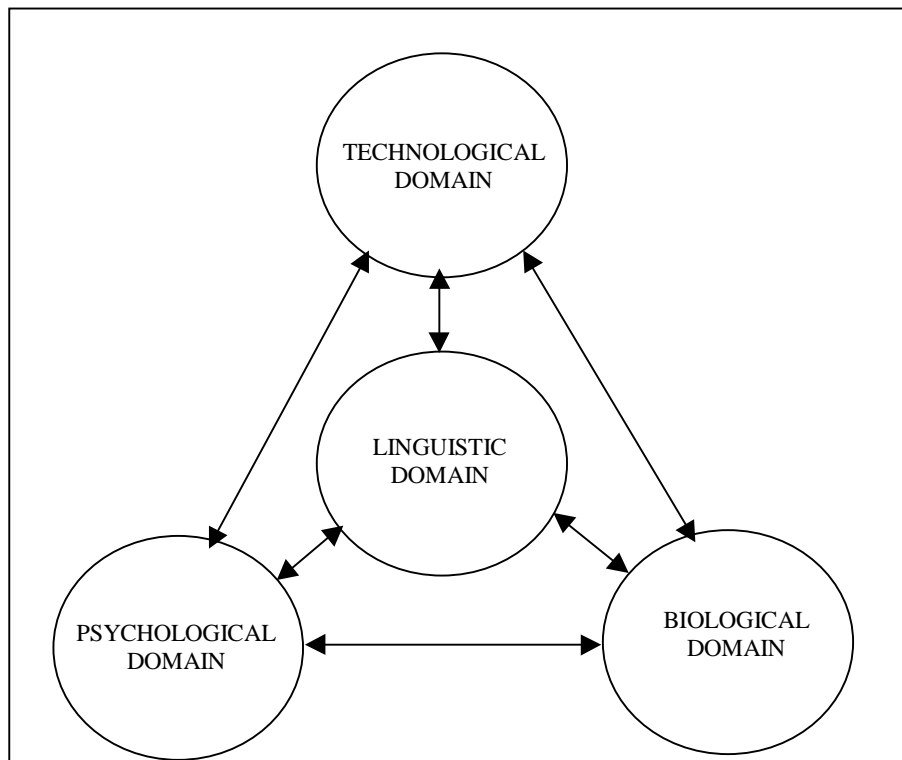


Figure 1. Techniques linked with and stemming from the concept of *intelligence*.

The goal to construct the architectures of intelligence and analyze their evolution can be achieved if a comprehensive definition of intelligence is introduced. It seems meaningful to derive the definition from integrating the phenomena characteristic for intelligence. Obviously, they can be demonstrated in relevant systems belonging to all domains shown in Figure 1. Interestingly enough, within each of these domains, there are common habits of discussing intelligence. Possibly, this is a result of the fact that all of them depend on the linguistic domain. The main habits of talking about intelligence can be listed as follows:

1. Functioning of intelligence is frequently characterized in the anthropomorphic terms of mental conduct.
2. Intelligent activities are attributed to levels of generality (levels of *scope*)

### 1.1 Features of Mental Conduct

The terms of natural language that characterize intelligence both positively and negatively, can be used for evaluating the richness of the concrete domain of discussion and judging whether domains from Figure 1 are well represented. One can make an observation that all of these properties can be quantitatively evaluated in a crisp or fuzzy manner.

**Table 1. Antonyms characterizing Intelligence (From [1])**

clever	↔	dull	observant	↔	unobservant
sensible	↔	silly	critical	↔	uncritical
careful	↔	careless	experimental	↔	unexperimental
methodical	↔	unmethodical	quick-witted	↔	slow
inventive	↔	uninventive	cunning	↔	simple
prudent	↔	rush	wise	↔	unwise
acute	↔	dense, obtuse	judicious	↔	injudicious
logical	↔	illogical	scrupulous	↔	unscrupulous
witty	↔	humorless	smart	↔	stupid

The tendency to using these adjectives for characterizing intelligent systems in all domains is unavoidable. Although, they could be called anthropomorphic, their use seems to be justified even as applied to living creatures different from humans such as apes, cats, dogs, horses, mice. Then, we might agree with using at least some of these terms to analyze intelligence of birds, fishes, reptiles. After getting used to see the common patterns we can expand some terms related to intelligence into domain of insects, and then, proceed toward bacteria, too.

Analysis of the intelligence related vocabulary helps to discover a number of other phenomena that should be taken in account in constructing definitions and models for *intelligence*. Indeed, from the fact that **stupidity  $\neq$  ignorance** we can conclude that **intelligent  $\neq$  possessing knowledge**. Thus, **having knowledge**, or **being informed** could not be considered a base for defining *intelligence*. On the other hand, *intelligence* is frequently associated with a comparably vague concept of the activity of **thinking**. The latter contains as a part, such activity as **theorizing**, and one can expect that **theory formation** should be represented in the architecture of intelligence. (It is the capacity for rigorous theory that lays the superiority of men over animals not the capacity to attain knowledge).

It would be desirable to embark on constructing the definition of intelligence focusing upon most of the factors that is linked with this complex phenomenon of mental conduct. Before introducing architectures of intelligence a set of mental conduct epithets was analyzed including such terms as:

**careful**                      **stupid**                      **logical**                      **unobservant**                      **ingenious**  
**vain**                      **methodical**                      **credulous**                      **witty**                      **self-controlled**

and their correlations so that they could be represented in the definitions and architectures..

## 1.2 Intelligence is a Property existing at all levels

In addition to multiple properties and phenomena related to the mental conduct, intelligence invokes talking about *level of intelligence*. The term intelligence is attributed to each of the interrelated levels including

- societal phenomena
  - group activities
  - individual activities
  - organ functioning
  - cell functioning
  - DNA functioning
- } scaled by the unit of *intelligent agent* (1)

Some of these levels emerged because humans introduced them. Some of them evolved naturally (biologically, ecologically, or psychologically). In all cases, the multiresolutional organization improves the efficiency of functioning [2]. Each particular level of resolution is scaled by the nature of the hierarchy (1). At the same time, for each agent within the hierarchy (1) another multiresolutional scaling can be introduced for units of interest existing within a level and requiring its own hierarchy of levels that makes operations with this unit more efficient. It seems that the ability to come up with a multiplicity of levels of resolution is a property of intelligence that produces these levels of resolution.

On the other hand, each of the levels mentioned above can be characterized by the ability to build and construct rules associating objects and activities at the level, and by the ability to introduce and use theories. Both rules and theories are formulated by the researcher observing and analyzing external intelligence. However, they reflect the properties and laws existing within the system of objects under consideration. Both rules and theories are applicable for the decision making processes that are utilized to control

- objects at a level
- levels as a whole
- the overall system that is combined out of levels and contains these objects.

Let us notice that the organization of the system to be controlled is affected by the intelligence, and the introduction of rules and theories is done by the intelligence, too. The source of the intelligent in both cases is not determined, and the intelligence as a phenomenon is undefined.

## 2. Introducing Formal Approaches

### 2.1 General Statements

It looks like the Theory of Control that does not take in account the phenomenon of intelligence, is not fully equipped for solving problems for the domain of intelligent systems, e.g. in robotics. The particular problem is in determining VECTOR OF INTELLIGENCE OF A CONTROLLER and putting it in a correspondence with the VECTOR OF PERFORMANCE. Designers are dealing with systems that are underspecified even as far as their *inputs* and *outputs* are concerned.

Thus, the first two emerging questions are: 1. What are the inputs into the system under consideration? and 2. What are the outputs of this system? The input can be introduced by the designer of the architecture and by the values of variables provided by a specific architecture (intelligence). The output is always understood in the terms of performance.

An attempt to answer the questions requires to revisiting the logical categories that are used for analysis of systems with intelligence (in particular, *intelligent control systems*). Simultaneously, we must determine whether we will discuss these issues in the terms of predicate calculus of the first order, in the terms of other logical systems, or in the terms of meaning extraction and interpretation of the Natural Language. The lists of inputs and outputs contain concepts that entail a diversity of various schemes of reasoning and

logical categories. The logical type of category, to which a concept belongs is a set of ways, in which it is consistent, i.e. it is logically legitimate to operate with. To determine a logical network of concepts is to review the logic of propositions, in which they are utilized including the following:

1. With what propositions of the classical control theory, the propositions related to intelligence are consistent and/or inconsistent
2. What are the new propositions of control theory that follow from the propositions related to intelligence

One of the challenges is to determine whether for the alternative definitions of intelligence and the associated processes we selected correctly the logical categories in terms that are consistent with the practice of design and application in the domains shown in Figure 1. In particular, we would be interested whether the concepts of thinking, mental powers, smartness, their components and operations they entail have been coordinated consistently. It should be demonstrated that there is no operations with these concepts and processes that breach logical rules. We suspect that the consistent system can be built if the logical consistency will be determined not in the terms of predicate calculus of the first order but in the terms of laws of interpretation determined for the Natural Language used for describing the real systems and situations.

## **2.2 List of Premises that Are Characteristic for Intelligent Control**

The following premises can be considered as following from the experiences in all domains of Figure 1 in the cases of exploring intelligent systems as objects and intelligence as a phenomenon of these objects.

### ***2.2.1 Cultivating redundancy is a prerequisite of intelligence***

Redundancy of systems is understood as having their resources, components, or properties in abundance, or in excess. It is a feature of intelligent systems that information they deal with is intrinsically redundant and the tools of processing this information are in excess of the minimally required set of tools. It is a feature of intelligent systems to cultivate this redundancy and it will be shown that intelligence is equipped by specific tools for doing this.

This property of redundancy is very important and very characteristic for intelligent systems. They should be always ready to withstand uncertainty, and since the survival is at stake, the property of redundancy helps to minimize the risk of failure. E. Ruspini has mentioned: the systems should have more intelligence than it needs for solving the problem<sup>1</sup>. Obviously, the same problem can be resolved with different level of intelligence. Then, the results of this problem-solving process could be used for evaluating the level of intelligence. This level might depend on the level of redundancy.

Although redundancy as a property is considered negative (it should waste resources), not only intelligent systems do not fight redundancy, it explore, use, and even cultivate the redundancy. Redundancy is the tool for combining and testing new alternatives of decisions. After evolving intelligent systems develop a mechanism of exploring things within its “virtual reality,” redundancy is becoming a tool for planning and a tool for learning without actually having physical experiences.

Autonomous systems should acquire info in physical (realistic) and/or imaginary playgrounds. The following factors are being displayed related to redundancy:

- Playfulness is a property observed in living creatures or linguistic systems that are characterized by a very high level of intelligence. Playfulness of an intelligent system is to be considered a part of the learning process.
- Redundancy supports various manifestations of the property called “desire” including all known classical desires that determine foraging and reproductive activities.

---

<sup>1</sup> In an exchange during the panel on Intelligent Control at IJCNN’2000, E. Ruspini commented that probably such creature as E.coli possesses all intelligence it needs for functioning. A. Meystel proposed a paradoxical definition for intelligence that further develops Ruspini’s statement: “The system is intelligent iff it has more intelligence than it needs.”

- Certainly, speaking about “playful ameba” might be a stretch however searching activities are observed even for amoebas [5] and E.coli’s [6] (and this allows to talk about certain degree of intelligence even in these classes of living creatures [7]).

Intelligent systems are equipped by multiple tools of acquiring and increasing their redundancy. Learning is one of the tools that employs actual experiences or imagination.

### ***2.2.2 Reduction of complexity is a working technique of intelligence***

How is it possible to cultivate redundancy, and yet fight complexity? This paradoxical ability is a hallmark of intelligence. Practically, it means that the tools of complexity reduction should not curb the combinatorial capabilities of the system. Such tool exists, and this is organization of information in a multiresolutional fashion (see [3, 4]). This organization of information actually determines appearance of the levels mentioned in sub-section 1.2.

The need to evaluate and reduce complexity was always clear in computational mathematics and this led to the concept of epsilon-entropy and techniques of its evaluation [8]. Many elegant mathematical techniques of complexity reduction has been developed (e.g. like in [9]). The specifics of application domain was appreciated (see [10] for the software complexity, [11] for syntactic complexity, [12] for complexity of information extraction, [13] for information of control system).

However, the need to use multiresolutional organization of information for complexity reduction was not immediately acknowledged and considered an understandable and desirable tool even after publication of [3, 4]. Further explanation of relations between multiresolutional tools of complexity reduction can be found in [14, 15].

In all systems (technological, biological, psychological and linguistic) formation of multiresolutional representation is a technique of complexity reduction. Even E.coli fights the complexity by forming at least two levels of resolution (high resolution – single E.coli, low resolution – swarms formed as a result of bacteria gathering in groups [7]).

### ***2.2.3 Loop of Semiotic Closure is the Primary Architecture of Intelligence***

The modules of (1) World, (2) Sensors, (3) Perception, (4) World Model, (5) Behavior Generation and (6) Actuators, connected in a loop of closure, are forming an Elementary Functioning Loop, or ELF. The module of World is the ambient environment including a source of information from the process generated by Actuation to be observed by Sensors. This component of the World also consumes the energy submitted by the module of Actuation. If one interprets Figure 2 as a general structure of an intelligent vehicle, then the module of World is the couple Vehicle/Road. The energy is conveyed through this couple to the body of the moving Vehicle, the vector of speed is measured for the Vehicle relative to the Road within this couple. Sensors are transducing the information from the domain of physical reality to the information carrier accepted by the system of computation. In addition, Sensors are responsible for complicated activities linked with organization and coordination of testing. These activities are a part of another loop of closure (see [17]).

The module of Sensory Processing organizes the information and submits it to the World Model that puts the units of acquired knowledge into a form appropriate for storing and utilization by the module of Behavior Generation. The latter may vary from the simple look-up table to the complex devices that explore alternatives of plan and simulate them before submitting them to the module of Actuation. The simple look-up table would contain the list of control functions  $f(t)$  together with previously experienced or expected measures of achievement  $J(f, x, x^*)$  for the given goals  $x^*(t)$  and present situations  $x(t)$  as couples

$$x^*(t), x(t) \rightarrow f(t), J(f, x, x^*). \quad (2)$$

The concept of semiotic closure is not an obvious one. It exceeds the straightforward idea of feedback that can be formulated as follows. In a system, there exists a monitor (human or electronic/mechanical) that compares what is happening at time  $t$ ,  $x(t)$ , with some standard of what should be happening  $x^*(t)$ . The difference or error,  $\Delta(t) = x(t) - x^*(t)$ , is fed to a controller for generating an action by a control function  $f(t)=y(t+k)$ , which can be taken only at a later time,  $t + k$ . Thus, the feedback equation presumes some

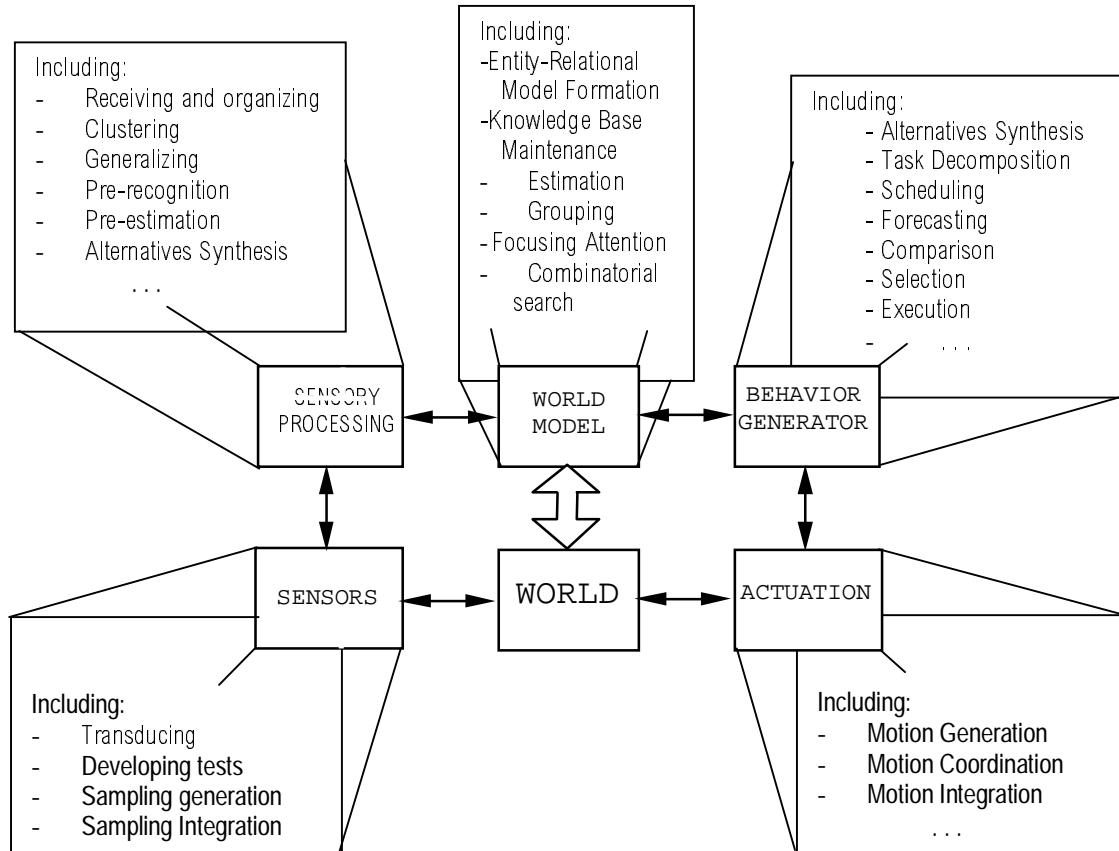


Figure 2. Semiotic Closure for a System With Motion

$$y(t + k) = f[\Delta(t)] = f[x(t) - x^*(t)] \quad (3)$$

standard assigned, some variable compared with this standard, and some device that computes “feedback compensation.” The standard might be assigned as a goal externally, or stored in the module of World Model. The device that computes “feedback compensation” can be associated with the module of Behavior Generation. Sensors, Sensory Processing and World are meant but not explicated. Certainly, this concept should be enhanced substantially to be transformed into the concept of semiotic closure.

Semiotic closure was anticipated in 1967 by L. von Bertalanfy [18] who considers feedback to be “a special case of general systems characterized by the presence of constraints which led the process in the way of **circular causality** and so making it self-regulating. This loop of “circular causality” was dubbed “semiotic closure” by H. Pattee in 1973 [19]. It was introduced to analysis of intelligent systems in [ 20] and [21]. Semiotic closure can be constructed for any domain and any system that exhibit elements of intelligence.

#### 2.2.4 Entity-relational network (ERN) is a frequent form of constructing the representation at a level of intelligent system

It would be more prudent to say that we simply do not know any alternative to ERN. Of course, we can approximate ERN by a multiplicity of tables and approximate each of the tables by an analytical function. We do this for the variety of manual activities. However, as computer permeates our workplace, we found that having ERNs even in a tabular form is the most flexible way of storing information.

Thus, a problem of generalization emerges as a problem of local substitution of large accurate tables by small tables with larger but still acceptable error. Thus, instead of a global gigantic ultimately accurate ERN, we receive a set of entity-relational networks  $\{ERN_i\}$ ,  $i=1,2,\dots, n$  where 1 is the index (number) of the level with highest resolution,  $n$  is the index of the level with the lowest resolution. The system does not have all these levels in its storage because the amount of information in  $\{ERN_i\}$  would substantially exceed the amount of information in its level of highest resolution  $ERN_1$ . The system remembers only levels with middle (average) resolution and selected traces at the level of higher and/or lower resolution. If it requires more lower resolution information, it generalizes the middle level information as necessary. If it requires higher resolution information, it instantiated (decomposes) the information top down as requested. The system  $\{ERN_i\}$  is a nested system, i. e. the conditions of inclusion should be satisfied for the ontologies constructed for the Worlds represented at each particular level of resolution. The same conditions should be realistically satisfied for the objects and actions represented at the levels. Such a system can exist if it is supported by the operators of grouping, focusing attention (selection), searching for combinations of interest (combinatorial search), and the operators that ungroup, defocus and eliminate the results of search.

### ***2.2.5 Constructing Multiresolutional Representation is a tool of intelligence***

Each level of representation has granularity that is a result of generalizing information from the lower level of higher resolution [16]. Both objects and actions of the real world have their representatives at several (at least at two) levels of resolution and therefore are multiresolutional. The mechanism of obtaining lower resolution objects and relationships out of higher resolution objects and relationships is called generalization.

The nature of generalization was envisioned by gestalt psychologists [22]. The need in the computational theory of generalization was emphasized by J. McCarthy in [23]. One of the possible algorithms of generalization is demonstrated in Figure 3. One can see in this example that the algorithm consists of operators that perform Grouping (G), Focusing Attention (FA) and Combinatorial Search (CS) together (the subscript means the level it works for). The joint set of operators G, FA, and CS we will call GFACS. Using this set: computational procedures of grouping focusing attention and combinatorial search (GFACS) is inevitable in an intelligent systems because the level of generalized information cannot be built otherwise. GFACS generalizes information bottom up. Decomposition top down requires for an algorithm of instantiation ( $GFACS^{-1}$ ). There exist a vast multiplicity of algorithms belonging to the class of GFACS: e.g. ARMA (auto-regressive moving average) as in [24, 25]; CMRA (convex multiresolutional analysis) as in [26] and other. CFACS-1 has its prototypes, too, such as Sieve Decomposition algorithm [27].

*Encoding* of stored information is done in a multiresolutional fashion too, and this leads to the further reduction of complexity because instead of storing the body of the message (the file) we can store onle the code and apply to this code the mechanism of restoring the body. It is a legitimate mechanism of storing informational entities by storing the code and regenerating (reconstructing) the information as necessary. Storing information in the form of DNA is an example of reconstructing the multiresolutional system of a living organism.

### **2.2.6 Cost-functional**

The need in a reduction of computational complexity would be easy to resolve by abandoning computation. Yet, this cannot be done because the system has a goal to fight for reducing the time and energy that are required to reach the target. This determines the conditions of the optimization process. The latter should be performed in correspondence with the calculus of variations and Euler-Lagrange equation. The central problem that emerges is to determine properly the Hamiltonian of the system, or its cost-functional.

As far as computational complexity is concerned, the results of optimization are driving the process of forming levels of resolution. The optimization for an E.coli sounds like working under the heuristically introduced cost-functional of foraging (see [6]):



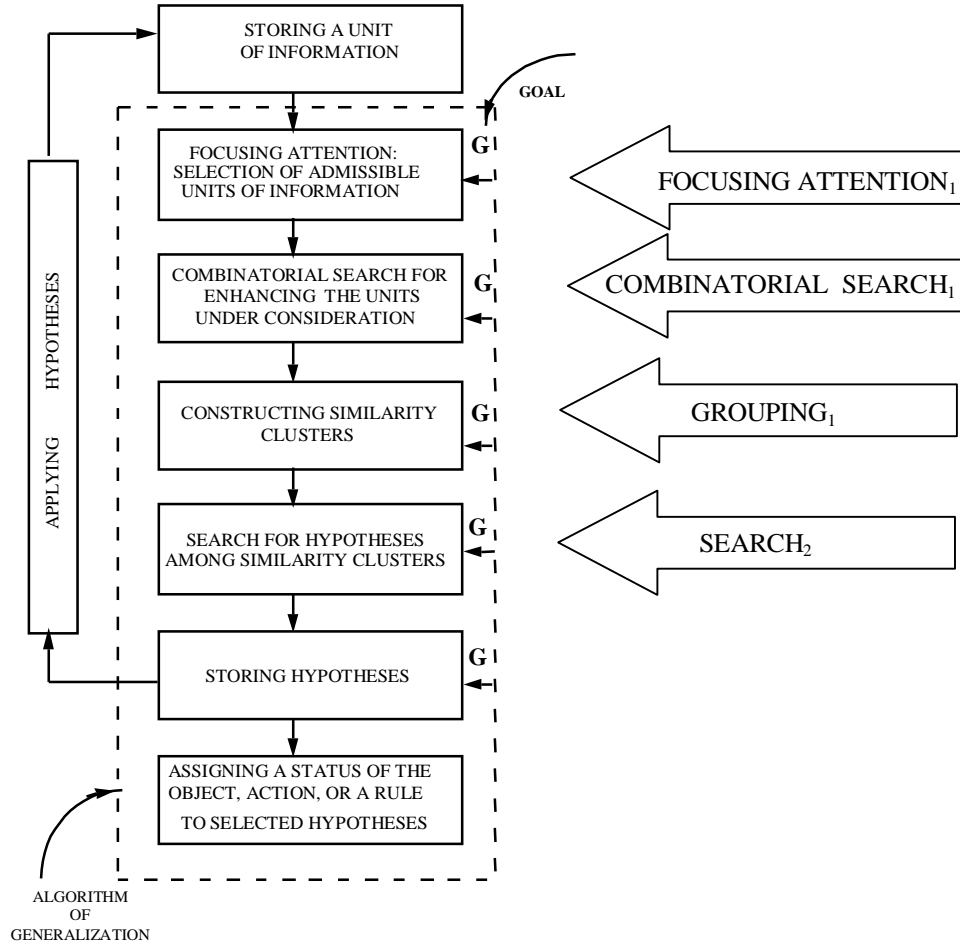


Figure 3. An Algorithm of Generalization and the Essence of its Operations

$$J = \frac{E_{consumed} \pm E_{lost}}{t_{curr} - t_0} \quad (4)$$

or

$$J = \frac{E_{consumed}}{E_{lost} \cdot (t_{curr} - t_0)} \quad (5)$$

Using (4) and (5) for performance evaluation is a not a very simple matter. The system might actually have many cost functionals pertaining to different levels of resolution. This can entail mutually conflicting processes of optimization. Therefore searching for an optimum motion trajectory in the multiresolutional state space would require recursive top-down/bottom-up algorithm of searching.

### 2.2.7 Ability to recognize and achieve goals

This ability should be considered an absolutely distinct feature of intelligent systems. In the simple artificial intelligent systems, only the highest goal (belonging to the lowest level of resolution) should be assigned to the ELF. The other goals will be obtained autonomously as a result of the planning process. Searching for an optimum motion trajectory at each level of resolution should be performed under a particular goal assigned for this level of resolution. In the case of intelligence for mobile autonomous vehicles a concept of *horizon of goal assignment*, or *horizon of planning* seems to eliminate many

difficulties in developing multiresolutional algorithms of behavior generation. The concept of “horizon” is introduced because of the following conjecture:

*Conjecture of Reduced Accuracy for Remote Objects and Events*

Under the same conditions and assumptions about the units of knowledge stored in the system of representation, the units that are remote spatially or temporally from the current state should be assigned lower accuracy because the risk increases of being affected by the sources of uncertainty.

As a result, the higher the resolution is the smaller is the horizon of goal assignment. Thus, from the results of finding the optimum trajectory of motion at low resolution, an intermediate state of this trajectory should be chosen as the intermediate goal-state for the level of higher resolution.

### 2.2.8 Emergence of “Self”

Discussions about intelligence are permeated by the statements related to “consciousness.” This paper decouples the issue of intelligence and the issue of consciousness by introducing the concept of representing “self.” The need in representing self arises at some level of early learning processes because of the need to increase the efficiency of planning [28]. At the initial stages of development of the robot intelligence, the whole World Model is being constructed relative to the robot. It is always situated in the center of the state representation. As the knowledge gets more complicated, the need emerges in representing the system in coordinates associated with the external system. This leads to a discovery similar to that known as the Copernicus Revolution (apparently, Ptolemy failed to put the “self” on the map).

The “self” emerges for an intelligent system (IC) after the representation of the IC itself becomes a part of the World Model constructed by IC and thus, the model of IC is shown within its own system of representation. Thus, the whole model of system (ELF, earlier shown in Figure 2) should emerge within the World Model as shown in Figure 4. If IC constructs its own ELF within its representation, it should have within its World Model a representation of itself, too. Thus, the idea of “self” leads immediately to a paradoxical demand of having within its system of representation an infinite system of nested models. Obviously, this is practically impossible. Of course, in practice one or two nesting would be totally sufficient.

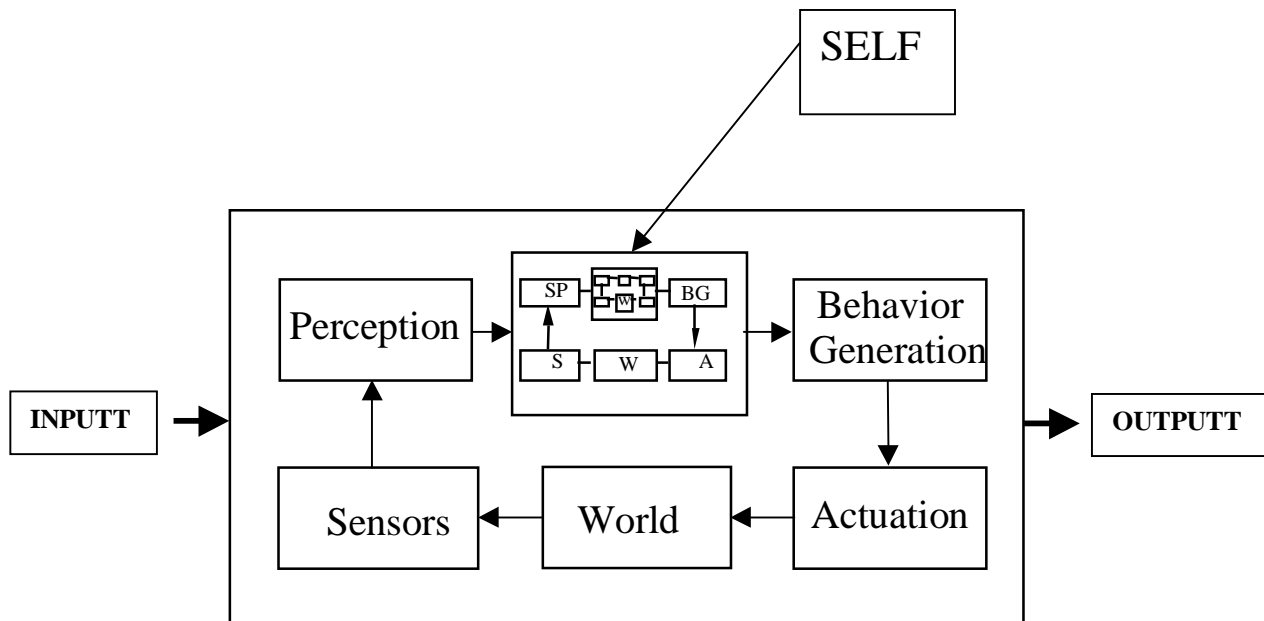


Figure 4. ELF with “self”

However, this is not the only paradoxical effect that can be listed for this phenomenon (called “reflexia” in scientific psychology). The situation gets more complicated when the World Model should also include the model of another intelligent system (IC) of a comparative level of intelligence. Then, the representation of

another IC should include its representation of the first IC, which contains in its representation the first IC with its representation of both the first and the second ICs.

One of the important consequences of the emergence of “self” is that a communication with this intelligent system is possible as if it would be an external system. Since the differences in World Models are possible between the initial ELF and the ELF of “self,” this inner self might have subtle differences in the decision making process. Algorithms of communications with “self” seem to be an interesting part of introspection, particularly, of “imagination.”

### 2.2.9 Imagination

This “self” can be considered a part of some more mundane processes that are known for many animals: the processes of imagination. Creation of “virtual reality” within our brains and supporting the decision making process by exploring alternative mentally seems to be a very powerful mechanism of intelligence increasing the efficiency of functioning. In artificial intelligent systems, “imagination” is synonymous with simulation of anticipated situations during the decision making.

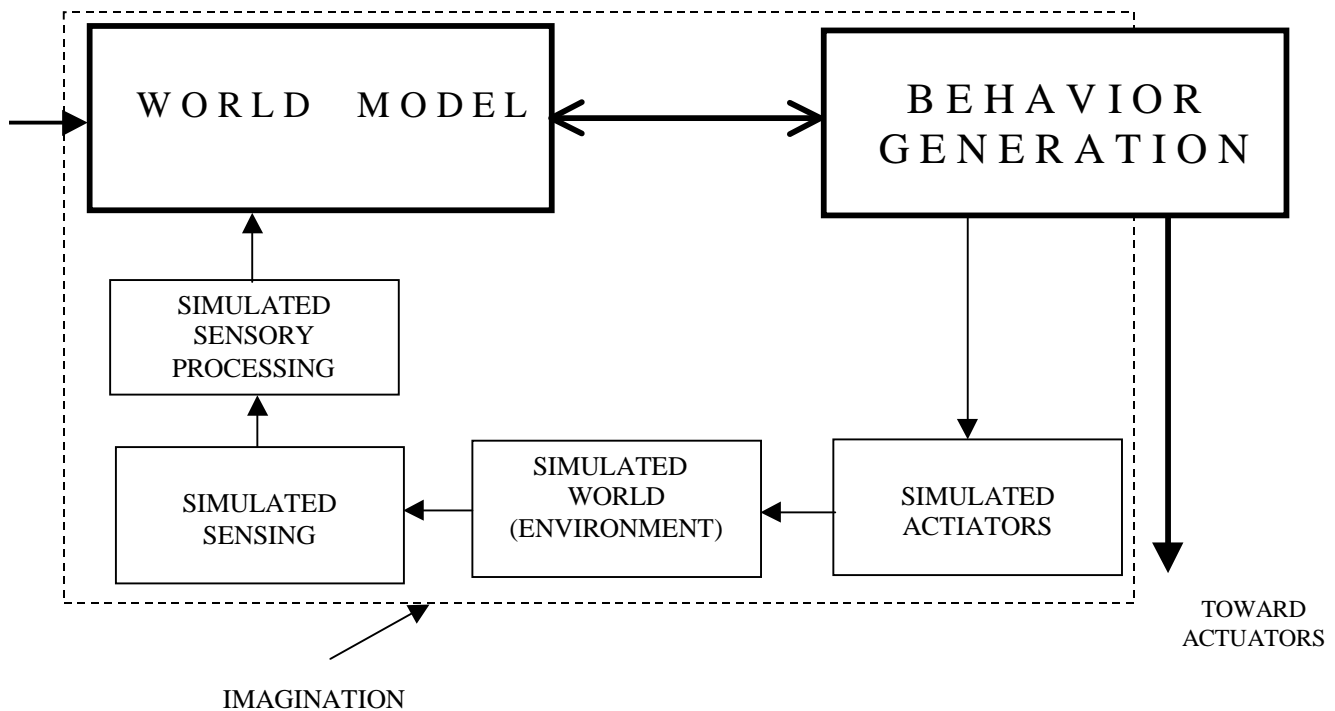


Figure 5. The system of Imagination emerging in the ELF

As Figure 5 demonstrates, instead of submitting the decision to the real actuators, the module of Behavior Generation submits it to the model of actuators, and simulates all consequences of this “WHAT IF” contemplation. IC simulates the events in the World, their development and simulates what will sensors deliver, and how sensory processing will work, and what will happen after new information is delivered to the World Model. Searching with simulating the consequences is a powerful tool of the intelligence, it is utilized for learning, planning, etc.

### 2.2.10 Autonomy

This property is frequently considered a synonym of “intelligence” since both of them presume each other. However, an objection is raised often that very autonomous systems can have low intelligence while very intelligent systems can be deprived of autonomy. Further analysis shows that the latter statement is not correct. If the system has low intelligence its autonomy is very limited within the world containing many systems of high intelligence. On the other hand, if the system has a high intelligence, it would require

a multiresolutional level of the effort to deprive it of autonomy. This means that it would require having other highly intelligent systems to curtail the autonomy of another highly intelligent IC. Frequently, introducing the autonomy constraints happens only for the one particular level of resolution.

The important implications for multiagent systems can be expected if this topic is pursued scientifically. At the present time there are many groups that pursue the research on autonomy of multiple agents. However, not too much research is conducted about multiple multiresolutional agents. One of the important issue is the following: how much should all agents-levels worry about cost-functions of each other taken in account that they are nested within some of them while other agents-levels are nested within them.

### 3. Terminological Notes

#### 3.1 Complexity

The term *complexity* is used in this paper in the following meaning: complexity is the property of a situation to consist of excessively large number of a) objects, b) relations between the objects and c) registered and modeled processes that include these objects and relationships as components, and d) unmodeled processes that depend on the stochastic factors and cannot be reliably modeled. The number should be considered excessively large if as a result of its value the cost-function that evaluates the goodness of the activities deteriorates. Evaluation of the number of components or connections, or processes, or all of the above factors should reflect the following facets of the performance:

- time of computation,
- reliability of functioning, or
- probability of emergence of the phenomena unaccounted for in the logical analysis.

In many recent publications there is a tendency to associate the term complexity only with the latter phenomenon from the list above (unmodeled processes). These references to something generated by complexity but difficult to model are actually references to the lack of knowledge of what is going on. Thus, in this paper we refer only to the phenomena “a” through “c” from the definition above.

#### 3.2 Reasoning

The term *reasoning* is understood as applying all or most of the rules consistently and directed toward the goal. Consistency of applying signifies the absence of contradictions (paradoxes), and provides for combining them in a proper sequence. Nevertheless, applications testify for existence of shortcoming in many techniques of reasoning stemming from the predicate calculus of the first order. This is known for a long time, and this is fly methods of fuzzy logic emerged together with the theories of belief and the possibilistic approaches to determine preferences.

It became clear recently, that the substantial part of failing cases of reasoning happens because the multiresolutional structure of representation is not taken in account by the process of reasoning, both in living creatures and in computer equipped intelligent systems. What is true in one level is not necessarily true in the adjacent levels. The temporal factor creates difficulties in a regular predicate calculus. Now, the situation gets aggravated by the different time scales. These considerations can be illustrated by the multiple examples. In many of them, reasoning is affected by the transformation of representation: while the quantities change the list of objects is changing, too [29], and this affects the results of generalization. The motion of “pointing” in living creatures was demonstrated to be affected by the different time scales at the different level of abstraction in brain [30].

Finally, it has been found from many observations that the logic of natural language is different from the one presented in the theory of predicate calculus of the first order. The inferences implied by the natural language discourse to not allow to be easily transformed into statements of the predicate calculus while their implications are eventually properly interpreted and understood by humans. It is tempting to develop a) a theory of natural language reasoning and b) an automated system that would allow to use the advantages of natural language reasoning for artificial intelligent machines. The researchers of Drexel University are working on these topics now.

### 3.3 Resolution

The term *resolution* related to the accuracy of detail in representation and sensor output is often confused with the term resolution from the subsections of logic in artificial intelligence (resolution-refutation). Resolution of the system's level is determined by the size of the indistinguishability zone (granule) for the representation of goal, model, plan and feedback law. Any control solution alludes to the idea of resolution explicitly or implicitly.

Resolution determines the complexity of computations directly because it determines a number of information units in a representation. In complex systems and situations one level of resolution is not sufficient because the total space of interest is usually large, and the final accuracy high enough. So, if the total space of interest is represented with the highest accuracy, the  $\epsilon$ -entropy (the measure of its complexity) of the system is very high.

The total space of interest is to be initially considered at a low resolution. Only one subset (or a limited set of subsets) of interest is further analyzed with higher resolution, and so on, until the highest resolution is achieved. This consecutive focusing of attention with narrowing the subsets' results in a multilevel task decomposition. The following terms are used with resolution intermittently: granulation, scale. "Granule" is another term of the distinguishability zone (pixel, voxel). Scale is considered to be equal to the inverted value of the granule (or an " $\epsilon$ -tile"). When the space is intentionally discretized, we use the term tessellation, and a single granule is called "tessellatum" or "tile."

### 3.4 Multiresolutional Representation

The term *multiresolutional representation* is defined as a data (knowledge) system for representing the model of our system at several levels of resolution (or granulation, or scales). In order to construct a multiresolutional (multiscale, multigranular) system of representation, the process of generalization is consecutively applied to the representation of the higher levels of resolution. As a result of applying the algorithm of generalization to the modules of ELF emerge (Figure 2) with the new level of Sensory Processing (SP), World Model (WM), and Behavior Generation (BG). These new, more generalized BG-WM-BG sets are attached to the initial ELFs as the next "floor" of this structure. If further generalization is performed on the modules of the new level, an additional level of SP-WM-BG of the structure would emerge.

Multiresolutional representation can be underlaid by an ERN principle of constructing the model. Objects, relations, and actions of the ERN at the new level are different, and thus, the rules are different and the results of searching for the best course of actions are presented in different terms. However, if necessary one can substitute it by other techniques of representing experimental knowledge, e.g. by using analytical models with different accuracy of approximation.

### 3.5 Generalization

The term *generalization* is a formation of new entities (groups, classes, assemblies) where parts to be assembled are not prespecified, and new classes of properties can emerge. *Synonym* - (sometimes) abstraction. *Antonym* - instantiation. *Generalization* usually presumes grouping (clustering) of the subsets focused upon as a result of searching and consecutive substitution of them by entities of the higher level of abstraction. This is why instead of term *resolution* levels we use sometimes an expression *levels of abstraction*, which means the same as levels of *generalization*, or levels of *granularity*. Example: In most of the cases when humans encounter new situations they face the need to create groups. They make groups or assemble together components, which are not specified as parts belonging to each other, and new classes of properties should be proposed on the flight.

From the definition of generalization, one can see that it can be performed through applying the following operators jointly: *grouping*, *focusing attention*, *combinatorial search* (or a simple *search*). There are many operators that exhibit these functions: many algorithms of clustering that can be used to perform *grouping*, many algorithm of choosing the subset of interest, e.g. windowing operators that perform focusing attention, many algorithms of search, or search equipped with combinatorial generation objects among which the search is done. To simplify further analysis of architecture we will call them operators of G, FA, CS, or about an integrated operator of GFACS.

It would be instructive to demonstrate how the term *generalization* differs from terms *aggregation* and *abstraction*. *Aggregation* is formation of an entity out of its parts. Each of the parts can be also obtained as a part of aggregation. *Synonym* - assembling. *Antonym* - decomposition. Example: The entity is formed out of its parts. Information of belonging is contained in the description of the objects. We will consider this process to be an example of a very simple group formation: we know what is the whole, and we know what are the parts. Assembling of parts into the whole, or formation of an aggregate is determined by specifications.

Formation of a class of objects which is characterized by the same property, and labeling this class with the name of this property is called *abstraction*. *Synonym* - class formation, sometimes, abstraction. *Antonym* - specialization. Example: The properties, which characterized objects can be considered objects by themselves. We won't be surprised if one calls *kindness* an entity. The fact that color is a property belonging to the most physical objects of the real world makes it an important scientific and technological entity of the system of knowledge. It is important to indicate that formation of such entity is possible only by grouping together all similar properties of different objects. A red apple, red ink, red bird, red cheeks, they all belong to the class of objects containing "redness".

So, generalization performs aggregation even when parts are not specified. This means that it subsumes the aggregation. It subsumes the abstraction, too. In all cases concerning abstraction the term generalization is applicable. Generalization is typically applied when a similarity and observations are discovered and a general rule should be introduced. The term abstraction is inappropriate in this case. Conclusion: *generalization subsumes both aggregation and abstraction*. This is a more general procedure for which aggregation and abstraction are particular cases.

### 3.6 Nesting

Nesting is a property of recursively applying the same procedures of multiresolutional knowledge processing by using the operator of processing at a level for consecutively processing information of all levels. The results of Sensory Processing of all levels are nested one within another, World Models are nested one within another, and the decisions generated within the module of Behavior Generation are nested one within another. Levels of a multiresolutional ELF are nested one within another, while the levels continue to function as separate independent ELFs. This separation of levels is a result of a need to reduce the complexity of computations. Thus, instead of solving in one shot the whole problem with the maximum volume of the state space and with the amount of high resolution details one may choose to solve several substantially simpler problems that are nested one within another.

### 3.7 Learning

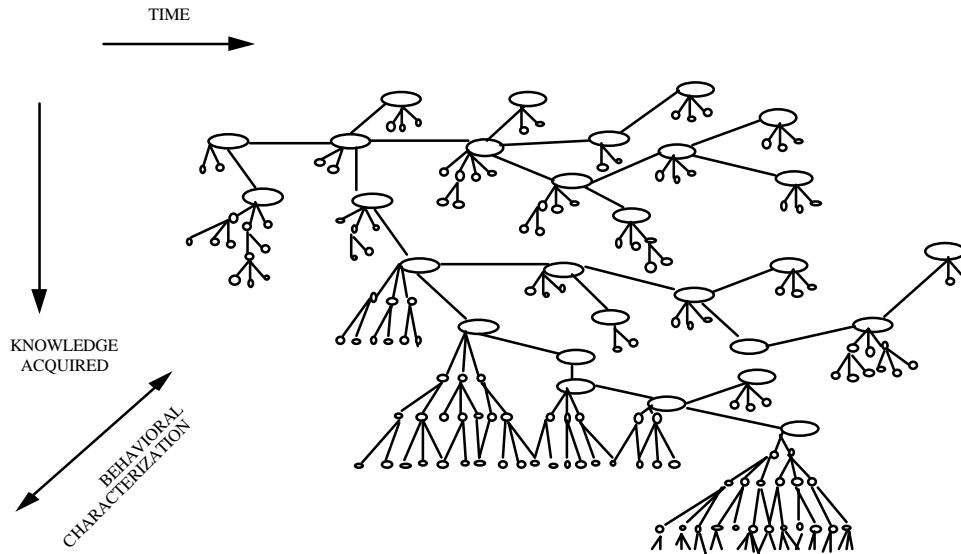
The process of generalization upon the time-varying functions of a control system is called learning. As a result of this process, the statements of experiences related to elementary objects, relationships between objects, statements of actions merge together into statements related to clusters of objects, relationships and actions. These, generalized statements allow for construction of rules. Then, all this newly obtained set of statements can be generalized again. This process that is being performed recursively and is successively applied to its own output results in creating and constant updating of the multiresolutional system of representation, and thus, in improvement of plans and feedback control laws. Learning is a component of this multiresolutional knowledge processing. Evolution of knowledge of the system can be demonstrated as shown in Figure 6.

Obviously, learning is tightly linked with the property of Intelligent Systems of being equipped by the systems of knowledge representation (e.g. the module of World Model in ELF). This module of representation might not necessarily be physically lumped in one specific place: WM can be distributed over a multiplicity of agents, or otherwise over the physical medium used in the intelligent system.

Updating of the World Model and enhancement of its multiresolutional system of knowledge representation is done by the process of learning, which employs the set of GFACS operators that has been described above. Levels of resolution are selected to minimize the complexity of computations. Planning and

determining of the beneficial feedback control laws is done also by joint using of generalization, focusing attention, and combinatorial search (GFACS).

The operation of learning was associated with layers: each layer learns separately. Learning experiences can be organized only by using a multiresolutional structure. Levels are not hard-wired, they are constructed



**Figure 6. Evolution of World Model as a Result of Learning**

from the information at hand. As it is done in neural net, for example. Mathematics of various operators of focusing attention, grouping and searching usually employed by GFACS algorithms can be found in [31].

One can see from Figure 7 that Learning in an intelligent system boils down to collecting experiences, applying GFACS to them, and explicating objects, actions, rules and theories that might be used by the module of Behavior Generation. Combining Figure 7 with Figure 5 gives an opportunity to learn not only from real experiences of acting within the environment but also from the imaginary experiences of simulating within the imagination of the intelligence.

### 3.8 Intelligent Control

Intelligent control is a computationally efficient procedure of directing to a goal of a complex system with incomplete and inadequate representation and under incomplete specification of how to do this in an uncertain environment. Intelligent control, typically, combines planning with on-line error compensation, it requires learning of both the system and the environment to be a part of the control process. Most importantly, intelligent control usually employs generalization (G), focusing attention (FA) and combinatorial search (CS) as their primary operator (GFACS) which leads to multiscale structure. In all intelligent controllers, one can easily demonstrate the presence of the GFACS operators. It also is possible to demonstrate that using the set of GFACS operators is not typical for conventional controllers, although the elements of GFACS are often utilized.

Not accidentally, at the dawn of intelligent control it was associated with using neural networks (NN), fuzzy sets (FS) and generic algorithms (GA) for control purposes. (In some publications, these tree subjects are considered a must for intelligent control). In fact, neural networks is a tool for generalization in the vicinity of the state space, fuzzy systems allow to expand the process of generalization to the larger domains of the state space. GA is just a particular case of combinatorial search with some component of internal generalization for learning purposes). In other words, NN+FS+GA is a particular case of GFACS. Thus, the views presented above are confirmed.

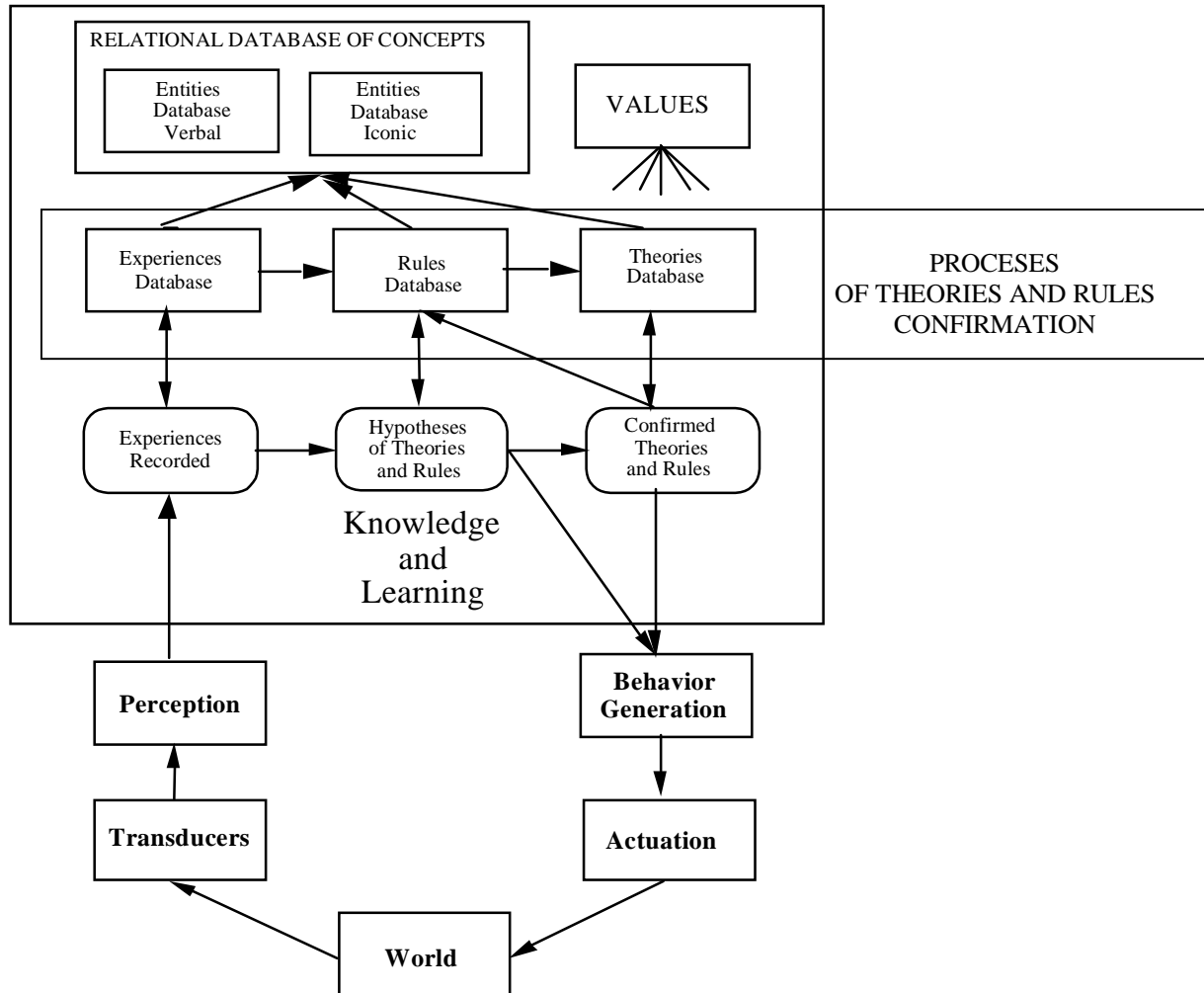


Figure 7. ELF with Learning

#### 4. More Formal Definitions of Intelligence

Most of the literature on *intelligence* can be found within the stream of publications related to psychological sciences. Most often, this is not exactly the intelligence that is discussed in this paper: they are talking about *human intelligence* primarily (like in [32]). However, even the most fundamental collections of sources do not define intelligence in the way other than listing of the mental abilities that are components of intelligence. We already spoke about immensity of *abilities* associated with intelligence. In the sources related to psychological science, intelligence is typically defined as a mental quality that combines:

1. ability to learn from experience
2. ability to adopt to new situations
3. ability to understand and handle new concepts
4. ability to acquire and use knowledge

The following definition is based on a term *thinking*:

“[An] action exhibit intelligence, if, and only if, the agent is thinking what he is doing while he is doing it, and thinking what he is doing in such a manner that he would not do the action so well if he were not thinking what he is doing” ([1], p.29). *Thinking* is understood as a process of mediation between inner activities and external stimuli. It always alludes to the need in a specific



language of thought [33] and provide for a substantive link between the mechanisms of intelligence and a computational process [34].

Several definitions are presented in [35]. One of them belongs to J. Albus and can be found in [21]:

“An intelligent system has the ability to act appropriately in an uncertain environment, where an appropriate action is that which increase the probability of success, and success is the achievement of behavioral subgoals that support the system’s ultimate goal.”

This definition generalizes upon multiple abilities mentioned in the psychological definitions and introduces a concept of a success associated with the behavioral subgoals that presume some hierarchy of activities (with an inevitable multiresolutional representation). This definition is dominating since it is not linked with a particular configuration, neither it alludes to any particular domain of application. Clearly, one can apply this definition for both living creatures and artificial intelligent systems.

The operational definition introduced by A. Meystel and partially presented in [33] explains how the intelligence works:

“Intelligence is a property of the system that emerges when the procedures of focusing attention, combinatorial search, and generalization are applied to the input information in order to produce the process of intelligent system functioning.”

Focus in this definition is how information is processed so that it makes this mechanism *intelligence*. Earlier in this paper, there was more about procedures involved (GFACS) and representations required (World Model in one of the available forms of e.g. ERN).

More technologically inclined definition from [33] demands to concentrate on the issue of uncertainty (that was already mentioned in Albus’ definition):

“Machine intelligence is the process of analyzing, organizing and converting data into knowledge, where machine knowledge is defined to be the structured information acquired and applied to remove ignorance or uncertainty about a specific task pertaining to the intelligence machine.”

Focus in this definition: converting data into knowledge by removal of uncertainty.

All the above definitions can be supplemented by informative statements describing features typical for intelligence but not reflected in the definitions. For example, a technological system with intelligence, i. e. intelligent system, undoubtedly can deal with unanticipated factors due to the ability to learn:

An intelligent system must be highly adaptable to significant unanticipated changes, and so learning is essential. It must exhibit a high degree of autonomy in dealing with changes. It must be able to deal with significant complexity, and this leads to certain sparse types of functional architectures such as hierarchies.

As a byproduct of all these abilities, a number of additional features emerge gradually in a developing intelligent system. For example, a feature of *autonomy* is associated with intelligence although we still do not know how. Dealing with complexity requires using multiple resolutions, because functional hierarchical architectures are declared. Thus, the feature of being based upon multiresolutional representations is a fundamental one. Having in its core a semiotic closure is typical for an intelligent system, too.

Now we try to create a synthetic definition absorbing the definitions and supplementary statements above:

Intelligence is a control tool that has emerged as a result of evolution by rewarding systems with increase of the probability of success under informational uncertainty. Intelligence allows for a redundancy in its features of functioning simultaneously with reduction of computational complexity by using a loop of semiotic closure equipped by a mechanism of generalization for the purposes of learning. Intelligence grows through the generation of multiresolutional system of knowledge representation and processing.

The multi-level systems fitting into this definition are not necessarily hardwired hierarchies. They are *virtual hierarchies* of perception, of knowledge- representation about the world model, and of decisions about

behavior generation. As a new concept of “knowledge” emerges, a new “node” of the representation ERN is being created.

From this effort to scan existing and create a new definition, new analytical and research tasks precipitate. It becomes clear that the system of intelligence should be equipped by a capability to properly measure the objects, relations, actions and behaviors. Thus, the problem of evaluating metrics of performance and intelligence emerges. It becomes clear that intelligence can be evaluated by “a degree of intelligence”. One can see that the definitions explored in this sub-section allude to the need of measure and perform quantitative ranking that is supposed to end up with a choice of a decision making. The definition of intelligent control should be based on the properties of intelligence as we understand them rather than the virtue of using some particular hardware components.

There is the list of factors that are supposed to be measured for evaluating an intelligent system:

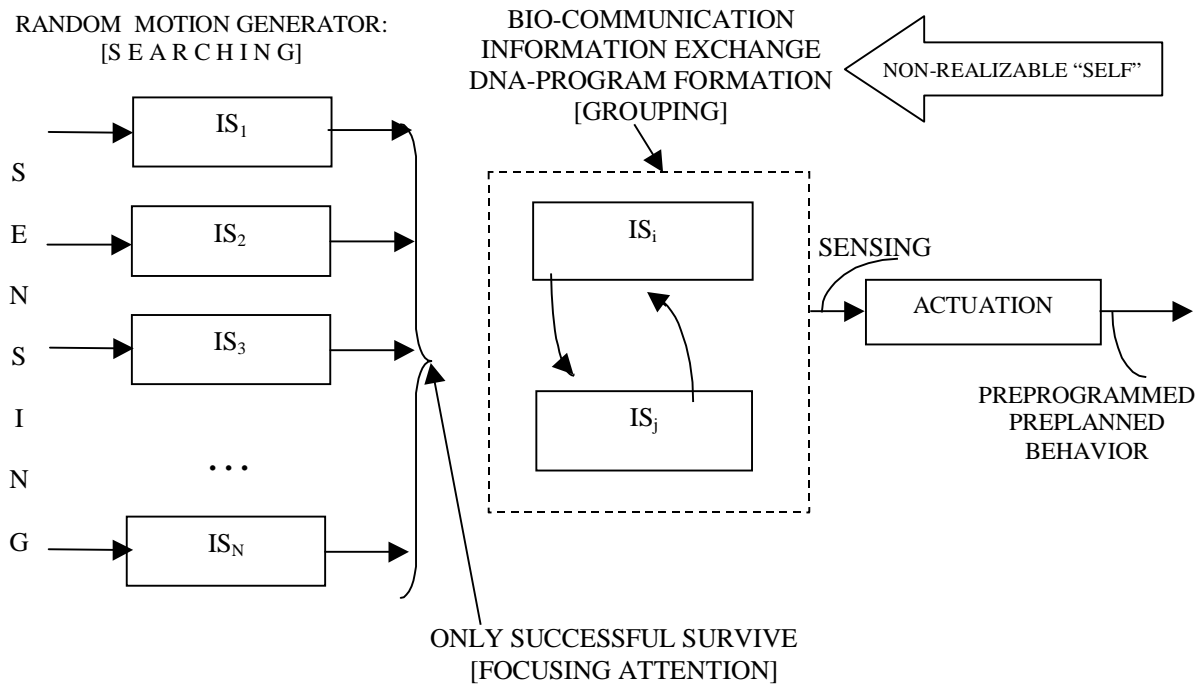
- *Complexity Reduction.*  
Complexity should be evaluated and possibly the lowest level of complexity should be preferred under other similar conditions. Reduction of complexity should not bind the capability to develop redundancy.
- *Redundancy*  
A measure of “exceeding” the immediate needs should be introduced; one can see that the ability to evaluate the “immediate needs” is required.
- *Increase in Functionality*  
The design specifications can be used to evaluate the measure for both “immediate needs” and items of “functionality.” However, in many cases, the specifications are not available. Definitely, an ability to evaluate functionality quantitatively would be an advantage, but we have to know how to restore their list.
- *Multi-level Systems.*  
The practice of intelligent system design demonstrates that the number of resolution levels is being selected based upon heuristics, not clear mathematical analysis of advantages and disadvantages this number entails. Designers do not know how many levels should a system have, and what are the other quantitative factors involved in assigning a number of levels to a multiresolutional system.
- *Degree of Intelligence.*  
A measure of intelligence is presumed to be known, at least a relative measure (which one system is smarter if general parameters are the same but different mechanisms of sensory processing, or different algorithms of planning are applied).
- *Degree of Autonomy.*  
A measure of Autonomy is presumed for the systems that are supposed to decide their own course of actions for themselves.
- *GFACS*  
At the present time there is a multiplicity of the mechanisms (algorithms) of generalization. We do not have any basis for comparing the results of their functioning. Even in more simple cases (e.g. focusing attention in ARMA algorithms we do not have recommendation of comparing different versions of ARMA.
- *Increase in probability of success.*  
How should success be evaluated depends on our ability to specify it. (Is this money, power, knowledge, ability to live longer? Are these outcomes anticipated as a measure of success when they are computed for the system under consideration, or for the group, or for several generations?)

## **5. Evolution of Intelligence**

In nature, the evolution of intelligence can be demonstrated as the development of a tool of survival. This tool evolved in living creatures (systems) as a control mechanism (a controller) to optimize needs satisfaction in changing environment. As the complexity of needs was growing, in addition to creating ways of their satisfaction the duty was performed to accordingly develop the mechanism of intelligence. The major destination of intelligence is to solving harmoniously the combined task of *NEEDS SATISFACTION + COMPUTATIONAL COMPLEXITY REDUCTION*.

Increasing functionality for performing this task can and should be measured. The evolution of intelligence presumes the evolution of both the system and the controller. The proper measure allows to judge results of evolution of intelligence. Evolution or development allows for increasing the functionality of the system jointly with reduction of its computational complexity. This is why the ability to *generalize* emerges, as the ability to lump entities of matter and/or information for more effective storing and computation. Generalization is a tool of creating new, abridged systems and their representations. It is a tool of creating representations in generalities, creating new levels (generalized) of lower resolution with new metrics or granulation. At the lower level of resolution, the tools of intelligence can afford a larger scope of attention, solve a problem of a larger picture, with a longer horizon of planning. So, the decision making on any given resolution should be preceded by the preplanning at a lower resolution level.

The biological models allow us to observe the growth of the degree of intelligence in the living forms starting with single cell organisms, through E.coli [6], via substantially more complicated living forms from mollusks to mammals [36], and concluding with a human being [32] (See Figures 8,9,10 developed for E.coli level within the paradigm of [6] with [37])



**Figure 8. Architecture of E.coli's Intelligence**

Figure 8 shows the group level of the E.coli intelligence architecture. Each individual E.coli is shown as a separate intelligent system  $IS_i$ . Random motion of all E.coli in this group (combinatorial search) leads to the situation that only those moving successfully survive (focusing attention). The survived E.coli individuals communicate and share their experience via reproduction (grouping). The successful behavior is a result of this bio-information exchange and the results of GFACS operations are stored in the DNA, and the group (as a level) changes its behavior, and produces a behavior which increases the fraction of survived individual in the group. Thus, it can be considered "preprogrammed, preplanned behavior" at the level of the group. The group ELF has actually as modified its World Model as shown in Figure 9.

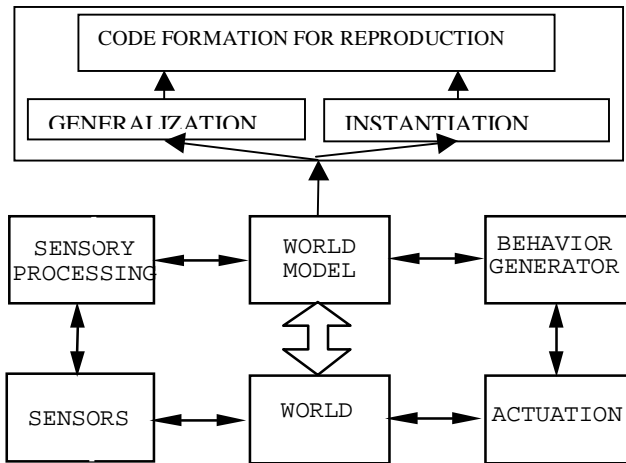


Figure 9. Developing World Model via reproduction

The architecture shown in Figure 8 can be developed into structure presented by Fig. 10, where learning is performed not through sacrificing unsuccessful individual but through abandoning unsuccessful “theories” tested by internal simulation within the rudimentary just emerging system of representation. Here, instead of the multiplicity of individuals we have ERN of objects and relations that are generalized not by communication via reproduction but computationally by applying GFACS embodied as a set of information processing procedures.

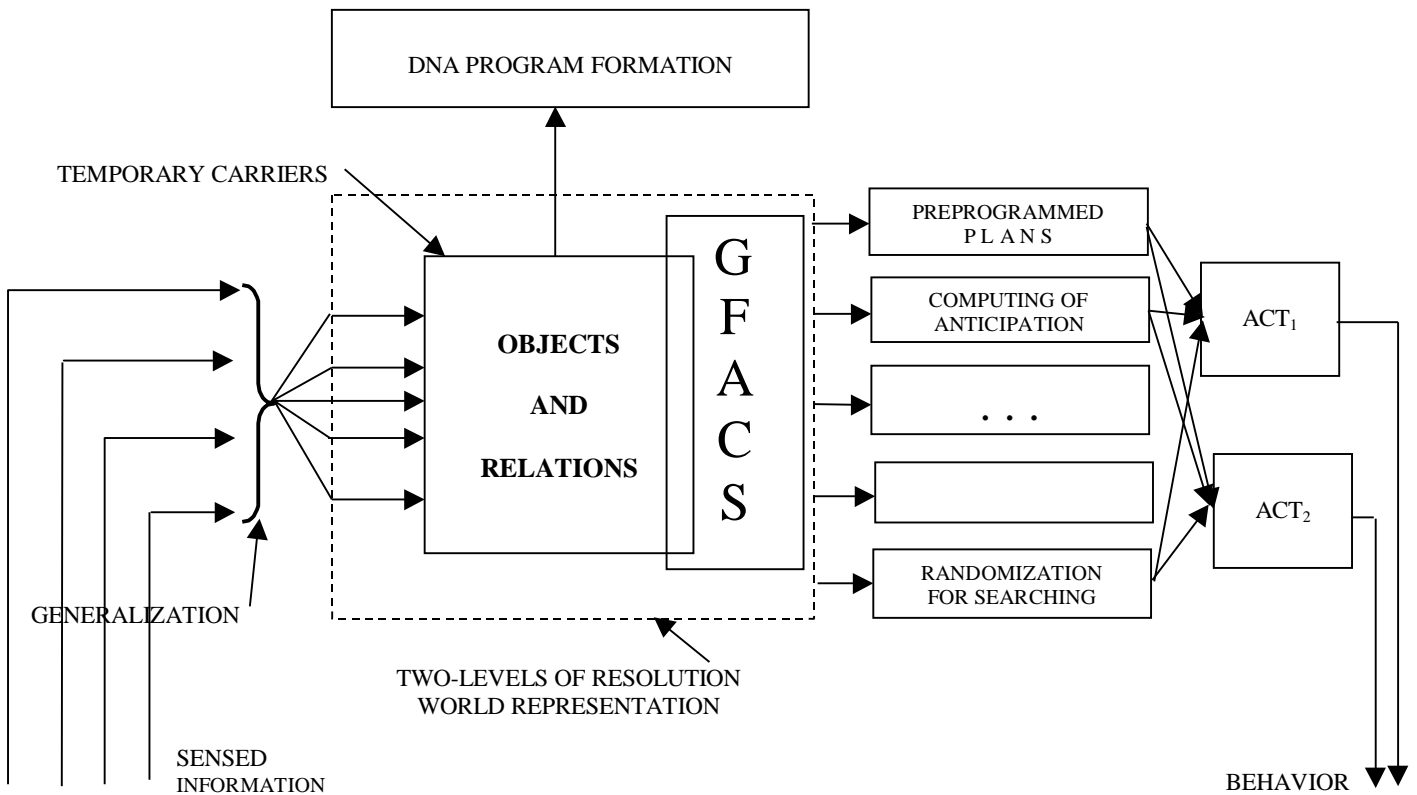


Figure. 10 Evolution from group intelligence to individual intelligence

Similar evolution can be observed in the domain of technology and in the domain of Linguistics. The processes of intelligence evolution extracted from these domains can be discussed in a generalized form by using architectures similar to Figures 8 through 10. Some of the advanced technological architectures (e.g. RCS) are described in [38].

Interesting temporal effect can be anticipated in the process of evolution. One can easily anticipate that the evolution is a “punctuated” one<sup>2</sup> (in the sense of [36]) since the new blocks only occasionally emerge in the architectures of intelligent systems.

## 6. Mechanisms of Intelligence

Analyses of the processes of structural evolution in the area of intelligence allow for discovering the following mechanisms of intelligence.

- A semiotic closure is the basic structure of intelligence (see Figure 2). It differs from a simple feedback loop because each element of the closure is a source of redundancy and a generator of the adjacent resolution levels by the virtue of GFACS operation.
- Evolution of multiple-choice preprogrammed behavior into a multiple alternative creation ends up with multiple theories development (the latter is performed in the *imagination*).
- Through combinatorial search, focusing attention and grouping performed in Nature by the mechanism of natural selection<sup>3</sup> the discovery of more efficient techniques was done. It was discovered by the intelligent agents that storing information about objects of the world, actions they encounter, and rules entailed by the changes is more efficient. Indeed, it is less expensive than testing the same material (often, living) samples again and again to receive similar results.
- Generalization and learning through natural selection from the choices created by material alternatives has demonstrated to be a waste of time, energy and matter. It is more efficient to learn by dealing with information only, i.e. by theorizing (THEORY → THE RESULT OF GENERALIZATION UPON RULES, RULE → RESULT OF GENERALIZATION UPON EXPERIENCES)
- Mechanisms of generalization give a consistent explanation to the semiotic tools of evolution discovered earlier in [39, 40]. The same mechanisms and tools determine that the ultimate methodology of analysis of the mechanisms of intelligence can be defined and realized successfully in the domain of Natural Language analysis.
- Any RULE discovered by an intelligence is a statement of some generality: it cannot refer to all details of realistic test cases. The selection of the proper details for the particular state of affair is performed by the mechanism of focusing attention.
- Multiresolutional storage obtained via consecutive generalization turned out to be the most efficient method of storing information.

## 7. Intelligence of the Conventional Controller

It would be desirable to determine what is the relation between the conventional control and intelligent control. The following statements are based on the preceding materials.

1. Conventional control is about feedback. The goal formation is external to the problem. When we include the goal formation the problem become IC-embedded because the goal for each level of the higher resolution is created as a result of BG-module functioning at the level of lower resolution.

---

<sup>2</sup> Punctuated evolution demonstrate periods of changes with intervals of the absence of any development.

<sup>3</sup> Actually, the reader should have already anticipated the conjecture that natural selection in the Nature played a role similar to the algorithmic mechanisms of generalization and learning.

2. The structures of intelligent control are formed as semiotic closures, mostly the multiresolutional ones, which contain an element that can be called “feedback”. But feedback is not the entire issue. The transformations within the feedback loop are more important. The classical feedback does not need to have any redundancy in it. This is why Y.-C. Fu associated the concept of Intelligent Control with “recognition” in the loop.
3. We would expect that the feedback of the semiotic closure contains GFACS as a rule.
4. Optimization as a part of functioning of the conventional controller presumes searching at a level but stops short from recognition its embedding within the multiresolutional hierarchy of top-down constraint propagation.

## References

1. G. Rile, *The Concept of Mind*, Hutchinson, London, 1949
2. A. Meystel, “Intelligent Systems: A Semiotic Perspective”, *International Journal of Intelligent Control and Systems*, Vol.1, No. 1, pp. 31-58
3. Y. Maximov, A. Meystel, “Optimum Architectures for Multiresolutional Control”, Proc. IEEE Conference on Aerospace Systems, May 25-27, Westlake Village, CA 1993
4. A. Meystel, “Planning in a hierarchical nested controller for autonomous robots,” Proc. IEEE 25th Conf. on Decision and Control, Athens, Greece, pp. 1237-1249
5. (self-organization)
6. K. M. Passino, “Distributed Optimization and Control Using Only a Germ of Intelligence”, Proc. of the 2000 IEEE Int’l Symposium on Intelligent Control, Eds. P. Groumpos, N. Koussoulas, M. Polycarpou, Patras, Greece, 2000, pp. P5-P13
7. J. S. Parkinson, D. F. Blair, “Does E.coli Have a Nose?” *Science*, Vol. 259, 19 March 1993, pp. 1701-1702
8. A. N. Kolmogorov, “On Some Asymptotic Characteristics of Bounded Metric Spaces,” Proceedings of Academy of Sciences, Vol. 108, No. 3, 1956 (Doklady Akademii Nauk, in Russian).
9. M. Rackovic, D. Surla, M. Vukobratovic, “On Reducing Numerical Complexity of Complex Robot Dynamics,” J. of Intelligent and Robotic Systems, Vol. 24, 1999, pp. 269-293
10. E. Weyuker, “Evaluating Software Complexity Measures,” IEEE Transactions on Software Engineering, Vol. 14, No. 9, 1988, pp. 1357-1365
11. D. Boekee, R. Kraak, E. Backer, “On Complexity and Syntactic Information,” IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-12, No. 1, 1982, pp. 71-79
12. Y. Abu-Mostafa, “The Complexity of Information Extraction,” IEEE Transactions on Information Theory, Vol. IT-32, No. 4, 1986, pp. 513-531
13. G. Zames, “On the Metric Complexity of Causal Linear Systems: Epsilon-Entropy and Epsilon-Dimension for Continuous Time,” IEEE Transactions on Automatic Control, Vol. AC-24, No. 2, 1979, pp. 222-230
14. A. Meystel, “Architectures, Representations, and Algorithms for Intelligent Control of Robots,” (Chapter 27) in *Intelligent Control Systems*, eds. by M. Gupta and N. Singha, IEEE Press, 1995, pp. 732-788
15. A. Meystel, “Multiresolutional Architectures for Autonomous Systems with Incomplete and Inadequate Knowledge Representation”, in *Artificial Intelligence in Industrial Decision Making, Control, and Automation*, eds. S. G. Tzafestas, H. B. Verbruggen, Kluwer Academic, 1995, pp. 159-223
16. A. Meystel, “Learning Algorithms Generating Multigranular Hierarchies,” DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 37, 1997, American Mathematical Society, pp. 357-383
17. A. Meystel, E. Messina, “The Challenge of Intelligent Systems,” Proc. of the 2000 IEEE Int’l Symposium on Intelligent Control, Eds. P. Groumpos, N. Koussoulas, M. Polycarpou, Patras, Greece, 2000, pp. 211-216
18. L. von Bertalanfy, *Robots Men and Minds*, Braziler, New York, 1967 [see p. 69]

19. H. Pattee, "Physical basis and origin of control," in Hierarchy Theory, Ed. H. Pattee, Braziler, New York, 1973 [see p. 94]
20. A. Meystel, "Theoretical Foundations of Planning and Navigation for Autonomous Robots," *International J. of Intelligent Systems*, Vol. II, 1987, pp. 73-128
21. J. Albus, Outline of the Theory of Intelligence, *IEEE Transactions on Systems, Man, and Cybernetics*, 1991
22. I. Rock, S. Palmer, "The Legacy of Gestalt Psychology," *Scientific American*, December 1990, pp. 84-90
23. J. McCarthy, "Generality in Artificial Intelligence," *Communications of the ACM*, Vol. 30, No. 12, December 1987, pp. 1030-1035
24. B. Porat, B. Friedlander, ARMA Special Estimation of Time Series with Missing Observations, *IEEE Transactions on Information Theory*, Vol. IT-30, No. 6, 1984, pp. 823-831
25. D. N. Politis, ARMA Models, "Prewhitening and Minimum Cross Entropy," *IEEE Transactions on Signal Processing*, Vol. 41, No. 2, 1993, pp. 781-787
26. P. Combettes, J.-C. Pesquet, "Convex Multiresolution Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 12, 1998, pp. 1308-1318
27. J. A. Bangham, P. Chardaire, C. J. Pye, P. D. Ling, "Multiscale Nonlinear Decomposition: The Sieve Decomposition Theorem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 5, 1996, pp. 529-539
28. A. Meystel, "Baby-robot: On the analysis of cognitive controllers for robotics," Proceedings of the IEEE Int'l Conference on Man & Cybernetics, Tuscon, AZ, Nov. 11-15, 1985, pp. 327-222
29. S. Murthy, "Qualitative Reasoning at Multiple Resolutions," Proceedings of the 7<sup>th</sup> Nat'l Conference on Artificial Intelligence, AAAI-88, Vol. 1, 1988, pp. 296-300
30. D. Ballard, M. Hayhoe, P. Pook, R. Rao, "Deictic Codes for the Embodiment of Cognition," BBS, Cambridge University Press, 1996
31. M. Vidyasagar, A Theory of Learning and Generalization, Springer, London, 1997
32. Handbook of Human Intelligence, Ed. R. J. Sternberg, Cambridge University Press, Cambridge, UK, 1982
33. J. A. Fodor, "Why There Still Has to Be a Language of Thought," in Psychosemantics by J. A. Fodor, MIT Press, 1987, pp. 135-167
34. G. Fauconnier, Mapping in Thought and Language, Cambridge University Press, Cambridge, UK, 1997
35. P. Antsaklis, "Defining Intelligent Control: Report of the Task Force on Intelligent Control," *IEEE Control Systems Magazine*, June 1994, pp. 4, 5, 58-66
36. S. Gould, "Triumph of the Root-Heades," *Natural History*, No. 1, 1996, pp. 10-14
37. M. Lieber, "Adaptive Mutations and Biological Evolution," *Frontier Perspectives*, Temple University, Spring-Summer 1991, pp. 23 and 26
38. J. Albus, A. Meystel, "A Reference Model Architecture for Design and Implementation of Intelligent Control in Large and Complex Systems", *International Journal of Intelligent Control and Systems*", Vol.1, No. 1, pp. 15-30
39. G. M. Tomkins, "The Metabolic Code," *Science*, Vol. 189, 5 September 1975, pp. 760-764
40. K. L. Bellman, L. J. Goldberg, "Common Origin of Linguistic and Movement Abilities," *American Psychological Society*, 1984, pp. R915-R921

# Machine IQ with Stable Cybernetic Learning with and without teacher

Harold Szu, Ph.D. , Fellow IEEE

Digital Media Lab, ECE Dept. GWU, 22 & H St. NW, Wash DC 20052

## 1. MACHINE IQ

Lotfi Zadeh has raised an interesting and philosophical question: what is the Machine Intelligent Quotient (IQ) needed for intelligent household consumer electronics and robots?

We wish to suggest a nonlinear but monotonic scale similar to the logarithmic scale adopted by C. Shannon information theorem based on the logarithmic phase space in L. Boltzmann entropy notion in Sect. 5. The justification is that the human-like creativity is rare and difficult and must be reached at the top 50% scale with unsupervised learning in Sect. 2 & 3, and the dumber machine near the low end of the scale. In between they are separated by dyadic basis. However, for household convenience, after taking the logarithmic nonlinear scale of human-like intelligence, the net result is further measured by taking the usual linear percentage scale. We concede that these double scales may be sensible in the operational definition but could not be fundamental. For we are not absolutely certain about what are the necessary and sufficient ingredients for a humankind or machine to be intelligent.

- (1) After we have taken the logarithmic scale, then  $MIQ=10\%$  of human being is loyal to human master and its own survivability, say, the robot having  $MIQ=10\%$  is able to find and differentiate the electric power plugs having two prongs of 110 Volts or three prongs of 200 Volts.
- (2) Then,  $MIQ=20\%$  is able to understand human conversation.
- (3) In that direction we can extrapolate  $MIQ=30\%$  to be able to read facial expression and voice tone for the emotional IQ to understanding irrational emotion need of human being.
- (4)  $MIQ=40\%$  is able to command and control a team of other robots.
- (5)  $MIQ=50\%$  is able to “explore the tolerance of imprecision,” e.g. using fuzzy logic to negotiate a single precision path finding in an open save terrain.

We divide MIQ into the supervised learning with an open lookup table having the

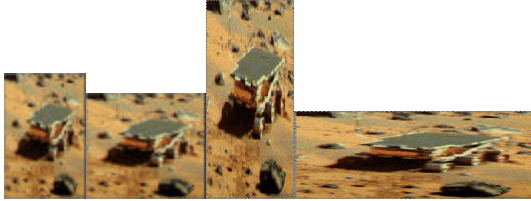
extrapolation and interpolation capability up to  $MIQ \leq 50\%$ . The key of human-like sensor systems is learning without supervision to be scored MIQ beyond 50%. Such a learning methodology is necessary, because, other than factory robots, any indispensable need of robots happens usually in an open, uncooperative and hazardous environment with an unforeseeable nonlinear dynamics interwoven with non-stationary complexity. We believe that unsupervised learning is necessary in building of human-like trial-and-error estimation systems, such that a major next step toward the intelligent robot is visual and natural language understanding needed for self-determination in uncontrollable environment. Thus, laboratory simulations of robotic teams are necessary with the help of the wireless video feedback control technology base (WaveNet video communication devices on wheels).

Intelligent processing is a need common to real world surveillance and control, especially in unmanned environment, namely the nucleus reactor chamber, the undersea, and the outer space. To set the research direction of real world applications, we consider a not-yet solved and perhaps a milestone problem to design an intelligent robot entering into a new challenging situation (not unlike a newborn infant facing a bustling and hustling world). This is challenging because the robot has not yet acquired any pertinent internal knowledge representation. The robot must learn how to process the unfamiliar sensory, hearing and vision inputs and to travel through an uncharted environment (even if the robot has been endowed with the past experience and has had a man in the loop as the coach for the remote guidance and control). In a distant and novel situation, we anticipate the robot having some difficulty in following the supervised learning given that the robot has not yet learned what is the desired output for the unfamiliar inputs. The milestone problem is thus due to the impossibility to specify all details ahead of the time, and therefore it is important to develop unsupervised sensory learning capability (which leverages subsequently the self-supervised learning with the gradually acquired experience).

A real world challenge happened recently during the NASA PathFinder exploring the Mar, of which the round trip time for Control, Command, & Communication (C3) is 5 minutes. A futuristic



suggestion would be using ANN for intelligent pathfinder leader to control the rest of team members, and then local control will be instantaneous, while we control the leader with non-real time commands (not unlike biologically a queen bee leading a hundred working bees).



Martian pathfinders would require a local adaptive and intelligent control because of time-delay of the ground station. Thus, instead of one-to-one, we have proposed a quarterback robot (25% IQ) controlling a team of linemen robots (not unlike a queen bee controlling a hundred working bees). A team of Unmanned Marine Vehicles, Dolphins, can parallel hunt for mines in the extended littoral battle space. Using wireless video feedback control (WaveNet via SINCGARS), a person as the coach may communicate with some delay (because of out of the line of sight) the goal with an intelligent robotic (quarterback of the dolphins team). The quarterback is able to execute locally and faster C3 of all linemen UMV's to identify objects in voting and going around local obstacles, using the prior GPS information from the wide receiver scoutimng UMV.

## 2. LEARNING METHODOLOGY

Recently, Irwin has edited the Industrial Electronics (IE) Handbook (CRC & IEEE Press, pp.1-1686, 1997) and devoted one thousand pages to the intelligent electronics (IE) describing comprehensively all enabling technologies. These are expert systems and neural networks, fuzzy systems and soft computing, evolution systems, computational intelligence, and hybrid applications, and emergent technologies. We believe that some degree of human-like intelligence is necessary for user-friendly interaction with IE. On the other hand, the classical artificial neural networks (ANN) with *supervised* learning strategy have reached the maturity and plateau with some mixed appraisals, although the interdisciplinary studies have just been bearing fruits (since the establishment of international neural network society a decade ago). For example, the neural physiological experiments of human sensors have culminated a truly *unsupervised* learning new paradigm. When a newborn baby faces the bustling and hustling world, he/she cannot grasp the changing signals

$s_i(t) \equiv s_i(t)\mathbf{a}_i$ , from noisy inputs  $x_1(t)$ ,  $x_2(t)$ . However, the intuition is that non-noises must be signals. Thus, the child recognizes the fluctuating noise that has zero correlation  $\langle v_1(t)v_2(t) \rangle_G = 0$  of the neuron outputs.

If one assumes a linear instantaneous inputs as

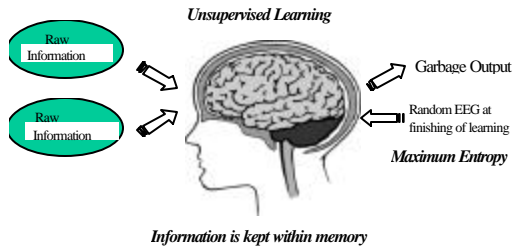
$$\mathbf{X}(t) = s_1(t)\mathbf{a}_1 + s_2(t)\mathbf{a}_2 \equiv [\mathbf{A}]\mathbf{S}(t) \quad (1)$$

where  $\mathbf{S}(t)$  and  $[\mathbf{a}_1, \mathbf{a}_2] \equiv [\mathbf{A}]$  are unknowns, then the artificial neural network (ANN) seeks a weighted sum:  $\mathbf{V}(t) \approx [\mathbf{W}]\mathbf{X}(t)$  which will produce garbage outputs defined by

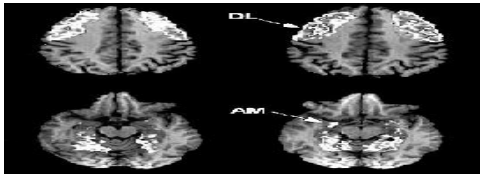
$$\langle \mathbf{V}(t)\mathbf{V}(t)^T \rangle_G = [\mathbf{I}] \approx [\mathbf{W}][\mathbf{A}] \langle \mathbf{S}(t)\mathbf{S}(t)^T \rangle [\mathbf{A}]^T [\mathbf{W}]^T$$

which implies  $[\mathbf{W}][\mathbf{A}] \approx [\mathbf{I}] \quad (2)$

because  $\langle \mathbf{S}(t)\mathbf{S}(t)^T \rangle \approx [\mathbf{I}]$  has a nontrivial higher order statistics (HOS). Thus, the internal knowledge  $[\mathbf{W}]$  is discovered as  $[\mathbf{A}]^{-1}$ . This math is called the Independent (i.e. joint probability density factorization) Component Analyses (ICA). For instance, after the external stimulus by light, sound, and perhaps touch sensations, one hundred millions of visual, hearing and tactile sensory neurons generate highly redundant collective excitations, which can not and should not sustain themselves. Local time scale complex nonlinear dynamics will always yield to decaying in the global time scale, according to Neurodynamics Lyapunov-like Theorem, proved in Sect. 6. Unsupervised learning takes the advantage of the necessary decay of those highly redundant excitations, as the mean of memory toward statistically independent components (IC) without knowing precisely what they are. Therefore, this output state can not be specified ahead of the learning in the truly unsupervised fashion. After a decade studies of neural nets, we have realized that the chief biological reason for a pair of sensors, eyes, ears, tongue sides, nose holes, hands, is to provide the robustness redundancy and the instantaneous spatial temporal de-noise without teacher together with a simultaneous recognition with teacher (associative memory). To simplifying the unsupervised portion of learning, two ears disagree must not be signal--a perfect de-noise algorithm (called mathematically Independent Component Analyses (ICA)), i.e. "pair of raw inputs, garbage output" as opposed to a dumb PC: "garbage in, garbage out". Since the output is garbage, no teacher is needed, and what's kept inside the brain without being squeezed out as noise is useful feature **Fig. 1**. Neuro Paradigm



Brain imaging experiments might support the hypothesis that learning of a newborn child might have various stages interwoven at ease. Advanced brain imaging, e.g. Functional Magnetic Resonance Imaging (fMRI) or Positron Emission Tomography (PET), can help substantiate the unsupervised learning. For example, Positron Emission Tomography (PET) image reveals all female subjects using both side of brains while all male subjects use only one-side to processing speech. That radioactive-labeled glucose supplies the nutrition needed in the brain processing can decay position rays, which, furthermore, decay into two opposite Gamma rays onto all around films. Image is synthesized like the Tomography, but the PET radiation comes from selectively inside rather than indiscriminately in traditional Tomography.



Should the collective neural activity randomizes into the de-correlation, as stated, the intensity of positions and associated 2 gamma rays that have collectively lit up the brain imaging would fade away as noisy-like. To carry the thought experiment further, we consider instead adults an infant just born. We conjecture that, without yet any internal knowledge representation, a newborn baby, who does not have anything pain/pleasure/movement etc. to be associated with, must utilize sensory input de-correlation process toward noisy output, as a truly unsupervised learning strategy to build up the internal knowledge representation. Let us imagine that the first sight of a face of mother composed of several millions of collective excitations in human visual systems (6.1 millions color perception cones and 150 millions dark light rods on retina). This first impression must be decaying toward randomizing responses as lacking of any imagination ability of association the infant can not sustain another concept or image as the only reference point for feature extraction stoppage criterion. This innate ability could be the foundation for any artificial endowment of machine IQ. Afterwards, the traditional ANN

supervised learning can be leveraged by the existence of internal knowledge representation. In this sense, the supervised learning may be called the second stage of learning. This unsupervised learning ability is amble demonstrated in blind de-mixing acoustic signals and images without knowing what the original sources are, and how they are mixed as an infant in a cradle.. This trait is mathematically referred to be ICA or BSS. The result is similar to the cocktail party effect that one can de-mix, during a noisy drinking party, the signals and detect one's name or other important messages among cross talks. There are three stages of learning as a new borne baby develops. The initial stage is related to the decay of collective excitation generated by millions of sensors (6.2 millions color perception cones and 150 millions gray-scale rods) connected to millions of neurons in the cortex area.

1. Initial Stage no details (< 1 week/month old)  
The first stage happens at the limiting case of eye-opening first sight without knowing the desired association memory and without acquiring yet any internal knowledge representation. This is characterized by the merely decay of collective sensory excitations to noise. "Mar, Bar, Dog, Cat, pleasure & pain, etc." learned without feature definition not yet having meaning of association; "(sensors) info inputs = garbage output (randomized EEG when CNS learning stopped)", namely the unsupervised learning by maximizing the output Entropy. (as if lemonade has been squeezed and kept in the synaptic junctions and the useless skin & junk is throw out).
2. The second stage is efficient by leveraging the IC knowledge as the desired output Intermediate State supervised learning sensory inputs--desired outputs forming associated pairs.
3. The third stage may be called the creative sate of thinking (< K12). The third stage takes the advantage of both the internal knowledge being internally generated as realistic excitations, called active imagination, together with the externally generated sensory excitation (to seek again by the unsupervised as new knowledge base). The external sensory inputs plus internal imagination inputs generate new thought.

These stages are alternatively going on effortlessly, and may become not separable subsequently. However, this initial condition of learning should help robots with intelligent sensory processing capability to deal with new environment. In real world applications, we often

lack the precise knowledge of the desired output features. Therefore, we cannot apply the supervised ANN to associate the input data to the output feature. However, without knowing the desired output, the new unsupervised ANN can extract the independent components of the input data, which itself becomes the desired feature.

In summary, when a raw information input through pairs of eyes, ears, nasal passages, after been sieving through the synaptic weight matrix for extracting IC features, the final neural outputs become noise-like. The ANN unsupervised learning changes the ANN weight matrix to sieve or squeeze anything useful (higher order correlation information) from the input sequence until the outputs are left with (nothing but maximum entropy) redundant garbage or noise. This strategy is on the contrary to the supervised one because in a truly unsupervised learning we cannot assume any output goal but the garbage-output for information-input. In paraphrase, after squeezing the juicy clean, ANN throws away the trash. It does not matter whether the input is an orange or a lemon, the end product of the unsupervised process is identical, being without a teacher the only logical choice is trash output meaning no more useful covariance information, i.e. the unique noise-like output state. While a traditional computer has a motto to describe a dumb and do-nothing computers as “garbage-in, garbage-out”, we dramatize that a smart neurocomputer has a motto that “raw information-in, garbage-out”. Such a novel “information-input and garbage/noise output” paradigm for the neurocomputer has accomplished a machine I/Q, which is surely higher than a traditional computer with the do-nothing motto. The new generation or 6<sup>th</sup> gen neurocomputer can at least do a sensor feature extraction job without supervision.

## 2. UNSUPERVISED LEARNING

We present simple mathematical models of unsupervised learning algorithms of artificial neural networks (ANN) as motivated by the biological principle of redundancy reduction (Barrow, 1953) via statistical decorrelation of sensory mixtures, known in ANN as Blind Source Separation (BSS) or Independent Component Analysis (ICA). Different stages of learning are conjectured which could help robots acquire additional knowledge needed in a hazardous and new environment. We begin with the familiar formalism of auto-regression (AR) which is easily generalized to the supervised back-error-propagation ANN, and then to the unsupervised sensory mixture decorrelation. This is initially based on higher orders of statistics, e.g. 4<sup>th</sup> order

cumulant-Kurtosis, that is, furthermore, led to the maximum entropy of all cumulants. These generalizations have been illustrated with computer simulations in a controlled setup. Real world applications are given in Part II, such as remote sensing subpixel composition, voice-dictation phoneme segmentation by means of ICA de-hyphenation, and cable TV bandwidth enhancement by simultaneously mixing all Sport and movie entertainment events.

Such a truly unsupervised learning strategy in terms of ANN is mathematically elucidated in terms of a pair of sensory inputs vector  $\mathbf{X}(t)$ . Assume a linear piecewise-stationary mixture model. The unknown but piecewise time-independent feature matrix  $[A]$  consists of column feature vectors  $[\mathbf{a}_1, \mathbf{a}_2]$  and the correspondingly unknown vector source  $\mathbf{S}(t)$  corresponds to the percentage of feature vector composition, then

$$\mathbf{X}(t) = [A] \mathbf{S}(t) \quad (1)$$

Through ANN feedback iteration unsupervised learning of the synaptic weight matrix  $[W]$ . The bipolar unitary output may be approximated at the maximum entropy (cf. Appendix A Proof)

$$\mathbf{V}(t) = \tanh([W]\mathbf{X}(t)) \approx [W]\mathbf{X}(t), \quad (2)$$

which has a linear slope at the threshold value of input for “may-be-yes may-be-no” no-information answer (proved to be at the maximum Shannon entropy with the minimum of information; rather than, the definite yes-or-no informative answers at the large input bipolar values). Instead of tanh one can also adopt the positive sigmoid function, whose outputs must be subtracted by the averaged value to be bipolar fluctuations of mean-zero. In any case, the output is not the traditional desired output, but must be reduced to be noise-like at the end of unsupervised learning process in consistent with the maximum entropy with no more output information at the second moment level. The off-diagonal random components on the time average vanish, while the diagonal elements are identically squared and never vanish and can be normalized to one.

$$\langle \mathbf{V}(t)\mathbf{V}^T(t) \rangle = [I] \quad (3)$$

To achieve it, a random permutation of a large block of data is often recommended to avoid the sampling inaccuracy. (The time order scrambled for fear of sampling error will be preserved in data  $\mathbf{X}(t)$ , once we have found  $[W]$  as the inverse matrix of  $[A]$ )

Specifically, the whitening of the second moment of the output shows:

$$\langle \mathbf{V}(t)\mathbf{V}^T(t) \rangle = [W][A] \langle \mathbf{s}(t) \mathbf{s}^T(t) \rangle [A]^T [W]^T = [I] \quad (4)$$

This is equivalent to  $[W] = [A]^{-1}$  provide that statistical de-correlation of sources

$$\langle \mathbf{s}(t) \mathbf{s}^T(t) \rangle = [\mathbf{I}] \quad (5)$$

is true. If not, a whitening in the data domain is needed to get rid of the second order statistics, namely eliminating Gaussian random process and keeping high order information. In electrical engineering, this operation is known as the gain normalization of different sensor inputs.  $\mathbf{U} = [\mathbf{Wz}] \mathbf{x}$  where the zero-phase or symmetric matrix [BS95]

$$[\mathbf{Wz}] = [\mathbf{Wz}]^T = \langle \mathbf{x} \mathbf{x}^T \rangle^{-1/2} \quad (6)$$

can be derived by setting

$$\langle \mathbf{U} \mathbf{U}^T \rangle = [\mathbf{Wz}] \langle \mathbf{x} \mathbf{x}^T \rangle [\mathbf{Wz}]^T = [\mathbf{I}],$$

and post-multiply with  $[\mathbf{Wz}]$ :  $[\mathbf{Wz}] = [\mathbf{I}] [\mathbf{Wz}] = [\mathbf{Wz}] \langle \mathbf{x} \mathbf{x}^T \rangle [\mathbf{Wz}]^T [\mathbf{Wz}]$ , and canceling the common factor  $[\mathbf{Wz}]$ :  $\langle \mathbf{x} \mathbf{x}^T \rangle [\mathbf{Wz}]^T [\mathbf{Wz}] = [\mathbf{I}]$ , where use is made of the symmetry to arrive at the result of inverse square-root of covariance matrix. Since  $[\mathbf{W}]$  is the statistical inverse of  $[\mathbf{A}]$ , one can use it to obtain the unknown source point-by-point in the identical time order

$$[\mathbf{W}]\mathbf{X}(t) = [\mathbf{W}][\mathbf{A}]\mathbf{S}(t) = \mathbf{S}(t) \quad (7)$$

While an ill-posed deterministic problem can not be uniquely solved, an ill-posed statistical problem has a lot more conditions in time to determine all those unknowns.

Deterministic: # of unknowns,  $\mathbf{S}$ ,  $\mathbf{A}$  > # of known  $\mathbf{X}$

Stochastic computing the covariance  $\mathbf{R}$

# of unknowns = # known; data  $\mathbf{R}_{xx}$  = source  $\mathbf{R}_{ss}$ ,

But if more than 2 sensors & de-correlated

$$\mathbf{R}_{ss} = \mathbf{R}_{ss} \hat{\delta}_{ss}$$

then it's possible to gain more  $\mathbf{R}_{x_1 x_1}$ ,  $\mathbf{R}_{x_2 x_2}$ ,  $\mathbf{R}_{x_1 x_2}$

This is not unlike the human experience, which by definition provides the statistics determination based on past experience. Real world applications are given in Part II, such as remote sensing subpixel composition, voice-dictation phoneme segmentation by means of ICA de-hyphenation, and cable TV bandwidth enhancement by simultaneously mixing all Sport and movie entertainment events.

The visual cortical feature detectors might be the end result of such a Redundancy Reduction Process (RRP), in which the activation of each feature detector is supported to be as statistically independent from the others as possible. Such as 'factorial code (of joint probability density)' potentially involves independence of all orders, but most studies have used only the second-order statistics required for de-correlating the outputs of a set of feature detectors. Field has observed that the early learning algorithms are mainly based on the second-order statistics, which might account for the missing opportunity. Current understanding is that the need of high order statistics such as the 4<sup>th</sup> order cumulant called Kurtosis may be captured

completely by the information-Theoretical approach of maximum mutual information entropy underlying the Independent Component Analysis (ICA). The fourth cumulant, the Kurtosis  $K(\mathbf{u})$ , is often used by Helsinki's Oja group to seek the statistical matrix inversion.

$$K(\mathbf{V}) = \langle \mathbf{V}^4 \rangle - 3(\langle \mathbf{V}^2 \rangle)^2 \quad (8)$$

because  $\langle v_1 v_2 v_3 v_4 \rangle_G = \langle v_1 v_2 \rangle_G \langle v_3 v_4 \rangle_G + \langle v_1 v_3 \rangle_G \langle v_2 v_4 \rangle_G + \langle v_1 v_4 \rangle_G \langle v_2 v_3 \rangle_G$  are reduced for identical process to (8). One considers a single weight vector update:

$$d\mathbf{w}/dt = dK/d\mathbf{w}. \quad (9)$$

The other weight vectors are found by the projection pursuits.

If each voice and image has its unique value of Kurtosis, then seeking a stationary Kurtosis yields the specific voice and image, without knowing what is the desired output: Gaussian, Laplacian, Multi-Modal Distribution & each has a Super-Gaussian,  $K > 0$ , or Sub-Gaussian.  $K < 0$ , Kurtosis value than Gaussian,  $K = 0$ . Note that speech oscillation has Laplace distribution decaying exponentially from the mean value, zero amplitude, which is faster than Gaussian quadratic decay. Therefore, the Kurtosis of speech is called super-Gaussian and by definition, has a positive value (subtraction of smaller variance than that of Gaussian). On the contrary, an image histogram has a bimodal distribution for most grey scale images, then the variance is bigger than that of Gaussian. Therefore, an image has a negative Kurtosis value, so-called the sub-Gaussian.

Imagery edges occur naturally in human visual systems as a consequence of redundancy reduction towards "sparse & orthogonality feature maps," which have been recently derived from the maximum entropy information-theoretical first principle of artificial neural networks. Singularity edge-maps are sparse and orthogonal for the uniqueness & robust features necessary for pattern recognition tasks. Sparseness of singularity edge map needs more than second order statistics the ICA to extract it.

That decay of excitation patterns towards noisy outputs results in the stored memory eight matrix  $[\mathbf{W}]$  among neurons. From the knowledge representation point of view, the more efficient and robust representation, the better. Two principles are the keys to achieve efficient representation: *orthogonality* and *sparseness* in the hits frequency of feature detectors leading to unique identification. For example, an edge-map with one's over zero background is clearly sparse, local, and almost orthogonal. The IC notion may be attributed first to Barrow in 1953 as the redundancy reduction process (RRP). In fact, the final IC State may be described by a factorized

joint-probability density function, and is sometimes called as factorized code. This factor-code corresponds to all sparse orthogonal edge maps in the early vision processing. The second moment of IC must be by definition a diagonal matrix, which appears like a Gaussian random process whose off-diagonal elements cancel one another. If the learning of weight matrix [W] can achieves the maximum entropy  $H(\mathbf{V})$  of the output  $\mathbf{V}$  or the linear slope portion of the maximum entropy sigmoidal neuron output  $H(\mathbf{V}) @ H(\mathbf{s}(\mathbf{U})) @ H(\mathbf{U})$  which implies that all nth moments of the ANN output components  $\mathbf{U}=\{u_1, u_2\}$  of two sensor neurons are independent in terms of the normalized statistical histograms  $\mathbf{r}(\mathbf{u})$  defined as: joint p.d.f.

$$\rho(x_1, x_2) = \rho(x_1)\rho(x_2)(1 + h(x_1, x_2))$$

Factorized Code [Ati92] implies:

$$E\{x_1^n, x_2^m\} = \int \int x_1^n x_2^m \rho(x_1, x_2) dx_1 dx_2 = \int x_1^n \rho(x_1) dx_1 \int x_2^m \rho(x_2) dx_2 = E\{x_1^n\} E\{x_2^m\}$$

Example: Gauss Center of Limiting Theorem

$$\rho(\xi) = \exp(-(\xi - \langle \xi \rangle)^2 / 2\sigma^2);$$

$$\langle \xi^4 \rangle = \langle \xi^2 \rangle^2, \langle \xi^2 \rangle = 0, \text{ if Gaussissn iid}$$

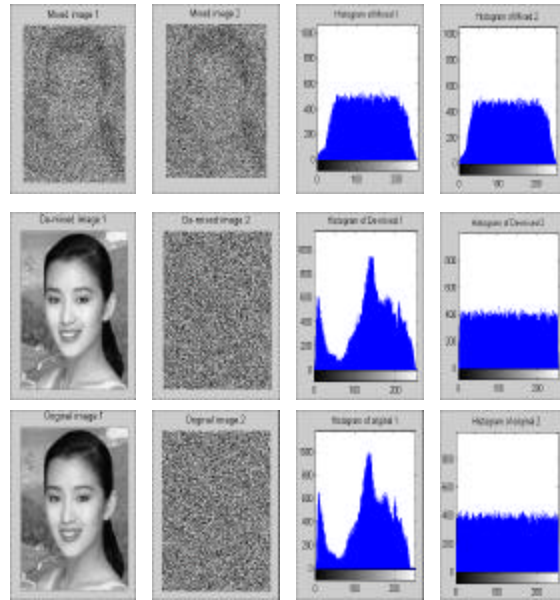
where

$$\langle u^n \rangle = \int u^n \mathbf{r}(u) du = E\{u^n\}$$

Biological evidence is first due to Nobel Laureats Hubel and Wiesel showing an oriented edge-map in the several octave scale in cats, similar to 2-D oriented Gabor Logon (information unit similar to a windowed FT or a WT without affine parameterization). Such an unsupervised learning methodology has been given in solving the statistically Blind Source Separation (BSS), as first introduced by C. Jutten, J. Herault.

$[w] = [ [I] + [s] ]^{-1}$  where  $[s'] = [s] + \mathbf{a} f(y)g(x)$  is an odd function for Blind Source Separation (BSS) of Super-Gaussian Laplacian distribution of speeches  $\Delta[W]=g(\mathbf{x})\tanh(\mathbf{u}^T)$  in terms of some ad hoc odd functions, in the first Snowbird ANN Conference in 1986. Both ICA subsequently coined by P. Comon [Com94], and BSS further elaborated by Herault & Jutten were appeared in Signal Processing Journal in 1991. Oja elaborated the nonlinear PCA learning, because neuron output  $\mathbf{V} = \tanh(\mathbf{U}) = \mathbf{U} - 2/3 \mathbf{U}^3 + \dots$  has a similar Taylor expansion as  $dK(\mathbf{V})/d\mathbf{w}$ . The first principle of ICA may have several forms, e.g. absolute entropy versus mutual entropy, Neg-entropy-- the distance from the normality, Edgeworth versus Gram-Charlier expansions (of pdf in terms of moments) which are related to the maximum Shannon entropy  $H(\mathbf{V})$ . The essential portion related to the change of weight matrix is equivalent in achieving the redundancy reduction toward independent components which gives rise naturally to a sparse

orthogonal edge map (unfortunately only at one wavelet resolution). The landmark accomplishment of ICA is to obtain, by unsupervised learning algorithm, the edge-map as image feature  $\vec{a}$ , shown by Helsinki researchers using fourth order statistics of  $\mathbf{V}$  -- Kurtosis  $K(\mathbf{V})$ , and derived from information-theoretical first principle of ICA by Bell & Sejnowski. Amari has further contributed to the speedup of learning by suggesting a natural gradient descent, rather than the original entropy gradient involving a non-local weight matrix inversion.



**Fig. 4** Why do we have two eyes? They can provide instantaneous spatial learning without teacher, i.e. two eyes agree must be signal, and don't noise. A perfect denoise is possible using two eyes or two ears, two sides of tongue and two nose passages, two hands, etc., but one sensor needs the slower help from the brain memory itself. Sophisticate cross-talk de-mixing is in Sect. 5.

#### 4. CYBERNETIC THEORY

Human intelligence can not yet be mathematically defined and addressed here. Instead, the supreme manifesto of the human intelligence might be the learning ability without teachers, which is modeled by the thermodynamics neural net learning theory. In so doing, we have discovered that the parallelism between the supervised associative recognition and the unsupervised ICA de-noise [1,2] (e.g. the cocktail party effect) is conveniently controlled by the Gibb's free energy temperature [3]. In this paper, we identify the temperature as the cybernetic temperature defined as a root-mean-square fluctuation of synaptic transmission activity.

During the last Russian Academy of Nonlinear Sciences Academician meeting (at St. Petersburg June 1999), I have proposed the role of cybernetic temperature for learning capability as warm blooded man, mammals, versus cold blooded dragons, reptile, lizards. Furthermore, I have investigated the information content of an unsupervised learning by means of Independent component analyses (ICA) with several students (cf. Wavelet Applications Orlando, April 2000 proceedings). This is based on joint probability density factorization for all independent moments (cf. Shannon's Principal Component Analyses (PCA) information content based on the second moment covariance only). Application to two eye de-noise, and image blind de-mixing and seven spectral band remote sensing are respectively given in [3,4,5].

Those who know of the animal intelligence having very little to do with brain sizes will be critical about the homeostasis theory having anything to do with the intelligence. Mathematically, neural network models of both learning seem to predict a constant temperature  $T$  for the minimization of the thermodynamic Helmholtz free energy,  $A = U - TS$  (equivalent ANN notation Lyapunov  $L = E - T H$ ), in order to achieve the synergistic learning balanced between the supervised energy,  $E(v_i)$ , Hopfield-like minimization of neuron firing rate  $v_i$ , and the recently breakthrough of the unsupervised sensory pre-processing based on the output entropy  $H(w_{ij})$  Bell-Sejnowski maximization. Since some mammals have bigger brains than Einstein's having a normal human being size, it implies not the brain size rather the interconnectivity wrinkles of the gray matter, which are responsible for associative memory. Again, it is not the degree of temperature rather the constancy of brain body temperature, which may be important to the kinetic diffusion rate controlling the chemical reactions that are vital to the healthy cellular functions (as evident in the excess fever causing by fatal diseases).

First, we define the supervised learning to include self-taught (involving higher motivation and intelligence) considered being equivalent to learning with a teacher either implicitly internally or explicitly externally. Secondly, we define the unsupervised learning to be the pre-attentive pre-processing of all real-time and short-term memory pairs of sensory inputs without conscience effort with associative recall. Neural network learning models suggest a constant brain cybernetic temperature, which balances the output energy  $E(v_i)$  for neuron firing rates for supervised

learning, and the maximum entropy  $H(w_{ij})$  of synaptic matrix for unsupervised excitation decays for redundancy reduction (to wavelet or singularity-map). While the supervised learning (with implicit or explicit teaching) may be driven by the internal energy minimization, the unsupervised learning (sensory pre-processing) may be driven by the relaxation decaying processes by means of the maximization of local entropy. For example, there are 6.1 millions cones for color vision and 150 millions rods for dark light vision, and any imbalance on the visual neural pathway might cause the hallucination. This balance is achieved by the thermodynamics  $L = E - T H$  at a constant temperature  $T$  to determine the internal energy  $E(v_i)$  minimization and the entropy  $H(w_{ij})$  maximization. According to the theory of statistical mechanics, such a thermodynamic balance between  $E$  &  $H$  is possible due to a constant temperature  $T$ . This natural thermodynamic equilibrium might be useful to help develop fully the innate learning ability of a mammal. Thus, it is conjectured that the homeostasis of mammals must have a profound effect upon learning ability of the mammals, which in turn affect the development of intellectual capability. The cold blood reptiles can obviously learn without such a thermodynamic equilibrium, and it is interesting to notice a lower intelligence associated with them.

These models suggest it may also be important to the intercellular communication-mediated learning mechanism. Furthermore, based on recent breakthrough of sensory learning, the minimization of Helmholtz free energy  $L \equiv E - TH$  at a constant  $T$  involves the internal energy  $E$  and the entropy  $H$  is believed to have maintained the thermodynamic equilibrium of those intercellular communication mechanisms useful for Hebbian synaptic modification. This free-energy minimization is mathematically shown to be Lyapunov functions that control a proper balance between the unsupervised sensory preprocessing based on maximum entropy of synaptic weights and the supervised learning based on minimization of neuron firing rate energy.

## 5. INFO-THEORETICAL MODEL

Recently, the biological edge map developed by the Nobel laureates, Hubel-Wiesel, was reproduced computationally by maximizing the neuron output entropy of among  $10^4$  images by means of maximum output entropy:

$$\partial H(\mathbf{V}) / \partial [\mathbf{W}] = \partial [\mathbf{W}] / \partial t. \quad (10)$$

Algorithmically, ANN adjusts  $[\mathbf{W}]$  at the linear output range,  $\mathbf{V}(t) = \tanh([\mathbf{W}]\mathbf{X}(t)) \approx [\mathbf{W}]\mathbf{X}(t)$ , so that  $\langle \mathbf{V}(t) \mathbf{V}(t)^T \rangle_G = [\mathbf{I}]$ . Note that no decision of the sign of the tanh function is necessary in the

linear range, implying that the maximal output entropy, and thus the input information is kept in [W]. There is a need to unify both the supervised learning of Principal Component Analyses (PCA) by Oja et al. and the unsupervised learning of ICA (advanced by Jutten & Herault, Comon (1991), Cardoso (1998) in France and by Bell-Sejnowski (1995) in U.S.A., and by Amari-Cichocki (1996) in Japan).

$\begin{aligned} \text{Bayesian prob } f(x,y) &= f(x y)f(y) = f(y x)f(x) \\ \text{Shannon Entropy } H(x,y) &= - \langle \text{Ln } f(x,y) \rangle \\ &= - \langle \text{Ln } f(x y)f(y) \rangle = H(x y) + H(y) \\ &= - \langle \text{Ln } f(y x)f(x) \rangle = H(y x) + H(x) \\ &= H(x) + H(y) - I(x,y) \\ \text{Mutual Info } I(x,y) &= \langle \text{Ln } [f(x,y)/f(x)f(y)] \rangle \\ &= \langle \text{Ln}[f(x y)f(y)/f(x)f(y)] \rangle = -H(x y) + H(x) \end{aligned}$
---

<p>Statistical information content similar to geometrical information content of PCA</p>
--

The associative recall by the associative memory outer-product approach can determine the center of training set clustered around each ICA basis, (11), and only 30% of them are significant in the direction cosine sense, and the rest ICA bases have no significant alignment with the training set. This is similar to PCA eigenvalues, which fall off drastically after the principal components, and is called by Shannon as the degree of freedom of the information content. We have generalized information content to statistical information content for those non trivial ICA bases.

$$\text{Define } [W] = \sum_i (\vec{W}_i \quad \vec{W}_i^T); \quad (12)$$

A fast estimation of the principal information content of a normalized ICA basis is denoted similarly by the eigenvalue  $\lambda_k$  that sums the squared magnitude of all the projection of normalized training data  $X_i$  upon k basis :

$$I_k = \frac{1}{M} \sum_{i=1}^M (\vec{x}_i, \vec{W}_k)^2 \quad (13)$$

### 6. LYAPONOV CONVERGENCE PROOF

Szu has postulated the Helmholtz free energy [3]

$L(v_1, \dots, v_n, w_1, \dots, w_n) = E(v_1, \dots, v_n) - T H(w_1, \dots, w_n)$  as the Lyapunov function, and proves the convergence  $dL/dt < 0$  of both supervised energy-E-minimization and unsupervised entropy-H-maximization dynamics in synergism: Given local gradients:

Min. energy:  $du_i/dt_i = - \partial E / \partial v_i$   
 Max. entropy:  $(\partial[w_i]/\partial t_i) = (\partial H / \partial [w_i]).$

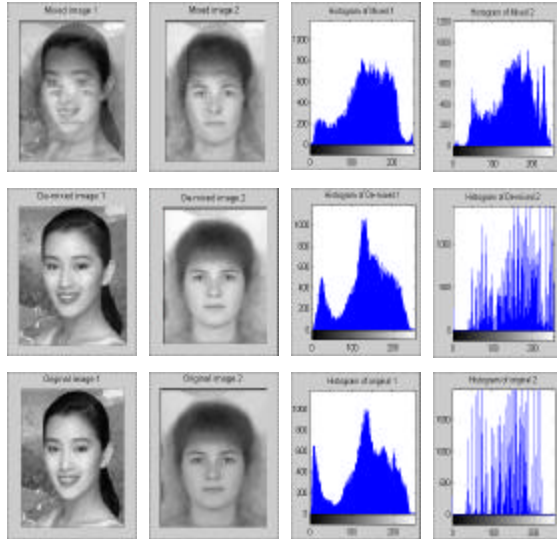
**Proof:**

Min. Lyapunov (namely Helmholtz free energy):

$$\begin{aligned} dL/dt &= \sum_i (\partial E / \partial v_i) (\partial v_i / \partial u_i) (\partial u_i / \partial t_i) (dt_i / dt) - T \sum_i (\partial H / \partial w_i) (\partial w_i / \partial t_i) (dt_i / dt) \\ &= - \{ \sum_i (\partial E / \partial v_i)^2 (\partial v_i / \partial u_i) + T \sum_i (\partial H / \partial w_i)^2 \} (dt_i / dt) < 0 \end{aligned}$$

**Q.E.D.**

Here use is only made of stable cybernetic temperature T and the local gradient dynamic equations and positive firing rates to eliminate the temporal derivatives by spatial derivatives to form two real quadratic expressions, which by definition are always positive.



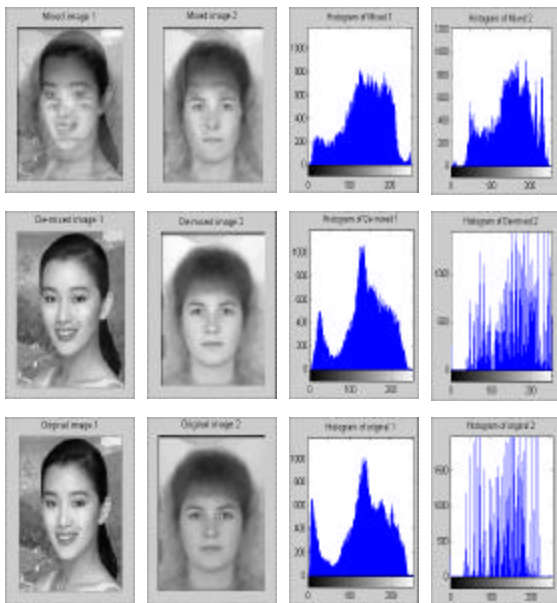
### 7 CONCLUSION

Helmholtz-Lyapunov drives the punctuated evolution for brain open systems at constant temperatures as opposed to less intelligent cold blood animals. One must go beyond the least mean square (LMS) error energy, and apply HOS to ANN. Applications are possible to multi-medium computers & machine intelligence.

### 8. REFERENCES

- 1 H. H. Szu, I. Kopriva, A. Peršin. Independent component analysis approach to resolve the multi-source limitation of the nutating rising-sun reticle based optical trackers, Optics Communications, Vol. 176, Issue 1-3, pp. 77-89, 2000
- 2 I. Kopriva, H. Szu, "Blind Discrimination of the Coherent Optical Sources by Using Reticle Based Optical Trackers is a Nonlinear ICA Problem", Proc. of the 2nd Int. Workshop on Independent Component Analysis and Blind Signal Separation, ed. P. Pajunen and J. Karhunen, June 19-22, 2000, Helsinki, Finland, pp. 51-56.
- 3 H. Szu, "Thermodynamics Energy for both supervised and unsupervised learning neural nets at a constant temperature," Int'l J. Neural Sys. Vol.9, pp. 175-186, June 1999.

- 4 H. Szu, "progresses in unsupervised artificial neural networks of blind image demixing, " New tech of IEEE Ind. Elec. Soc. Newsletter, pp. 7-12, June 1999.
- 5 H. Szu, "ICA-an enabling tech for Intelligent Sensory Processing", IEEE Circuits and Systems Newsletters December of 1999.





# Properties of Learning Knowledge-Based Controllers

Edward Grant

Associate Professor

Director of the Center for Robotics and  
Intelligent Machines

Electrical and Computer Engineering  
Department

North Carolina State University  
Raleigh, NC 27695

Gordon K. Lee

Professor

Assistant Dean for Research, College  
of Engineering

Mechanical and Aerospace Engineering  
Department

North Carolina State University  
Raleigh, NC 27695

## ABSTRACT

This paper addresses the field of knowledge-based systems, and in particular the sub-field of knowledge-based control systems. The rule-based approach used here, particularly in its machine learning or rule induction mode, continues as a major theme in the emerging field of data mining - the extraction of usable insights from large databases.

**KEYWORDS:** *knowledge-based control, learning control*

## 1. INTRODUCTION

The core tenet of this paper is that rule frameworks (i.e., directly programmed rule bases, rule bases derived by rule induction over experimental data and rule bases derived from induction over data produced by rule-based qualitative modeling with rule-based simulation) can be applied to achieve successful control of diverse systems.

The results obtained show that the underlying goals of the knowledge-based approach are as valid as ever and are particularly relevant to many of today's critical applications. In certain specific areas, they remain superior to all others. These areas include: (1) the inherent ability of knowledge-based systems to make their operation transparent to computer experts, to subject domain experts and to their users - a consequence of the systems' representing their knowledge directly in the form of symbolic rules; (2) the ability of the technology to capture the knowledge of the best experts in the field, to refine consistent and understandable symbolic rules from cases and empirical datasets; and (3) the ability to generalize such rules to cover a much

larger set of possibilities than can feasibly be detailed explicitly by the domain expert or empirical dataset by using knowledge-based simulation, which is important in the field of diagnostic systems.

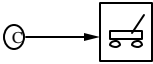
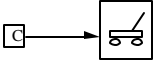
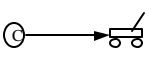
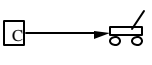
This, at its highest level, is what is demonstrated in this study. The authors methodology for designing knowledge-based (K-B) systems will be described, along with descriptions of its application to systems which can be to produce designs that proved successful in practice.

## 2. DEVELOPING KNOWLEDGE-BASED CONTROL

The focus of this paper is on mechanisms and technologies for implementing machine intelligence. Nevertheless the ability to learn must be one criterion for describing intelligent behavior. In robotics terms, intelligence is the ability of a machine to act autonomously in the presence of uncertainty. The ability of a robot to adjust its actions based on sensed information [1, 2, 3, 6, 12, 13, 14, 15] is another prerequisite for intelligence. In this work, the actions taken by the machine are considered to be intelligent if the actions reflect the action that a human would take, given the same conditions.

In advanced robotics systems [1, 2, 3, 6, 12, 13, 14, 15], robots are equipped with networked sensors: vision, tactile, proximity, speech recognition, voice synthesis, robot controllers, conveyors, vision processing equipment, and computers, an ideal domain for researching into machine intelligence (see Figure 1). However, the interconnection of physical systems, or the task

undertaken by the system, does not make a machine intelligent. Intelligence comes from the manner in which the system is controlled or from the reasoning and decision making that the machine performs. In our terms, “intelligent control” is closely associated with “machine intelligence” [9, 10].

Schematic	Controller Type	MLC Location	Type of Controlled System
	Human	None	Simulator
	Auto	None	Simulator
	Human	None	Physical
	Auto	None	Physical

**Figure 1.** Automatic or Human Control (MLC - machine learned control)

Intelligent control systems must deal with sensor data and task specification and the task-state derived from integrated sensor data. An intelligent-control system must handle information about its own state and also the state of the environment; it must be capable of reasoning under uncertainty. Intelligent control commonly involves the use of both heuristic and algorithmic programming methods.

First we review hierarchically ordered control architectures for intelligent control. After this we concentrate on controlling dynamic systems with a variety of rule-based and machine learned programs. The final section deals with “human-in-the-loop” control as a knowledge-based controller.

### 3. ARCHITECTURES FOR INTELLIGENT CONTROL

Saridis [11] states that intelligent machines require the use of “generalized” control strategies to perform intelligent functions such as the simultaneous utilization of memory, learning or multi-level decision-making in response to “fuzzy” or qualitative commands. His work proposes that intelligent functions can be implemented using “intelligent control”.

Intelligent control combines high-level

decision-making, advanced mathematical modeling, and synthesis techniques of systems theory. These approaches along with linguistic methods attempt to deal with imprecise or incomplete information from which appropriate control actions evolve. The control functions in an intelligent machine have been implemented as a hierarchy of processes [1, 2, 3, 7, 11]. The upper layers concentrate on abstractions, decision-making and planning, while the lower levels concentrate on time-dependant sub-tasks, such as processing data from sensors or operating an actuator. Hierarchical decomposition is applied to complex control problems to reduce them to smaller sub-problems.

In the hierarchical control architectures of Albus [1, 2, 3] and Meystel [7], each layer essentially possesses the same processing nodes. These two architectures include a knowledge-base, sensory processing, task decomposition, and communication. But Saridis [11] recognized that each layer in a hierarchy need not perform the same activity over time and he and Albus [6] recognized that middle layers are frequently hierarchies of linguistic or heuristic decision structures that handle imprecise or “fuzzy” information. The National Institute of Standards and Technology (NIST) implemented Albus’ architecture in manufacturing control (AMRF) [2] and in the NIST/DARPA Multiple Autonomous Undersea Vehicle (MAUV) [1]. Meystel [7] used an autonomous undersea vehicle as a demonstrator and Saridis [11] applied his architecture to space station robot and control applications.

### 4. REINFORCEMENT LEARNING

Since the mid-1970's, artificial intelligence (AI) methods have been continuously developed and applied by industry, business, and commerce. Expert systems are the most successful implementation of AI. However, the difficulties surrounding the development of the production rules for expert systems, going from the general to the specific led to the development of a subdivision of expert system technology known as “machine learning”. In this section, we will look at the how the production rules for “rule-based” control can be produced manually and automatically and we will discuss approaches for achieving machine -learned control (MLC). We will describe a controller based on the machine learning algorithm BOXES [10], an algorithm that

uses a reinforcement learning approach and we will discuss the implementation of neural networks for control. Both reinforcement learning and competitive learning are considered [13].

$$\ddot{q} = \frac{g \sin q - \cos q \left[ \frac{\dot{F} + m_p l \dot{q}^2 \sin q}{m_c + m_p} \right]}{l \left[ \frac{4}{3} - \frac{m_p \cos q}{m_c + m_p} \right]}$$

$$\ddot{x} = \frac{F + m_p l [\dot{q}^2 \sin q - \ddot{q} \cos q]}{m_c + m_p}$$

Figure 2. The Pole and Cart Equations

### Rule-Based Control

Humans are capable of deriving a set of control rules through the process of interpretation. For example, consider the equations for the pole and

- if  $\theta_{dot} > THRESHOLD$   $\theta_{dot}$  then push RIGHT
- if  $\theta_{dot} > -THRESHOLD$   $\theta_{dot}$  then push LEFT
- if  $\theta > THRESHOLD$   $\theta$  then push RIGHT
- if  $\theta < -THRESHOLD$   $\theta$  then push LEFT
- if  $\dot{x} > THRESHOLD$   $\dot{x}$  then push RIGHT
- if  $\dot{x} < -THRESHOLD$   $\dot{x}$  then push LEFT
- if  $x > THRESHOLD$   $x$  then push RIGHT
- if  $x < -THRESHOLD$   $x$  then push LEFT

Figure 3. The Makarovic Rule Derived from Interpreting the Equations of Motion

cart problem (see Figure 2). Makarovic [8] derived a rule by examining the system's differential equations of motion (see Figure 3). The Makarovic rule worked well when the parameters of the system remained constant. When system parameters changed, the Makarovic rule cannot guarantee success. This showed that the arbitrary choice of one set of threshold values is not ideal for a system whose configuration changes. In contrast, a rule derived from observing a physical system's performance [18] can be written without any threshold values placed on the observation. Here the condition part of the rules only deals with the sign of the errors and with the sign of the variations of observed system state variables. This approach reflects human control heuristics and it

can adapt to varying system configurations.

Schematic	Controller Type	MLC Location	Type of Control System
	Human	Off-line	Simulator
	Auto	Off-line	Simulator
	Human	Off-line	Physical
	Auto	Off-line	Physical

Figure 4. Machine-Learned Control (Passive Learning)

### Machine Learned Control

Machine learning as presented here is classified into two areas: (1) artificial-intelligence type learning based on symbolic computation and (2) neural nets. These are chosen because we have first-hand experience of applying them to real-

```

if ((theta(k) > THRESHOLD)
then
  if ((theta(k) < theta(k-1))
    and (theta(k) - theta(k-1) > |theta(k-1) - theta(k-2)|))
  then
    apply a RIGHT force
  else
    apply a LEFT force

if ((theta(k) < -THRESHOLD)
then
  if ((theta(k) > theta(k-1))
    and (theta(k) - theta(k-1) > |theta(k-1) - theta(k-2)|))
  then
    apply a RIGHT force
  else
    apply a LEFT force

if (abs(theta(k)) <= THRESHOLD)
then
  if (x(k) >= 0)
  then
    if ((x(k) < x(k-1)) and (abs(x(k) - x(k-1)) > |x(k-1) - x(k-2)|))
    then
      apply a LEFT force
    else
      apply a RIGHT force
  else
    if (x(k) < 0)
    then
      if ((x(k) > x(k-1)) and (abs(x(k) - x(k-1)) > |x(k-1) - x(k-2)|))
      then
        apply a RIGHT force
      else
        apply a LEFT force

```

Figure 5. A Control Rule Derived from Experimentation with a Pole and Cart Simulator

world applications. An effective machine learning system must use sampled data to generate internal updates and also be capable of explaining its findings in an understandable way, e.g., symbolically. The learning system must also be

able to provide an explanation of its results to a human-expert. The findings should also improve the human expert's understanding and verification. Artificial-intelligence type learning originated from an investigation into the possibility of using decision trees or production rules for concept representation. Since then, the work has extended to the use of decision trees and production rules to handle most conventional data types, including noisy data sets, and as a knowledge acquisition tool (see Figures 4 and 5).

Reinforcement Learning

Reinforcement learning, of the type produced by Michalski [9] and Michie [10], is similar to feedback for adaptation. However, unlike supervised learning, reinforcement feedback learning only gives an indication of the value of the system's action. Reinforcement is a feedback on the correctness of an action; it is not information on what the correct action is. Reinforcement learning is useful in cases where supervisory information is not available (see Figure 6).

Also, reinforcement learning falls into two categories: (1) non-associative type, which only receives a reinforcement signal from the environment and (2) associative reinforcement learning, where the system receives both a reinforcement signal, and sensory information, on the state of the environment. Sensors are used to discriminate between different situations. This we considered more suited to our particular needs with the pole and cart application. We will discuss the

```

theta_dot > 0 : RIGHT
theta_dot <= 0 :
  theta <= -2 : LEFT
  theta > -2 :
    theta_dot <= -1 : LEFT
    theta_dot > -1 :
      x_dot <= -6 :
        theta <= 1 : LEFT
        theta > 1 : RIGHT
      x_dot > -6 :
        x <= 0 : LEFT
        x > 0 : RIGHT

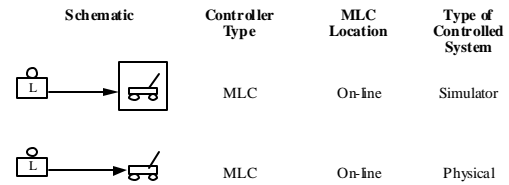
```

**Figure 6.** A Control Rule for the Pole and Cart-Derived Using the BOXES Algorithm

application of rule-based (MLC) and neural network controllers to control a pole-cart system by using AI techniques.

'BOXES'

In Michie and Chamber's learning algorithm 'BOXES' [10, 13], the physical state space is



**Figure 7.** Machine-Learned Control

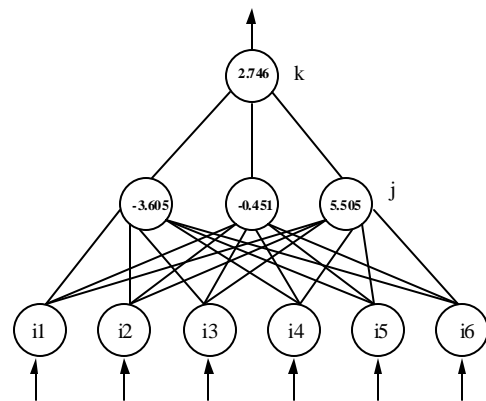
partitioned into boxes. The algorithm learns to set correct decisions for each box through trial-and-error [10, 13]. Unfortunately, state space partitioning prior to experimentation is arbitrary because it is reliant on human knowledge. If the original partitioning is wrong, the algorithm can not learn to correct it. In the following, we will show how our rule can be used to partition the state space in the pole-cart application.

**5. NEURAL NETWORK - BASED REINFORCEMENT LEARNING**

A learning controller consisting of a two-layered neural network was used to implement the input-output transfer function and an evaluation network, a look-up table, which provides the necessary reinforcement signal for evaluative feedback via a goal oriented performance index. The high-level architecture for the teaching controller is given in Figures 7 and 8.

Neural Networks

Neural networks implement information storage with synaptic weights storing information and distributed patterns acting as keys; they combine the benefits of both the computational method and look-up tables. With neural networks,



**Figure 8.** A Neural Network Controller for Controlling a Pole and Cart

Pattern Number	Pattern Features u1 u2 u3 u4 u5 u6	Actual Output	Classification
1	0 0 0 0 0 0	0.006910	0
2	0 0 0 0 0 1	0.003990	0
3	0 0 0 0 1 0	0.002360	0
4	0 0 0 0 1 1	0.001337	0
.	.	.	.
.	.	.	.
32	0 1 1 1 1 1	0.063977	0
33	1 0 0 0 0 0	0.959916	1
34	1 0 0 0 0 1	0.958932	1
.	.	.	.
.	.	.	.
64	1 1 1 1 1 1	0.952255	1
65	0.5 0 0 0 0 0	0.819989	1
66	0.5 0 0 0 0 1	0.616265	1
.	.	.	.
.	.	.	.
96	0.5 1 1 1 1 1	0.824466	1

**Figure 9.** A Rule-Based Table Look Up for Determining Neural Network Control Actions

information about control situations is coded in terms of distributed patterns; hence they can support distributed representation and reduce the storage requirements associated with control surface dimension.

A neural network can specify control actions for a given situation not visited during learning; it specifies according to its similarity. This associated structure automatically generalizes according to degree of similarity. The trade-off between computation time and storage space is resolved using neural networks.

#### Establishing the Look-up Table

In reinforcement learning, at every time step during learning, control actions are evaluated with respect to a sub-goal. The action that maximizes the sub-goal is regarded as the optimal control action and is rewarded; all other actions are punished. In the pole-and-cart, learning is difficult because the effects on choosing different actions cannot be tested. So, here we evaluate alternative control actions with respect to a small region of the state space. We also assume that they have the same reinforcement value (see Figure 9).

#### Decoding the State Variables

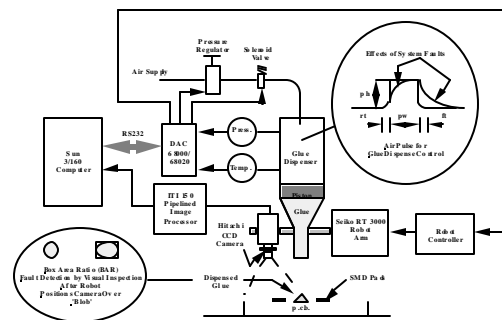
The term “decoder” describes the process of accepting an input situation and transforming it into one activity from a choice of a large number. Hence evaluation signals are stored as a look-up table where an input situation appears as an

activity on a single path-way to a storage location. The storage location contains the appropriate evaluation specification. This approach was motivated by ‘BOXES’ [10]. Here, the four-dimensional state-space is divided into disjoint regions (‘BOXES’) by quantizing the state variables. The evaluation of different control actions was made with respect to a sub-goal. This estimate provided the necessary reinforcement signal for a reinforcement learning neural-network (RLNN) for control [13].

## 6. LOUGHBOROUGH GLUE DISPENSING WORKCELL

After the researchers at Loughborough University of Technology had tried numerous methods to visually inspect a dispensed ‘blob’ (e.g., inspecting for the “blob volume” using striped light) it was found that a simple feedback measure gave acceptable control. Tests using the “blob area” as the measured variable showed that this parameter could keep the process within a desired operating bandwidth. The measured variable based on the blob area, termed “box-area-ratio” (BAR), could distinguish between many of the common faults associated with the process.

Common faults observed in the process include: (1) a blob collapsing; (2) stringy (stitching) attachments; and (3) the presence of entrapped air. The researchers used on-line data recorded from the process to elicit information from that data. The extracted information is then presented as rules. After the process expert had verified these rules, the rules became a “knowledge-based” controller. During this period of interaction with the process expert, an interesting observation was made; it appeared that the expert measured the performance of the process through fault diagnosis (Williams, West,



**Figure 10.** A Schematic of the LUT Glue Dispensing Workcell

and Hinde (1992) [12, 13]). This observation led to the development of a knowledge-based controller, one that was tested on the process (see Figure 10).

Before a knowledge-based controller can produce the rules inherent or embedded in the process data, a model must be established. Such a model produces rules from process data (Shepherd (1992) [12]). The model was constructed in three parts: image capture, feature extraction, and classification (see Figure 11).

The data from the LUT process was first normalized, so that it matched the integer requirement of the rule induction software. Thus, a blob area of 1103 is equivalent to a normalized area of 1.103. Also, since the bubble threshold is set for a 10% increase for the 'blob area', the data are classified as `bub_inc`. When there is more than one fault present in the data, each encountered fault is recorded as an attribute exceeding a threshold. Multiple fault conditions are obtained when a combination of attributes has exceeded their limits and an attribute-class vector is repeated for every exceeded threshold. Every exceeded threshold is represented as new class vector and they are added to the class list. Note that in order to be consistent, these may also combine any original class.

```
IF BAR > threshold THEN
  IF area_diff < bubble_threshold THEN
    IF area outside control limits THEN
      apply rule-based control action
    ELSE
      do nothing
  IF risetime > risetime fault threshold THEN
    flag air
  IF falltime > falltime fault threshold THEN
    flag pulse width and pulse height
```

**Figure 11.** Pseudo-code of Process Operators  
Control Rule

Two knowledge-based controllers were tested on the LUT adhesive dispensing process: (1) a controller based on operator derived rules alone, and (2) a controller based on the VACLS rule induction algorithm (control via the fault detection rules derived from the process data). A simple BANG\_BANG controller was written in C; this was used to maintain the dispensed blob area within 5% of a target area of 30,000 pixels. The results were remarkable in that the knowledge-based controller using the VACLS rule induction algorithm was highly successful in dealing with this application environment that is inherently difficult to control, and where knowledge

elucidated from the expert human operator proved crucial, when combined with rule induction over empirical data from the multiple sensors. These results reinforced the value of human-in-the-loop type controllers whereby information from the human along with results from simple experimentation improves system performance.

## 6. CONCLUSIONS

Knowledge-based controllers (e.g., those constructed using expert verified rules) were tested on the various dynamic systems including the LUT industrial process control system. The control experiments tested overall system performance based on data from dynamic system and process parameters. To extract the knowledge-based control rules experiments were conducted on simulators and physical systems. Human control with simulators is achievable, but difficult with physical plant with fast response times. Passive learning proved useful but machine learned control had limitations, particularly when used with physical systems. In terms of the LUT process this included the variation of area and measured and programmed pulse height variation.

Two factors that have to be taken into account when using rule induction algorithms are timing, e.g., what are the overheads associated with implementing rules, and clashes. These influence of these two factors is reduced if the following procedure is adopted when preparing data prior to submitting it to the algorithm being used. First, divide the data set into two and train the algorithm on one half of the data set. Second, build the rule-based controller and test its performance on the other half of the data set. Third, after refining the controller, install it into the process and obtain test results. This was the procedure adopted when working with VALCS and it produced the richest amount of information.

Although this amount of information may not be wholly necessary for controlling a process, it does aid the expert in understanding the process performance and focuses on important inter-relationships. The importance of clashes is that they present a logical interpretation for fault monitoring and diagnosis. Clashes aid the expert understanding of the flags that are set as the process operates.

## ACKNOWLEDGEMENTS

The authors would like to thank the faculty and students of the College of Engineering at North

Carolina State University, the Department of Computer Science at Strathclyde University, the Turing Institute, British Aerospace (HOTOL). Also, thanks go to Drs. David Williams and Andrew West of the Dept. of Manufacturing Eng., at Loughborough University of Technology.

## 7. REFERENCES

- [1] Albus, J. S., "System Description and Design Architecture for Multiple Autonomous Undersea Vehicles", NIST Report 1251, Gaithersburg, MD, September 1988.
- [2] Albus, J. S., McCain, H. G., and Lumia, R., "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", NIST Technical Note 1235, 1989.
- [3] Albus, J. S., "A Theory of Intelligent Systems", in Proceedings of the 5th IEEE International Symposium on Intelligent Control, Eds: A. Meystel, J. Hereth and S. Gray, 5-7 September, Philadelphia, PA, USA, pp 866-875, 1990.
- [4] Efstathiou, J., Davies, B., Razban, A., and Harris, S., "Expert Systems for an Adhesive Dispensing Robot", in Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE'93), Edinburgh, Scotland, pp 94-97, 1993.
- [5] Grant, E., "Machine Learned Control", Final Report to the NEL/BAe/BT/SERC, The Turing Institute, Glasgow Scotland, UK, 1992.
- [6] Grant, E., *The Knowledge-based Control of Robot Workcells and Dynamic Systems*, Ph.D. Dissertation, University of Strathclyde, 1999.
- [7] Isik, C. and Meystel, A., "Pilot Level of a Hierarchical Controller for an Unmanned Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. 4, June 1988, pp 241-255, 1988.
- [8] Makarovic, A., "A Qualitative Way of Solving the Pole-balancing Problem", *Machine Intelligence 12*, J. E. Hayes, D. Michie and E.Tygu (eds.), Oxford University Press, Oxford, England, 1989.
- [9] Michalski, R. S. and Larson, J., "Incremental Generation of the VL1 Hypotheses: the Underlying Methodology and the Description of Program AQ11", Technical Report: ISG 83-3, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, ILL, USA, 1969.
- [10] Michie, D. and Chambers, R. A., "BOXES: An Experiment in Adaptive Control", *Machine Intelligence 2* E. Dale and D. Michie (eds.), Oliver and Boyd, Edinburgh, Scotland, 1968.
- [11] Saridis, G. N., "On the Revised Theory of Intelligent Machines", CIRSSSE Report 58, ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, 12180-3590, USA, 1990.
- [12] Shepherd, B. A., *High-level Programming of Vision Guided Assembly Tasks*, Ph.D. Dissertation, University of Strathclyde, 1992.
- [13] West, A. A., Williams, D. J., and Hinde, C. J., "Experiences of the Application of Intelligent Control Paradigms to Real Manufacturing Processes", in Proc Instn Mech Engrs, Vol. 209, pp 293-308, 1995.
- [14] Williams, D. J. West, A. A. and Hinde, C. J., "A Discrete Process Control Software Testbed", Science and Engineering Research Council Report, Grant GR/F 37101, 1992.
- [15] Williams, D. J. West, A. A. and Hinde, C. J., "The Selection of Software Techniques for Discrete Process Control Applications", Science and Engineering Research Council Report, Grant GR/F 71973 1992.
- [16] Zhang, B., *Experiments in Learning Control Using Neural Networks*, Ph.D. Dissertation, University of Strathclyde, 1991.

# Assessing the Run-Time Performance of Artificial Intelligence Architectures

S. A. Wallace, J. E. Laird, and K. J. Coulter

University of Michigan  
1101 Beal Ave.  
Ann Arbor, MI 48109-2110

## ABSTRACT

*As intelligent systems are pushed forward to become more autonomous, there is a tendency for the underlying software architecture to grow in complexity to support these new behaviors. However, with the addition of new features, two potential costs may be incurred: increased execution time and additional memory requirements. As architectures evolve, it is important to continually evaluate the costs and benefits of each new change. Seemingly very similar architectures may require significantly different resources; small changes to the features in a single architecture may have a large impact on its performance. Thus, it is necessary to understand and to quantify the resources consumed by different architectures and by the components of a single architecture. Unfortunately, there is no standard method for evaluating features of an architecture or for comparing sets of architectures. In this paper, we begin by discussing such a methodology. We then dissect the Soar architecture into a core set of functionality and examine how incrementally adding each of the features found in the original implementation affects the overall performance and resource requirements. Next, we show how the same methodology can be used to compare two different architectures. Finally, we discuss initial results of a comparison that indicates both qualitative and quantitative differences between the Soar and CLIPS architectures.*

**KEYWORDS:** *Architecture evaluation, Soar, CLIPS*

## 1. Introduction

As artificial intelligent agents become increasingly robust and autonomous, the software underlying their behavior also becomes more and more complex. Success with simple agents in simple domains inspires research into the capabilities required to operate more efficiently and effectively. This in turn causes the software architectures to evolve, as functionality is added to support the new demands. Because this is a common process, many architectures have been developed incrementally over the course of many years as they become increasingly sophisticated.

Design decisions made at implementation time often play critical roles in the efficiency (both in time and space complexity) of the architecture. The impact is seen both when the new features are used by an agent and in some cases even when the features are not used. Thus, after a feature is added to an architecture, agents operating in complex domains and relying heavily on the new feature may operate more

efficiently than was previously possible, while agents that do not rely on the new architectural feature may become less efficient. To properly assess the impact of an architectural modification, it is necessary to quantify the resource consumption of that modification. In most cases, it is extremely difficult to draw meaningful conclusions using analytical methods, although in some cases, a formula that relies on prior knowledge of a relatively few variables may be obtainable. Even in these instances, however, comparisons between two such formulas are further hampered by the fact that constant factor differences may have profound implications on their relative suitability in real-world tasks. As a result, we believe that empirical methods are currently the most suitable way to evaluate the impact of design decisions.

Additionally, two distinct agent architectures are likely to yield agents with differing efficiencies even if the architectures (and agents) appear otherwise similar. As architectures become increasingly divergent, it may become overtly obvious that the features of one architecture are better suited to a particular task than are those of another. In many cases, however, this is not necessarily clear a-priori. As a result, designers of intelligent agents, and of agent architectures may benefit from understanding the relative differences in resource consumption between two or more architectures. As in the single architecture case, empirical methods can yield approximate answers to these questions. Unfortunately, however, there are no standard methodologies for evaluating the resource consumption of a particular architecture or of components of a single architecture.

In this paper, we discuss a methodology that can be used to examine the resource requirements of an architecture as a whole, or of particular aspects of that architecture. We present a practical example by applying this methodology first to components of the Soar architecture and then to the standard version of both the Soar and CLIPS architectures. Our results show both qualitative and quantitative differences between these two architectures and show how components of the Soar architecture contribute to its overall performance. Early versions of some of the material and results in this paper appeared in [15,16].



## 2. Architectures, Knowledge and Modularity

The class of AI symbolic architectures we are interested in are those that support the development of general, intelligent knowledge—rich agents. Following Newell's description [10], an architecture is the fixed set of memories and processing units that realize a symbol processing system. A symbol system supports the acquisition, representation, storage, and manipulation of symbolic structures. An architecture is analogous to the hardware of a standard computer, while the symbols (which encode knowledge) correspond to software. The role of a general symbolic architecture is to support the representation and deployment of diverse types of knowledge that are applicable to various goals and actions.

The basic functions performed by an architecture usually consist of the following (from Newell [10] p. 83):

- The fetch-execute cycle
  - Assemble the operator and operands
  - Apply the operator to the operands using architectural primitives
  - Store the results for later use
- Input and output

Architectures are distinguished by their implementation of these functions, and the specific set of primitive operations supported. For example, many architectures choose the next operator and operand by organizing their knowledge as sequences of operators and operands, incrementing a program counter to select the next operator. They also have additional control constructs such as conditionals and loops, but depend on the designer to organize the knowledge so that it is executed in the correct order. Other architectures, such as rule-based systems, examine small units of knowledge in parallel, selecting an operator and operands based on properties of the current situation. Some examples of these architectures include: Atlantis [4], CLIPS [1], Soar [7] and PRS [6].

Because the definition above leaves a fair amount of room for interpretation, architectures can often be further distinguished by the inclusion of additional functions, such as interruption mechanisms, error-handling methods, goal mechanism, etc. The inclusion of such functionality illustrates the necessarily blurry distinction between knowledge and architecture. Because most agent architectures are Turing complete, features not supplied directly by the architecture can often be emulated by the appropriate addition of knowledge, but with additional execution time overhead. However, it is often unclear a-priori how different design decision will affect future performance, and designers may choose to construct architectures modularly.

Architectures are modular in so far as features can be removed while still preserving the basic requirements of an architecture. Potentially, modules can be added or removed in order to optimize the architecture for a particular situation. Note that this is different from simply being able to refrain from using certain features because it suggests that the internal

design of the architecture with and without a modular feature is different.

## 3. A Methodology for Agent Architecture Evaluation

Our methodology begins with dissecting the architecture into constituent modules, leaving a core set of features intact. In many cases, such as when an architecture is developed incrementally, certain features may be naturally modular. In other cases, a great deal of thought may be required to determine what aspects of the architecture can be removed while still allowing the core functionality to meet the design goals of the researchers. In either situation, modifications to the source code will undoubtedly be necessary to construct a set of architectural variants that combine different modular features with the core functionality.

The second step in our methodology consists of determining a class of situations in which to examine the architectural variants. Particularly interesting problem classes may be found at both ends of a spectrum from situations that do not rely on a specific architectural feature to those which rely very heavily on such a feature. Although any single study is likely to be limited to examining a relatively small problem class, as the number of studies increases, it is anticipated that general trends will emerge indicating which architectural variant is most suited for a particular class of problems.

The third step involves selecting an environment in which to examine the problem class selected in the previous step. Because there is no single environment that can be used to represent "environments" as a whole, selection must be made with care, and equal care must be used to ensure that results are not over generalized. Understanding how the environment fits within a typical taxonomy (e.g. from Russell and Norvig [12]) may help moderate this problem.

Fourth, for each architectural variant, an agent must be designed to solve the specific problem within the selected environment. Agents solving the same problem form a group. All agents within a group must utilize the same problem solving methods. The effect of this constraint is that any two agents within a group must not only have identical interactions with the environment, but must also utilize the same internal problem-solving methodology. Proper implementation of this step is critical; otherwise there is a serious risk of confusing the contribution of different architectural aspects and different knowledge (i.e. problem solving methods) on the overall results. However, in certain circumstances this pitfall is eliminated because all of the agents within a group can be implemented using identical knowledge. This exceptional case occurs when architectural variants differ only in their inclusion or exclusion of unused features. Once a group of agents has been fully implemented, the performance of agent/architecture pairs can be directly compared.

## 4. Soar and Its Modular Components

The Soar [7] architecture is a forward chaining production system based on the RETE matching algorithm [2,3]. It contains a long-term memory (LTM) that stores production rules, and a short-term memory (STM) containing elements that are matched by the rules.

Short term, potentially volatile, knowledge is stored in STM in the form of a directed graph with labeled edges. Each memory element can be thought of as an ordered triplet whose slots refer to the parent node, the edge name and the child node respectively. Because this structure is so generic, it can be used to represent a multitude of more complex data structures.

Long term, stable knowledge, is stored in LTM as a set of productions. Productions are created explicitly by the programmer, or may be generated automatically by Soar's learning mechanism. The condition of a rule may contain either variables or constants, and variables may be bound to any of the three slots in a memory element's ordered triplet. This ability allows a large amount of flexibility in terms of how a rule is designed, but it can also greatly increase matching costs when it is used indiscriminately. The condition side of Soar's rules may also ensure that values bound to a variable satisfy one or more basic predicates (e.g.  $>$ ,  $<$ ,  $=$ ). Generic predicates, however, are not supported in a rule's conditions. The right hand, or action side of a rule, can be used to modify the contents of STM. Additionally, it can propose architectural-constructs called operators or preferences for such operators.

In Soar, knowledge is deployed by rule firings. This process begins as follows:

- First, determine which rules, if any, match the current contents of STM.
- Next, fire all matching rules in parallel, by executing the instructions in their right hand side.

These two steps, called an elaboration cycle, are repeated until a quiescent state is reached in which no more rules can fire. Parallel rule firings allow Soar to make use of all relevant knowledge in a given circumstance. It also forces programmers to explicitly encode control knowledge into rules to select operators instead of relying on a potentially cryptic architectural mechanism to determine which rule among the current matches should actually be fired.

In addition to the basic execution supported by the elaboration phase, Soar also has an architecturally supported decision-making phase that occurs immediately after elaborations have ceased. During the decision phase, operators representing actions of higher-level goals, which have been proposed during the elaboration phase, are examined. The operators are ranked according to their relative preferences, which have also been specified during the elaborations. At this point, Soar selects the operator with the highest preference to be pursued. Although the serial nature of pursuing operators

may seem similar to productions systems that fire rules serially, this is not typically true. One important distinction is that in Soar, knowledge about proposed operators is explicitly declared, and is available to be used for further reasoning, whereas information about matched rules in a serial system is typically not available to be used in this way.

In certain cases, Soar may decide that it is no longer making progress on the current problem (e.g. the elaboration phase terminates without any rules being fired, or Soar cannot select between two operators). In such cases, Soar will react by creating a sub-state in which further reasoning can take place. Within this sub-state, operators can be proposed and pursued just as in the super-state. The sub-state vanishes when Soar has done enough reasoning to resolve the problem that triggered its creation. It is during the resolution of sub-states that Soar's learning mechanism creates new search control knowledge (in the form of a rule) and adds it to LTM so that similar sub-states (and the additional reasoning to resolve them) can be avoided in the future.

Although Soar has been developed incrementally over a number of years, the mechanisms needed to modularize the architecture were not completely in place. Nonetheless, some features were clear candidates for modularization, and these are listed below:

- Detailed Timing Facilities - Soar has the ability to keep track of the time spent on various aspects of execution, but in many cases this information is not critical to the task.
- Callbacks – Soar has the ability to call user-defined functions during execution. Some of these callbacks are invoked many times per decision cycle, and even if no functions are registered with the architecture, some overhead is incurred due to looking up and testing one or more variables.
- Learning - Each time Soar completes reasoning within a sub-state, the architecture has the ability to learn a new rule. When using Soar in certain domains, however, learning has not been employed because these forces have been expected to perform at an expert level without undergoing a potentially costly training phase.
- Backtracing Mechanism - Soar also has the ability to keep (potentially elaborate) information as to how it reached a particular conclusion. The full power of this feature is used only during learning. Thus, as only a small portion of this mechanism is required for other purposes, significant amounts of source code can be removed or optimized when learning is also removed.

The four features we have identified above are only a subset of the features in the Soar architecture that could be modularized. However, this partitioning of the architecture was particularly suitable for our initial exploration because in certain testbed environments, a single set of knowledge could be used to examine all of the resulting architectural variants.

Three variants of the Soar architecture were examined for our tests by including or excluding some or all of the modular features described above. Variant 1, which we will also refer to as the standard version of Soar, includes all of the modular features. Variant 2, removes the Detailed Timing Facilities as well as the Callback module. Variant 3 removes all of the modular features described above.

## 5. Decision-Making Strategies

The class of problems we have selected for the initial implementation of our methodology is what we refer to as decision-making strategies. Most, if not all, agents are similar in that they must examine their current state and decide which of the many possible options to pursue. This process can take place in a variety of ways. In particular, one set of methodologies that can be used by Soar (as well as by a potentially large set of agent architectures) focuses on the individual pieces of knowledge which must be brought to bear in order to make the most appropriate decision about the next action. Some agents, for example, may use knowledge that directly ties a particular state or set of states to the most appropriate action. If the preconditions for each action are disjoint, only a single piece of knowledge will be brought to bear in any given situation, and the decision will essentially make itself. This is analogous to the operation of a lookup-table. Other agents may bring multiple pieces of knowledge to bear in order to make their decision. As the knowledge becomes hierarchically organized, the agent will go through an increasing number of refinement steps (reflected by a path in the tree from the root to a leaf) before it is able to select the most appropriate action for the circumstances. It is this general process of refinement that we have used as the basis for this study. Below is a list of decision-making strategies in which the refinement process is increasingly complex:

- Simple, Declared Actions - Actions are represented declaratively to the system, in Soar this is done using operators. The programmer supplies enough knowledge to guarantee that only one action is applicable at any given time, thus no conflicts between courses of action can arise.
- Three-Phase Decision - The decision takes place in three distinct phases. In the first of these, actions are proposed, in the second phase actions are ranked according to their relative preferences and finally the most preferred action is selected and pursued. This allows for multiple layers of refinement in the decision making process, potentially decreasing the size and complexity of the knowledge base.
- Goal Directed - A goal is a subtask that requires the application and pursuit of a sequence of one or more actions. In this strategy, goals are selected the same manner as primitive actions and may improve performance by constraining the subsequent problem

solving. Soar expresses goals with high-level operators, and uses sub-states to perform the reasoning needed to achieve these goals.

## 6. Towers of Hanoi

The Towers of Hanoi problem is well known to the AI community and has an equally well-known optimal solution. Although it is a relatively simple problem, it is complex enough to examine the class of decision-making strategies outlined in the previous section. Moreover, within this domain it is possible to limit differences between the agents' knowledge to exactly what is required to implement each decision making strategy. It is important to remember that we intend this environment to be used as a starting point for further investigation, and as a proof of concept. No single domain can claim to be representative of all situations an agent may face in general.

Table 1 shows the runtime performance of the Soar architectural variants described in section 4. Across all problem-solving strategies, significant timesavings are achieved between variants 1 and 2 as unused features are removed from the architecture. Further savings are achieved in the Tower of Hanoi subgoaling agent because the differences between variants 2 and 3 affect the efficiency of the architectural subgoaling process in situations where learning is not employed. Based on these results, and knowledge of how the architecture was modified, we expect that all Soar agents that do not require learning will achieve some performance savings by using the more streamlined architectural variants. Moreover, we further expect that agents that solve problems similarly to the Towers of Hanoi subgoaling agent above will be most enhanced. That is, agents that use a large number of subgoals, each of which requires relatively little reasoning to resolve on its own.

Variant	Declarative	3-Phase	Goal-Directed
Standard Soar	12.45	21.98	22.17
Variant 2	4.19	11.06	7.92
Soar LITE	4.13	11.42	5.64

Table 1. Soar Performance in Towers of Hanoi

## 7. Complex Real-Time Task: Quake II

The tests we conducted in Section 6 seemed to indicate that a substantial savings could be gained in situations that do not require learning. To substantiate this belief, we looked for other previously developed agents that shared this attribute and could be used for additional testing.

The agent we selected for this set of tests was an obvious choice. Constructed by one of us (Laird) to run with the latest version of Soar, it is suitably complex (employing ~600 rules)

and operates in the highly dynamic and complex environment of the Quake II computer game. Although Quake II shares few, if any, attributes with the Towers of Hanoi puzzle, application of our evaluation methodology within this new domain was straightforward. As in Towers of Hanoi, a single set of knowledge could be used to test all of the Soar architectural variants, and testing followed the same basic procedure. The only significant difference resulted from the fact that in the Quake II environment, exogenous events are possible. Unless the world's events unfold in exactly the same manner between tests of two architectural variants, it is impossible to determine whether the agents interacting with the world underwent the same processes of reasoning. As a result, whether or not the performance of the architecture/agent pairs is comparable also depends on the ability to ensure that the world's events unfold in a repeatable manner.

To ensure that this did happen, the agent was initially allowed to operate in the Quake II environment by competing against a human opponent for a predetermined amount of time. During this phase, the agent's sensory inputs were recorded and stored in a file. During benchmarking, however, agents did not actually communicate with Quake II. Rather, their sensory input was replayed in exactly the same manner as occurred during the initial recording phase. Not only did this allow us to ensure that agents always performed the same reasoning, but because agent inputs were read from disk and stored in memory prior to benchmarking, it also guaranteed that timing results would reflect Soar's true performance, and not be skewed by a communication bottleneck with the environment.

Figure 1 shows the run time performance in Quake II for the standard version of Soar (variant 1) and Soar Lite (variant 3). The performance was measured by recording the time required to complete each of 380 successive decision cycles. The histogram in Figure 1 shows the number of cycles that were performed within specific time frames. The best behavior is to have all of the decision cycles execute in the minimal amount of time (to the left). As this behavior is difficult to achieve, a secondary goal is to have a low variance without any outliers so that there are no decisions that disrupt the overall system execution. In the figure, the standard version of Soar does have a high variance and many outliers. In contrast, the Soar Lite version shifts the histogram to the left so that almost all of the decisions execute in .03 seconds or less. There is one significant outlier at .08, but that is the first decision when working memory is initialized and it is irrelevant to the overall runtime performance of the bot. This illustrates that Soar Lite not only improves the aggregate execution time (in this case there is a factor of 3 improvement in average execution time) but improves it at the level of individual decisions in a such a way as to decrease the overall maximum computational requirements of any single decision.

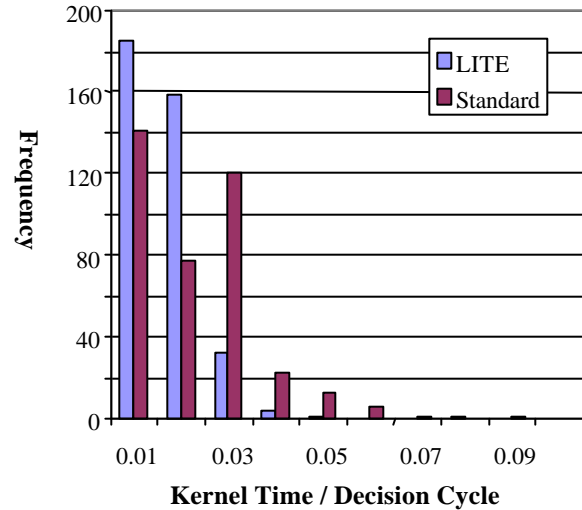


Figure 1. Execution cost in Quake II

## 8. Comparing Multiple Agent Architectures

The methodology described in section 3 and that we have employed to examine the performance of the Soar architecture and some of its variants can also be used to examine or compare two distinct architectures directly. The same steps are applied as outlined previously, but the architectures need not be split into modules. The most difficult aspects of using our method for distinct architectures are deciding what class of problems to examine and how to implement the agents. The difficulties stem from the fact that problem definitions must be highly constrained so that each agent's knowledge is extremely similar, if not identical. At the same time, however, these problem definitions are likely to require more flexibility than in the single architecture case, because perfect behavioral analogues may not exist between two architectures. Thus, the burden is on the research team to ensure that agents are appropriately similar and that they encode the same knowledge. As in the single architecture case, once agents have been created, their performance in the problem domain can be measured and compared.

### 8.1 The CLIPS Architecture

As an initial choice of a second architecture with which to conduct our evaluation, we have selected CLIPS [1]. Like Soar, CLIPS is a forward-chaining production system based on the RETE matching algorithm. In CLIPS, short term, potentially volatile, knowledge is stored in STM in the form of lists. Each list is given a name, or type, which is essentially the first element in that list. The remaining elements are labeled either explicitly or implicitly by referring to their position in the list. Each element is also a constant value, either numeric or string, and there is no architectural mechanism for referring to the contents of another list, or pointing to another slot.

As in Soar, long-term knowledge is stored as rules that are defined by the programmer. The conditions of these rules match against the contents of STM. Conditions can contain combinations of both constants and variables; however, variables may not be bound to list names or to slot labels. CLIPS, however, is not limited to using simple predicates in the right hand side of a rule as is Soar. A large variety of predefined predicates, as well as user defined predicates and functions can also be used as conditions. The action side of a CLIPS rule is used to modify the contents of STM or to execute external procedures.

CLIPS deploys knowledge via serial rule firings. The basic execution cycle consists of two steps:

- First, rule matches are calculated by comparing the conditions of each rule to the contents of STM.
- Second, successfully matched rules are placed into an ordered list such that the instantiated rule at the top of the list has highest priority.

Priority is defined using two methods. The first of these is a rule level conflict resolution mechanism called salience, which can either be a constant value, or a value calculated at run time. Rules with higher salience are placed higher in the list. In many cases, salience alone is not enough to determine a single highest priority rule. In these cases, CLIPS defers to one of a few user selected architectural mechanisms called search strategies, which orders rules of equal salience. At this point, the first rule in the list is fired and the entire process repeats itself. When no more rules are able to fire, the system halts.

Although there are many similarities between the Soar and CLIPS architectures, the differences are equally significant. These differences occur in each of the three architectural areas we have discussed: knowledge representation, knowledge deployment, and execution cycle. Recall for example, that Soar stores short-term knowledge in a directed graph structure and can perform variable binding on any slot in a memory element. CLIPS, on the other hand, stores short-term knowledge in lists, and cannot bind variables to the list name or to the names of its slots. Moreover Soar fires all matching rules in parallel whereas CLIPS fires only the highest priority rule. An additional difference is that Soar natively supports the decision making process within its execution cycle whereas CLIPS does not.

## 8.2 Towers of Hanoi revisited

We have examined CLIPS in the Towers of Hanoi domain using the same parameters that were used in our earlier evaluations of the Soar architecture. Note, however that the absolute timing data is not the same as in the first runs. Previous runs in this domain were done on different machines and measure total CPU time, not just Soar kernel time. Below, we briefly review the decision-making strategies of each agent

and discuss the particularities of the CLIPS implementation. Notice that two additional categories have been added to further constrain the implementations and to examine areas that may be more amenable to the CLIPS architecture.

- **Mutually Exclusive Reactions** - Action conditions are mutually exclusive, and no symbol is declared to represent the action being pursued. In both Soar and CLIPS this is done by the construction of individual rules which specify the preconditions of an action and its effects. Actions are applied sequentially within the world, and the programmer must ensure that no conflicts arise between two actions.
- **Simple, Declared Actions** - Similar to the first category, but in this case the action being pursued is declaratively represented. In Soar this is done using operators to represent the action. In CLIPS a fact is asserted which describes the current action being pursued. Once again however, the programmer must ensure that action preconditions are mutually exclusive.
- **Two-Phase Decision** - Two distinct phases are used to make the decision. In the first phase, actions are proposed via declarative symbolic representation. In the second phase one of these actions is selected and pursued. Note that this means that preferences corresponding to a specific action must be expressed simultaneous to the creation of the action symbol (e.g. within the same rule). In Soar, this is done using the architecturally supported decision phase, and the same rule is used both to propose an operator as to express its preference. In CLIPS, partitioning knowledge into a salience hierarchy supports the two phases. This guarantees that the first phase (action proposal) completes before the second phase (selection) begins.
- **Three-Phase Decision** - Three distinct phases are used to make the decision: proposal, preference and selection. These distinct phases help support situation dependent preference structures without an explosion of individual rules. In CLIPS this is done using a three-stage salience hierarchy.
- **Goal Directed** - High-level actions, possibly requiring more than one action to complete, are used to constrain rule matching. In CLIPS, goals are maintained declaratively and represented in a stack. Two Soar implementations were examined, one using Soar's native mechanism as demonstrated in the previous trials, and the other using a declarative stack similar to that used in the CLIPS implementation.

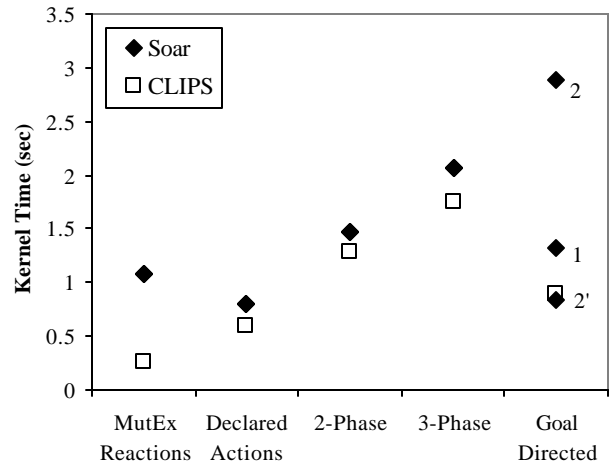
Figure 2 shows CLIPS and Soar performance in the Towers of Hanoi domain. Qualitatively, performance is very similar between the architectures except at the end points. On the left-

hand side of the graph, the Soar agent performs markedly worse than the corresponding CLIPS agent. This performance difference can likely be explained by the fact that this Soar agent does not use the operator construct. As a result, it does not benefit nearly as much from constraining the rule matching as the other Soar agents do, and thus suffers an increase in execution time. At the other end of the graph, Soar and CLIPS behavior are once again divergent. In CLIPS we can attribute the performance increase to the fact that the problem's recursive nature allows the proper puzzle-solving knowledge to be easily expressed with a goal stack, and results in highly constrained rule matching. In the standard version of Soar (point 2), however, we have already seen that the benefits of subgoaling are dominated by the costs of Soar's architecturally supported subgoaling mechanism. However, when performance is re-examined using architecturally supported subgoals in the Soar-Lite variant (point 2') or when using a declarative subgoal stack similar in nature to the CLIPS implementation (point 1) the difference between the Soar and CLIPS agent's performance is minimal. In all, the similarity of performance between the declarative goal stack implementations in both Soar and CLIPS, and the architectural implementation in Soar-Lite, indicate that in simple environments such as Towers of Hanoi, declarative subgoaling provides a simple and efficient means of problem solving. As tasks become increasingly complex, however, we expect that the rule-based techniques employed by these implementations will become significantly less efficient than the lightweight architectural counterpart of Soar-Lite

## 9. Related Work

Examining differences between agent architectures has received relatively little attention compared to the complementary task of examining how different agent strategies are more or less suited to a particular problem. Nonetheless, a variety of approaches have appeared in the literature. The majority of architectural evaluations can be placed into a single group that we refer to as categorical comparisons [5,8,9,14]. Within this body of work, architectures are evaluated at a high level, in a domain-independent manner, typically based on whether they natively support certain features (e.g. backward or forward chaining, or the ability to make real-time commitments). The benefits of this approach are that the concise tabular data, representative of these studies, may allow architectures to be quickly assessed as having or not having the minimal necessary capabilities to perform the task at hand. Categorical evaluations are most useful when they examine aspects of the architecture that are extremely difficult, or impossible, to emulate using additional, programmer supplied, knowledge.

However, the high-level approach of categorical evaluations can also be a short-coming. In particular, the many situations in which architectural features can, in fact, be successfully emulated with addition knowledge are often not explored. More over, because these studies rarely incorporate



**Figure 2.** Execution Time of Soar and CLIPS in Towers of Hanoi

benchmarks, there is often no indication as to the relative performance of different architectures or their underlying features.

In contrast to high-level categorical comparisons, the Sisyphus-VT initiative examined the problem of implementing a complex real-world problem on a number of different architectures [13]. Although the pursuit of complex, real world, problems as test bed domains is a laudable undertaking, the implementation overhead is extremely high. As a result, independent teams of programmers, expert in one particular architecture, carried out the implementations. A critical difference between the methodologies used in the Sisyphus-VT study, and the one we have presented is that we emphasize that the problem solving methods used by two comparable agents should be strictly specified and adhered to. Sisyphus-VT, on the other hand, allowed relative freedom in this area. Although this freedom allows programmers to use a problem solving method which they feel is best suited to their architecture, it also means that differences in two agents' performances might be attributable more to differences in their knowledge, than to differences between the architecture which serve as their foundations. Plant and Salinas attempted to circumvent the problem of confounding the contribution of knowledge and architecture to the overall performance rating in their 1994 study [11]. Under their methodology, agents were constructed in a generic manner so that they had minimal reliance on architecturally specific constructs. This allowed them to create agents for each architecture based primarily on syntactic transformation of a single, handcrafted, specification. This methodology certainly adheres to our requirement of strictly specifying the agent's underlying problem solving methods. But it deviates from our requirements because it does not examine a range of these underlying methods. As a result, it is less likely that the benchmarks will include near-optimal implementations for any architecture, especially since

reliance on architecturally specific constructs is purposely minimized.

## 10. Discussion

The methodology we have presented allows the performance of two architectures, as well as variations of a single architecture to be compared directly. Our methodology is an evolution of prior research, and emphasizes aspects of the benchmark design (e.g. problem-solving specification), which help ensure that agents built using two different architectures use equivalent knowledge. An initial application of our comparative approach has shown significant differences between 3 variations of the standard Soar architecture when Soar's learning capabilities are not required. This hypothesis was further supported by examining the performance of an agent in the complex, real-world, environment of Quake II. The broader implication of this finding is that knowledge both about the domain and about the implementation of the agent should play a role in deciding what architecture (and what architectural features) are most suitable for a particular circumstance.

We have also shown that the same methodology used to compare variations of a single architecture can also be used to compare two distinct architectures. We have illustrated this application with an initial comparison of Soar and CLIPS. Results from this set of tests indicated both qualitative and quantitative differences in their performance, and have also illustrated the potential performance savings that can be achieved by an architecture whose features are well suited to the current task.

We believe that the work presented in this paper provides a good foundation for addressing the question of what are the resource requirements of architectural properties, or, which properties of an architecture are most suitable for a given situation. Because the needs of intelligent agents often simultaneously push architectures to support a wide array of features and to be highly efficient in terms of run-time performance, an improved understanding of the answers to these basic questions is important.

## 11. Acknowledgments

This research was funded in part by grant N61339-99-C-0104 from ONR and NAWCTSD.

## 12. References

- [1] CLIPS Reference Manual: Version 6.05.
- [2] R. B. Doorenbos. *Production Matching for Large Learning Systems*. PhD thesis, Carnegie Mellon University, 1995.
- [3] C. L. Forgy. *On the Efficient Implementation of Production Systems*. PhD thesis, Carnegie Mellon University, 1979.
- [4] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for mobile robots. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 809-15.
- [5] W. B. Gevarter. The nature and evaluation of commercial expert system building tools. *Computer*, 20(5):24-41, 1987.
- [6] F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34-44, Dec. 1992.
- [7] J. E. Laird, A. Newell, and P.S. Rosenbloom. *Soar: An architecture for general intelligence*. *Artificial Intelligence*, 1987.
- [8] J. Lee and S. I. Yoo. Reactive-system approaches to agent architectures. In N.R. Jennings and Y. Lesperance, editors, *Intelligent Agents VI - Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2000.
- [9] W. A. Mettrey. A comparative evaluation of expert system tools. *Computer*, 24(2): 19-31, 1991.
- [10] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, 1990.
- [11] R. T. Plant and J. P. Salinas. Expert system shell benchmarks: The missing comparison factor. *Information & Management*, 27:89-101, 1994.
- [12] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter 2, pages 31-52. Prentice-Hall, Upper Saddle River, NJ, 1995.
- [13] A. Th. Schreiber and W. P. Birmingham, Editorial: the Sisyphus-VT initiative. *International Journal of Human-Computer Studies*, 44(3): 275-280, 1996.
- [14] A. C. Stylianou, R.D. Smith, and G. R. Madey. An empirical model for the evaluation and selection of expert system shells. *Expert Systems With Applications*, 8(1): 143-155, 1995.
- [15] S. A. Wallace and J. E. Laird, Toward a methodology for AI architecture evaluation: comparing Soar and CLIPS, in N.R. Jennings and Y. Lesperance, editors, *Intelligent Agents VI - Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2000.
- [16] S. A. Wallace, J. E. Laird and K. J. Coulter. Examining the resource requirements of artificial intelligence architectures. In *Proceedings of the Ninth Conference on Computer Generated Forces and Behavioral Representation*, pp. 73-83

# A Metric For Monitoring And Retaining Flight Software Performance

Chariya Peterson

Computer Sciences Corp. 7700 Hubble Drive,  
Lanham-Seabrook, MD 20706, [cpeters5@csc.com](mailto:cpeters5@csc.com)

## ABSTRACT

The control of spacecraft dynamics are handled by on board flight software which are typically based on sequential algorithm such as Extended Kalman filter (EKF) to perform closed-loop automatic control. This level of automation does not require any decision-making or learning capability. Decision-making capability comes to play at the higher level of autonomy in task management, such as mode/model selection, planning and scheduling. Most of these functions are still performed on ground and are not fully autonomous. In this paper, we propose the concept of intelligent flight software that is capable of learning, and improving its performance in the future based on information gained in the past. This capability will enable the software to appropriately deal with uncertainty or incomplete knowledge of model or environment. To be precise, we will focus on on-board Attitude Determination and Control software (ADCS). A typical ADCS is an automation where attitude solutions are computed dynamically via a filter and controlled by a closed-loop PID process. The performance of a typical ADCS is maintained by Flight Dynamics ground personnel. These tasks involve, among other things, attitude determination and validation, and attitude sensor model calibration. In this paper, we propose an intelligent ADCS that is able to monitor its own performance and able to perform a self-calibration when needed.

**KEYWORD:** *Control Theory, Intelligent software, Uncertainty Management, Machine Learning*

## 1. INTRODUCTION

The task of maintaining long-term performance and accuracy of software onboard a spacecraft can be a major factor in the cost of operations. In particular, the control and maintenance of constellation or distributed spacecraft undoubtedly pose a great challenge, since the complexity of multiple spacecraft flying in formation grows rapidly as the number of spacecraft in the formation increases. Eventually, new approaches will be required in developing viable control systems that can handle the complexity of the data and that are flexible, reliable and efficient. These new approaches will have to face the problems that are encountered during the development of a control system, in particular how to deal with uncertainties in the application domain and how to balance between efficiency and complexity of the system. The accuracy of control software

depends on how much information about the domain is modeled into the system. The more information taken into account, the more complex the system becomes, leading to higher computational cost. Hence pure model-based approaches will undoubtedly be too costly for a large control system.

Most of the material covered in this paper is in the paper presented at the SpaceOps Symposium, Toulouse, France June 19-23, 2000 [7].

## 2. SOFTWARE PERFORMANCE

Typical flight software performs closed-loop automation control without any high level decision-making, or learning involved. On the other hand, autonomy are added to the flight software in terms of flight or ground component that aims to increase operational range of the software, involving model selection, performance monitoring and self-calibration and tuning. We identify the intelligence of modern flight software with its decision-making capability, which results in the autonomy level of the software. We measure the intelligence of onboard software in terms of its ability to learn from experience and its rate of success. In the lowest level, we define the performance of flight software as a measure of the closeness between the observed and the predicted state of the systems. These quantities are usually referred to as *residuals*. Understanding the uncertainty underlying these residuals, identifying their controlling factors, and quantifying the propagation of these factors through the model for the system can lead to an improvement in the intelligence of the software.

On-board ADCS generally reacts directly with input sensor measurements and thruster control via simple closed-loop process. The typical operational range of such standard ADCS is narrow, and as a result, the system may perform poorly under uncertain conditions such as incomplete knowledge of world model, or unanticipated changes in the environment. To cope with this problem, the models used in the software are parameterized. The model parameters are adjusted regularly to maintain the accuracy level of the software. These tasks are typically performed manually on



the ground in a regular basis. This suggests that, the intelligence of flight software may be increased by enable the software with self-monitoring and self-calibration functionality. Recently, there have been a few research efforts in increasing the intelligence of flight software: for instance, the Remote Agent Experiment (RAX) onboard DS-1 spacecraft [1], and autonomous on-board dynamic monitoring developed at Jet Propulsion Lab, [BEAM].

We propose to develop the Monitoring and Autonomous Self-Tuning (MAST) system that aims to maintain the efficiency of onboard software by dealing with uncertainty in an appropriate way. MAST is an extension of a project at NASA/Goddard Space Flight Center (GSFC): Autonomous Model-based Trend Analysis System (AMTAS) [2]. MAST extends the objective of ASCAL from health and safety management of hardware to dynamic applications. MAST uses machine learning approach to handle uncertainty in the problem domain, resulting in the reduction of over all computational complexity. The underlying concept of this technique is a reinforcement learning scheme based on cumulative probability generated by the past performance of the system. The success of MAST depends largely on the reinforcement scheme used in the tuning algorithm and its ability to remember and learn from its experience.

### 3. THE MONITORING PROCESS

MAST consists of two main parts: a *monitor* and a *tuner*. The monitor is a real-time dynamic system that monitors relevant residual output of the software it is monitoring. The step size of the sampling time varies depending on the parameters being monitored. The state of the monitor is the quantity representing software performance in real time. When the state of the monitor approaches a given threshold, the tuning process will be initiated. This process has no intelligence i.e. it does not require any decision-making capability.

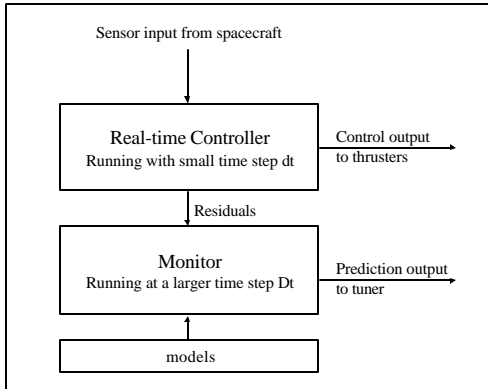


Figure 1. Monitoring Mode

Figure 1 demonstrates the monitoring mode, which consists of the software being monitored and a monitor, both running in real time. The detail description of the monitor depends on the software being monitored. It is necessary that the monitor have sufficient knowledge of the software in order to make an accurate prediction and diagnosis of the problems.

For intelligent ADCS where Kalman filter is used for real-time computation of attitude solutions, the performance of ADCS is monitored by trending the attitude solutions and the effective sensor measurements. In a nutshell, attitude of spacecraft is continuously propagated through time using angular rate measurements from gyroscopes. These attitude solutions are not very accurate since gyroscope measurements are usually erroneous. The accuracy of attitude computation can be improved by occasionally comparing the attitude with vector (directional) measurements from available sensors on-board such as star trackers, magnetometers, sun or earth sensors. Kalman filter is an algorithm that performs such sequential process of propagating and measurement updating. The size of the residuals of sensor measurements reflects the performance of ADCS.

Let  $x$  denotes the state vector estimated by the software and  $s$  denotes the vector of sensor parameters being monitored and calibrated. Assume that an expected state vector  $x_a$  is given.  $x_a$  may be obtained in various ways depending on the software and on sensors and parameters being monitored. Let the software be driven by the dynamic system

$$\begin{aligned} \dot{x}(t) &= f(x(t)) + u(t) \\ z_{r,k} &= G(s_r, x(t_k)) + w(t_k) \end{aligned} \quad (1)$$

where  $z_{r,k}$  is the measurement for sensor  $r$  at time  $t_k$ , and  $s_r$  is the parameter vector associated with the model of measurement  $r$ . The process noise  $u$  and measurement noise  $w$  is assumed to be uncorrelated white Gaussian noise with zero mean. During the normal mode of operation  $s_r$  are kept constant. The performance of (1) is observable from the deviation of certain quantities, such as state residuals  $x - x_a$ , and sensor residuals,  $z_{r,k} - G(s_r, x_a(t_k))$ . Let  $\mathbf{x}$  represents the vector of the desired residual observations. The monitoring process is then defined via a tracking process, i.e. the linear dynamic of  $\mathbf{x}$  and its slope  $\dot{\mathbf{x}}$ :

$$\mathbf{x}(t_{K+1}) = \mathbf{x}(t_K) + \Delta t \cdot \dot{\mathbf{x}}(t_K) + \frac{1}{2} \Delta t^2 \ddot{\mathbf{x}}(t_K)$$

$$\dot{\mathbf{x}}(t_{K+1}) = \dot{\mathbf{x}}(t_K) + \Delta t \cdot \ddot{\mathbf{x}}(t_K)$$

where  $\mathbf{n}$  is a zero mean white Gaussian uncorrelated acceleration noise. The time step  $\Delta t = t_{k+1} - t_k$  for residual samplings may be larger than the time step of the input system (1). Let  $\hat{\mathbf{x}} = [\mathbf{x} \ \dot{\mathbf{x}}]'$ . Then the state-space representation of the predictor can be written as

$$\begin{aligned}\hat{\mathbf{x}}(t_{k+1}) &= A \cdot \hat{\mathbf{x}}(t_k) + V \cdot \mathbf{n}(t_{k+1}) \\ \hat{z}_k &= H \cdot \hat{\mathbf{x}}(t_k) + \mathbf{g}_k\end{aligned}\quad (2)$$

where

$$A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, V = [\Delta t^2/2 \ \Delta t], H = [1 \ 0].$$

Note that, the measurement  $\hat{z}_k$  represents the residual sampling while the state  $\hat{\mathbf{x}}(t_k)$  measures the level of performance of (1) during the time  $t_k$ . A propagation of  $\hat{\mathbf{x}}(t_k)$  predicts if and when the performance of (1) approaches an acceptable threshold at a certain time in the future. The system (1) and the predictor (2) connect as shown in Figure 1. Higher order derivatives of state residuals can also be included in  $\hat{\mathbf{x}}(t_k)$  in a similar way. In which case, we would have a higher order predictor. Higher order derivative may be crucial for software systems that are sensitive to uncertainties in measurement models, which is generally the case for a highly non-linear, chaotic or unstable systems.

#### 4. TUNING PROCESS

The tuning process is a closed-loop learning algorithm based on a reinforcement learning scheme. The goal of the tuning process is to restore the performance of the software by iteratively adjusting relevant model parameters in a “*certain way*” until a cost function is minimized. The tuner possesses two types of intelligence:

- 1) During each cycle the tuner will select which parameter to adjust. This selection is MAST’s long-term knowledge on its past tuning experience. This intelligence is measured by the rate of success in software tuning.
- 2) The amount of adjustment for each parameter. This selection is a short-term knowledge generated by the reinforcement scheme of the learning algorithm. This type of intelligence is measured by the rate of convergence for each particular tuning process.

Note that, the learning approach does not give an optimal solution, but it has a much wider operational range than the conventional optimal batch least square or filter techniques. This is simply because; MAST automatically accumulate and reuse its past activities in its long-term

memory, which will enable the system to react and adapt to changes in the environment. This approach is therefore appropriate for problems with large degree of uncertainties. Moreover, this technique is not critically dependent on the detailed knowledge of the software being tuned. As a result, some of the technical restrictions generally required in conventional techniques such as linearity, or conditions on process and measurement noises are not required if a learning algorithm is used. It should be noted that the tuner is an off-line algorithm, or a process running in parallel and isolated from the routine operation of the software. Not until the tuning goal has been reached, that the software will be updated with the new values for the model parameters. Hence, the tuner may be performed on the ground or on an onboard computer.

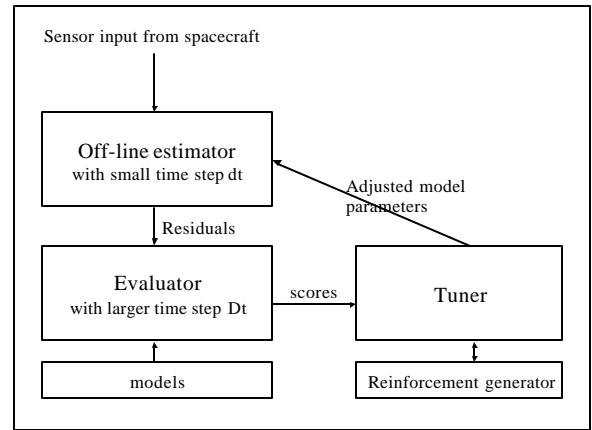


Figure 2. Tuning Mode

Figure 2 demonstrates the tuning mode. In this mode, there are three components connected in a closed-loop: an off-line copy of the software being monitored, the evaluator, and the tuner. The evaluator measures the convergence of the tuning solutions and the tuner makes appropriate adjustment to certain model parameters of the software guided by a reinforcement learning scheme, generated by an uncertainty handling technique. Several techniques have been used by various research projects in reinforcement learning. In MAST, the scheme is based on the Local Dempster-Shafer theory (LDS) which is a modification of the Dempster-Shafer theory of belief and evidence [4,5]. For the detail description of LDS we refer to [2,6]. LDS is specifically developed to deal with systems with large number of variables. As opposed to the monitor, the evaluator and the tuner are generic processes that do not require detailed knowledge of the software being tuned. Their basic requirements are a set of software parameters to be tuned and an appropriate cost function that models the inaccuracies of the software. The evaluator evaluates and scores the result of each cycle by examining the effect of the parameter adjustment on the cost

function. Based on this score, the tuner continues to adjust the parameters until the process converges.

Reinforcement learning is the type of learning that is popular among most current researches in machine learning and statistical pattern recognition. Other popular type of learning systems such as artificial neural network, requires *a priori* training from examples provided by an experienced supervisor. Such systems are not quite appropriate for problems involving learning from interaction. In interactive problems it is often impractical to obtain examples of desired behavior ahead of time, which are both correct and representative of all the situations to which the system has to react. In an unknown situation, where learning is most beneficial, the system must be able to learn proactively from its own experience.

During the tuning process, the parameter adjustment is based on the rate of convergence (or divergence) of the residuals during the previous two (or more) cycles. Assume there are  $n$  sensor parameters to be adjusted, i.e. the dimension of the parameter vector  $p_k$  is  $n$ . The parameters can be increased or decreased by  $\Delta p_k$ . The set  $H$  of all

possible adjustments has  $\sum_{i=0}^n 2^i \binom{n}{i}$  elements. Each

element is a set of parameters with a plus (+) or minus (-) sign to denote if the parameter is being increased or decreased. For instance, an increase in parameter  $a$  and a decrease in parameter  $b$  is represented by the "signed" set  $\{a_+, b_-\}$ . During each loop  $K$ , the step size  $\Delta p_k$  is computed, and the set  $H$  is constructed. An indexed by a cumulative probability distribution  $p_k$  which generated by LDS theory. The learning process in the tuner is precisely the mechanism that adapts  $p_k$  to obtain the new index  $p_{k+1}$  for the next cycle. The original Dempster-Shafer theory is defined on a set of  $n$  elements. Recall that,  $H$  is a set of all possible ways of modifying model parameters being tuned. A mass function on  $H$  is a probability function that assigns a degree of belief to each of its element. The mass function satisfies the following conditions

$$\sum_{A \supseteq H} m(A) = 1, \quad \text{for } A \neq \emptyset \quad \text{and}$$

$$m(\emptyset) = 0$$

Two mass functions  $m_1$  and  $m_2$  on  $H$  can be combined into a single mass function  $m_1 \otimes m_2$  by the Dempster composition rule:

$$m_1 \otimes m_2(A) = \frac{\sum_{B \cup C = A} m_1(B) m_2(C)}{1 - \sum_{B \cup C = \emptyset} m_1(B) m_2(C)}$$

for  $A \neq \emptyset$ ,  $m_1 \otimes m_2(\emptyset) = 0$ .

These mass functions are used to generate the degree of belief associated to each element of  $H$ . A belief function generated by a mass function  $m$  is defined as:

$$p: H \rightarrow [0,1]; \quad b(A) = \sum_{B \supseteq A} m(B)$$

where the union between two signed sets is obtained by "adding" all elements in the two sets according to their sign. This way, every subset of the form  $\{a_+, a_-\}$  will all be cancelled out. In statistical terms, the belief function is a cumulative probability on  $H$ .

During a tuning cycle  $K$ , the belief function  $p_k$  is evaluated and used as an index set for  $H$ . If the resulting residuals are found to decrease with a faster rate or increase with a lower rate, the tuner will re-compute the next belief vector  $p_{k+1}$  by applying a positive learning algorithm described in [1,9]. The new index will strengthen the performance of the cycle  $K$ . Conversely, if the residuals performed in the negative manner, then the negative learning algorithm will be applied, resulting in lessening the degree of belief on the failed action.

The learning process discussed above is the simplest application of the (modified) DS theory to the tuner. In practice this algorithm can be enhanced in various ways to increase the performance and robustness of the tuner. First, the localization of the DS theory on  $H$  defined in [1,9] will reduce the size of search space. Second, the size of parameter increment may be decreased as the residuals begin to converge. Third, the use of hierarchical or multilevel learning systems accelerates the learning process (more so for the initial rate of learning) and simplifies the structure of the tuner in each layer.

## 5. TWO APPLICATIONS OF MAST

The attitude monitoring and self-calibration (ASCAL) [3], and the maintenance of spacecraft formation. In the first application, the accuracy of attitude software depends on, among other things, the accuracy of sensor models. These models are generally a function with parameters representing relevant uncertainties such as bias, scale factor or misalignment. In the beginning, these parameters are set at certain pre-calibrated values and are manually tuned and updated periodically throughout the life of the spacecraft. Some tuning processes are routine activities, while others are elaborated and performed on ground by attitude specialists. In this proposed application, MAST will automatically monitor and tune a set of sensor parameters.

The second example is the maintenance of large formation of spacecraft. The task of controlling a number

of spacecraft to fly in formation is more complicated than controlling a single spacecraft. One problem that may be encountered in the development of formation control algorithms for large formation is the complexity that arises from the high degree of freedom of the system. In practice, the conventional approach based on state-space representation is manageable only for formation of a small number (2-3) of spacecraft. The complexity increases in a large formation, which makes the control algorithm computationally intensive. Moreover, uncertainties in the system models or from environmental disturbances can be propagated and magnified. To correct these errors the control system has to be tuned often and regularly to keep the formation intact by continuously monitoring and adjusting the position of each individual spacecraft. Ideally, these tasks should be performed onboard, and hence efficient and fast algorithms for the real-time solution of such a large-scale optimization problem are needed.

## 6. REFERENCES

- [1] N. Muscettola, B. Smith, C. Fry, S. Chien, K. Rajan, G. Rabideau, and D. Yan, *On-Borad Planning for New Millennium Deep Space One Autonomy*, Proceedings of the IEEE Aerospace Conference, Aspen, CO 1997.
- [2] C. Sary, C. Peterson, J. Rowe, K. Mueller, W. Truskowski, T. Ames, N. Ziyad, *Automated Multimodal Trend Analysis System*, Proceedings of the AAAI Spring Symposium 1998.
- [3] C. Peterson, J. Rowe, K. Mueller, N. Ziyad, *ASCAL: Autonomous Attitude Sensor Calibration*, proceedings of the Flight Mechanics Symposium, NASA/GSFC May 1999. M. D. Shuster, D. M. Chitre, and D. P. Niebur, *Inflight Estimation of Spacecraft Attitude Sensor Accuracies and Alignments*, J. of Guidance and Control, 5, 4, 1982.
- [4] G. A. Shafer, *Mathematical Theory of Evidence*, Princeton U. Press, 1976.
- [5] A. P. Dempster, *Upper and Lower Probabilities Induced by Multivalued Mappings*, Annals of Math. Stat. 38, pp. 325-329, 1967.
- [6] C. Peterson, *Local Dempster Shafer Theory*, CSC-AMTAS-98001, C.S.C Internal Report
- [7] C. Peterson and N. Ziyad, *Autonomous Performance Monitoring System: Monitoring and Self-Tuning (MAST)*, Proceedings of the SpaceOps 2000 Symposium, Toulouse, France, June 19-23.

# Flexible Robotic Assembly

David P. Gravel  
Senior Technical Specialist  
Ford Advanced Manufacturing Technology Development  
Detroit, MI 48239

Wyatt S. Newman  
Professor EECS Dept., Case Western Reserve University  
Cleveland, OH 44106

## ABSTRACT

Application of robots in automobile manufacturing plants is primarily limited to material handling and spot welding operations. Only 3 percent of all robotic applications are currently performing tasks related to assembly [1]. These assembly tasks often have the characteristic that the positional uncertainties in parts to be mated exceed the assembly tolerances by many times. Humans are particularly capable of part assembly under these conditions for three reasons: (1) Humans can apply compliant, goal directed forces and positions to the assembly task. (2) Humans have a powerful vision capacity that integrates well in the application of these goal directed forces and positions. (3) Humans are also quick to learn new assembly techniques and can often perform complex assembly tasks easily after a short training period.

This paper will detail the work done at Case Western Reserve University (CWRU) in Cleveland, OH, and by Perceptron in Plymouth, MI on a NIST ATP for Flexible Robotic Assembly for Powertrain Applications (FRAPA). FRAPA is a Joint Venture comprised of the following companies: Ford Motor Co., MicroDexterity Systems (MDS), ComauPICO, Perceptron, and the National Center for Manuf. Sciences. Also participating as subcontractors in the FRAPA project are Sandia National Laboratories, the University of Michigan, and CWRU.

This work will lead to the production of an autonomous system that exhibits all three characteristics that humans are naturally endowed with, that is: goal directed compliant forces and positions applied to part assembly, visual feedback to reduce the part location uncertainty, and learning algorithms that improve the performance of the assembly task over time.

## STATUS OF CURRENT TECHNOLOGY: POSITION CONTROLLED ROBOTS

Robots used as position-servoed mechanisms are ineffective as an assembly tool in cases where the assembly tolerance is less than the positional uncertainty. This can be illustrated using a simple example of a peg-in-hole problem.

Suppose a position-controlled robot attempts to assemble a peg into a hole, but the hole position is not precisely controlled, and further, the assembly tolerance is comparatively small. Unless the centerlines of the peg and the hole are nearly parallel and lie closer together than the assembly tolerance, the robot will not be able to insert the peg into the hole. In a misaligned state, if the robot does attempt to perform the assembly, unacceptably high contact forces will be generated as the robot attempts to push the peg into place. Even if the peg is chamfered, a misalignment between the peg and hole would produce large side loading forces as the robot struggles to move the peg along the programmed centerline of insertion. This misalignment of the peg is likely to cause a jam if no means for compliance, such as a remote center compliance (RCC) device, is employed (see e.g [2]).

In practice, the geometry of real parts is often more complex than a peg and hole which, can further complicate robotic assembly and decrease the chances for a successful automated assembly using position-controlled robots.

## I. A NEW ROBOT CONTROL PARADIGM

Wyatt Newman at Case Western Reserve University in Cleveland, OH has implemented a robotic control strategy called "Natural Admittance Control" (NAC) [3,4,5], on several robot platforms. Instead of having the robot stiffly track a precise

trajectory, it is guided by an attractor point with pre-programmed stimulus-response dynamics. This new robot control strategy provides excellent rejection of Coulomb friction, good force responsiveness, and guaranteed contact stability.

A programmed point called the "attractor point" controls the direction of the applied force. It is the attractor point that is manipulated by the trajectory generator in the robot. The attractor point pulls the robot tool center point (TCP) towards it with strength proportional to programmable spring constants in the X, Y, Z, Y, P, R directions. The damping of the robot TCP is accomplished in a similar fashion with programmable damping coefficients  $B_x, B_y, B_z, B_p, B_r$  that model a damping forces proportional to the velocity components in the  $V_x, V_y, V_z, V_p, V_r$  directions.

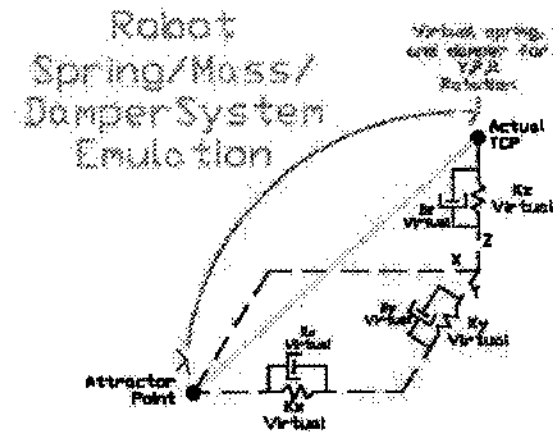


Fig. 1 Simplified schematic of robot TCP modeled as a programmable spring/mass/damper system.

Changes in the state of assembly can be accomplished by either altering the attractor point position or by changing the virtual spring or damping coefficients. In this manner it is possible to change the applied forces or endpoint dynamics as required for the assembly task.

In addition to the NAC robot control scheme, it is desirable to implement force controlled robotics on a robot platform that has low inertia, low gear friction, and high mechanical stiffness and is backdrivable. To address these control concerns and to provide a high fidelity response to input forces, a robot called Paradex has been created.

Paradex is a parallel robot structure comprised of six linear motor axes joined at a common distal

platform. This configuration of robot meets the criterion for low inertia, excellent mechanical stiffness, and back drivability. The first generation of Paradex, shown in Fig. 2, is currently undergoing testing and development at CWRU.



Fig. 2 FRAPA Paradex I robot

The NAC paradigm of robot control can be made to act as a conventional position controller by using very stiff virtual spring coefficients, e.g. for handling parts that are precisely or for spot-welding applications.

Given that NAC can be accomplished on a responsive robotic platform, the remaining task is to provide programmed strategies based on machine assembly states that can be first identified and then resolved by intelligently changing the virtual parameters and endpoint dynamics. This will be further discussed in section III.

NAC provides robots with the first human attribute of actively controlled, goal directed forces and positions being provided to parts to be assembled.

## II. MACHINE VISION

Humans can easily visually locate and acquire parts to be assembled. Machines however, do not possess such a facility. Machine vision has historically been plagued by a lack of robustness in factory applications. Many of the problems related to the robustness of grayscale vision processing are due to a lack of contrast of features of interest, and to

changes in lighting that cause object segmentation errors in the image.

3-D range imaging technology does not suffer from the contrast and lighting issues of grayscale vision processing, but many of the algorithms that can be commonly found for grayscale vision processing do not operate on 3-D range data. Under a NIST ATP for Flexible Robotic Assembly for Powertrain Applications (FRAPA), Perceptron in Plymouth, MI is developing new vision processing algorithms that will provide a robot with part pose information that can be used to acquire randomly located parts.

One disadvantage of range imaging machine vision systems, however, is that the speed of image acquisition is about 2 s compared to 0.016 s for a CCD grayscale camera. This limits 3-D range imaging to applications to ones where the scene is static or slowly changing in time. Fortunately, in automotive applications, there are many examples of static scenes that this technology can be utilized.

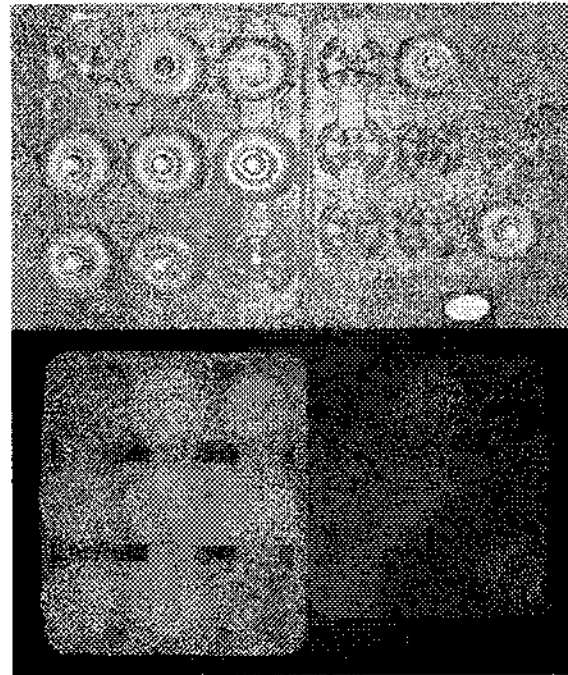


*Fig. 3 Example of parts delivered to an imprecise location that need to be acquired prior to assembly. Currently most of these operations are done manually in automotive plants.*

Another disadvantage of range images is that distances measured from 0 -∞ show a periodicity. This results from the measured phase shift reflected from an object back repeating at a distance called an "ambiguity interval" due to the phase changing from 0 to 360 deg. and starting over at 0 deg. again. Fortunately, the ambiguity interval is somewhat large (~2m) so there are a number of plant applications that satisfy the condition of using a single ambiguity interval over the workspace.

The raw sensor data from the range vision system is acquired in a spherical coordinate frame with more pixels spatially located in the central region of the image than on the edge. An image region of interest (ROI) is rectified into a regularly spaced (X,Y,Z) 3-D array and processed by a new class of operators [6] that can quickly perform part identification and deliver part pose information as well.

These new vision operators are model based and yield a set of scalar values that represent both the existence of an object and pose information about the object. These operators are also relatively fast compared to correlation techniques and have full 6 degrees of freedom (DOF) isometry invariance.



*Fig. 4 shows a comparison of an intensity image(top) and a range image(bottom). These two pallets contain automotive torque converters. The bottom left hand pallet is a brighter image because it is closer in the range image.*

The methodology employed by Perceptron to locate parts in roughly placed pallets follows these steps. (1) Locate the boundaries of the pallet and calculate ROI windows to isolate individual parts. (2) In each ROI window transform the row-column-range raw data into a uniformly spaced X-Y-Z representation. (3) Use the new vision operators to locate the part pose in each ROI. (4) Calculate robot coordinates in the robot world coordinate frame so the robot can pick up the parts from the pallet.

The capacity to locate roughly placed parts in 3-D will be used to robotically acquire these parts. This capacity provides automation with the second human attribute of coordinated vision to perform robotic assembly.

### III. MACHINE LEARNING APPLIED TO ROBOTIC ASSEMBLY

Humans are able to learn complex assembly tasks relatively quickly. Machines, however, are notoriously difficult to "teach". Any ability to provide a machine with a learning capacity to perform a task will require decomposing the task into subtasks that can be understood and modeled in ways a machine can use.

Work at CWRU on machine learning of automotive assembly tasks has focused on four areas. (1) Gain an understanding how people perceive edges and boundaries when attempting to mate two parts together. (2) Understand strategies people invoke under various force feedback conditions (3) Create a model based system that uses signal feedback to direct the robot motion to the proper position for assembly (4) Implement the knowledge gained in steps 1-3 into a neural net robot controller.

At CWRU an idealized peg-in-hole virtual model was created for round and square peg-in-hole applications. Moments can be shown to exist as a portion of a peg passes over the region of a hole. A peg-in-hole model is similar to a simplified set of problems in transmission assembly. The goal is to extend this work to automotive assembly tasks such as a sun gear insertion into a planetary gear set.

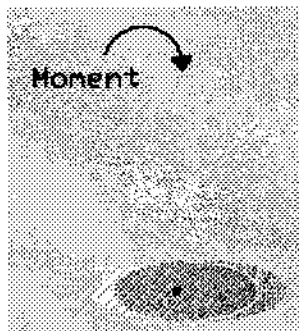


Fig. 5 Near the hole moment information exists that be exploited for determining the location of the center of the hole for automated assembly.

Figure 6 shows the computed direction of the reaction moment vector for a square peg and hole

when the peg z-rotation is close to the assembly orientation. Looking for patterns in the robot force/torque sensor signals to match these theoretical moments may provide a robot with the ability to infer the location of the assembly position and orientation as the assembly attempt proceeds.

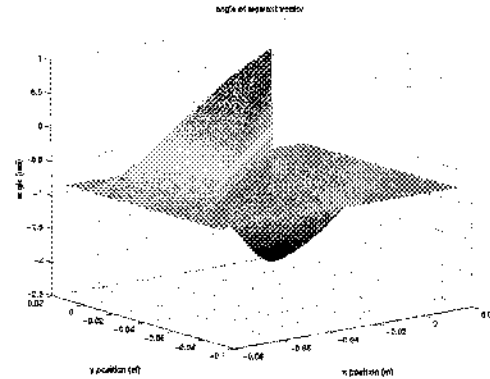


Fig. 6 Theoretical moment plot for a virtual square peg-in-hole simulation (peg z-rotation nearly aligned with hole)

CWRU researchers were able to show the dramatic effect that moments play in the human perception of assembly by feel. Experiments were conducted performing virtual-reality assemblies with a Cybernet force-reflecting hand controller. Mobility was restricted to the equivalent of a 4-dof SCARA robot (i.e., wrist pitch and roll were fixed). Computed interaction moments about the x and y axes (e.g., as shown in Fig 6) were reflected back to the hand controller as the operator attempted the virtual assembly. Without the benefit of vision, the operator had to search blindly in x and y (and z-rotation, for square pegs) for the insertion location. For a small-clearance assembly, a blind search without sensory feedback was typically unsuccessful within 2 minutes per trial. However, when the moment information was displayed haptically, human operators were able to learn how to interpret these signals and guide the peg to the assembly location. For round pegs, the mean assembly time for a trained operator using moment-feedback cues was less than 7 seconds.

While we can conclude that humans can exploit perception of reaction moments to dramatically improve assembly performance, it is not obvious how these signals are being interpreted and utilized. To



help expose the essential features of such feedback cues, the moment information was deliberately corrupted before being presented to the user. Low-pass filtering, high-pass filtering and nonlinear transformations were performed to determine the influence on usability of the signals by humans. Figure 7 shows a record of high-pass filtered computed interaction moments presented to the operator during an assembly trial. This limited feedback was also successfully interpreted by humans to achieve rapid successful assemblies. This feedback was even further corrupted to display (exert) pulsed moments of fixed amplitude and duration when the rectified, high-pass filtered moment data exceeded a threshold. Remarkably, humans were just as adept at keying off of this impoverished feedback to achieve rapid assembly success. Such experiments are helping to identify the essential features of sensory feedback for expert, sensor-based assembly.

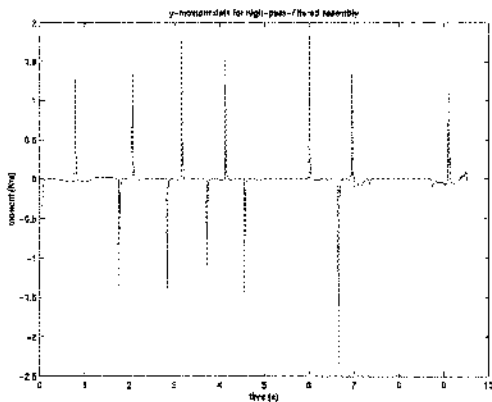


Fig. 7 Highpass filtered moment data is sufficient for humans to perform edge identification

In addition to testing which sensory cues are most effectively utilized by humans, we can also observe human assembly strategies from records of virtual assembly trials. Figures 8 and 9 show records of novice and trained human operators attempting the teleoperated virtual assembly of a round peg in a round hole. Both the experts and novices exhibit the behavior of attraction towards a sharp discontinuity in reaction moments (the vertical line segment separating large negative moments from large positive moments). The expert operator utilizes this discontinuity in a purposeful manner. The expert behavior shows an initial scan (apparently seeking the discontinuity boundary) followed by tracking the

moment discontinuity boundary towards its apex. If the assembly location near the apex is overshoot, the operator detects the loss of signal and loops back to restart the search within a small neighborhood of the solution.

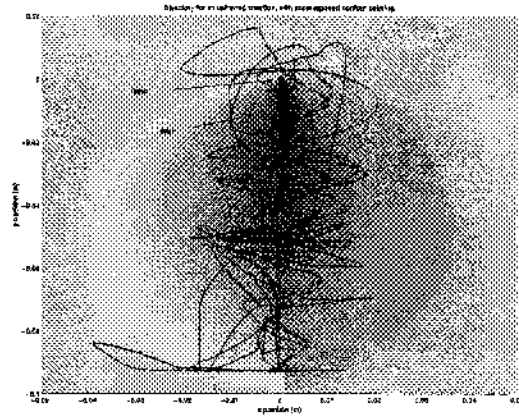


Fig. 8 The path taken during an untrained assembly trial.

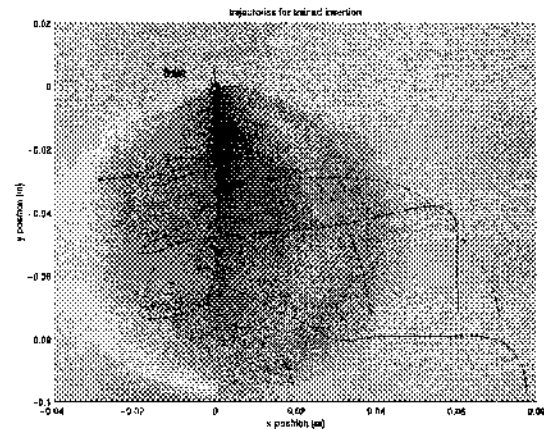


Fig. 9 The path taken during an trained assembly trial. Notice that this path is much more of an optimal trajectory to achieve assembly.

Based on such observations of human strategies for assembly, researchers at CWRU have been able to construct a neural-net based controller that exhibits similar sensory interpretation to perform guided assembly. Using an NAC-controlled AdeptOne robot equipped with a JRRR 6-axis force/torque sensor, a map of reaction moments vs displacement errors was experimentally obtained for a peg-in-hole assembly task. The acquired raw data was used to train a neural net to recognize assembly coordinates from x and y moment signals.

After training, the quality of the learned mapping was tested. Over a large region of displacements, the mapping was untrustworthy. In contrast, over a small region, the mapping was reasonably reliable and precise. Significant additional regions were capable of coarse but useful predictions. These results are illustrated in Fig 10. The sparse x's indicate regions of high-quality predictive capability, the triangles and squares are regions of coarse but useful mappings, and the regions of diamonds and circles correspond to poor predictive capability.

Following the strategy of human operators, our intelligent controller incorporates a sensory-driven behaviors. When in the low-quality mapping regions, the robot is controlled to perform a compliant raster search for the hole. When useful sensor information becomes available, the robot alters its search direction as indicated by the vectors in Fig 10. The coarse information is sufficient to guide the robot towards the region of high information, after which the robot can follow a reasonably precise path towards the goal.

The raster-search phase is like the approach phase of the human operator illustrated in Fig 9. In the region of the moment discontinuity, the operator (and the robot) are drawn towards the discontinuity boundary. Upon reaching the discontinuity boundary, both the human operator and the robot follow this narrow region of high-quality sensory information towards the goal.

The intelligent control algorithm was tested on an AdeptOne robot controlled by NAC. It demonstrated searching behavior like that of humans for a simple (large-clearance, round peg-in-hole) task. For tighter clearances, however, the algorithm was not sufficiently robust. Subsequently, sensory interpretation was based on sensory patterns (moments vs time) rather than point-by-point measurements. With this augmentation, it was much easier to recognize key features (peaks and discontinuities) from a sequence of measurements, as opposed to depending on fortuitous samples within the slim region of high-quality information. It seems likely that humans perform similar processing, interpreting moment feedback in terms of temporal patterns rather than processing of distinct, point-wise measurements. (This would be consistent with, but not deducible from the recorded trajectories of Fig 10).

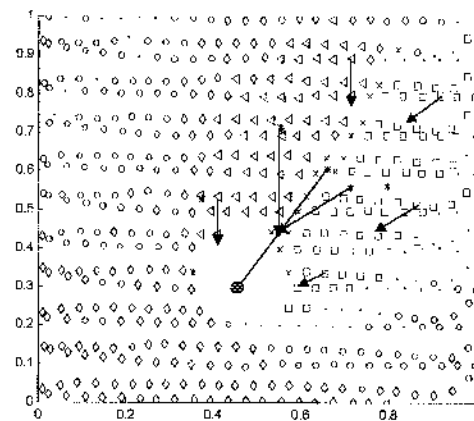


Fig. 10 NN calculated hole center direction vectors based on actual sensor feedback. Note: Where diamonds and circles are indicated sensory cues are unreliable, whereas x's indicate high-information sensory data.

The above methods are promising for generating sensory-driven soft attractor trajectories. In addition to the goal directed manipulation of an attractor point, it is also possible to change the end-point dynamics of the robot by altering the spring and damper coefficients. The alterations of these parameters during the assembly process enable complex capabilities, such as tracking a trajectory quite stiffly (e.g. to approach a high-confidence target location) followed by a sudden relaxation of stiffness or damping parameters (e.g. to accommodate contact constraints while searching for an uncertain assembly location), as required by a particular application.

Such work in sensory-driven, behavior-based control may provide the foundation for realizing the third important quality of expert assembly exhibited by humans: the ability to improve performance through experience. It is our hope that our intelligent robot controller will be capable of autonomous data collection and autonomous neural-net training for automatic generation of programs for new assemblies. Such capability would eliminate the need for expensive, time-consuming robot programming and would enable robots to acquire expertise through experience

## CONCLUSION

Attributes once thought only to belong exclusively to humans have now been demonstrated on the NIST FRAPA project using NAC robot controllers, vision, and neural networks. A large class of applications that have resisted automation due to positional uncertainty being greater than assembly tolerances, and the need to control the forces of contact of workpieces manipulated by robotics are now feasible.

In the near future robots will be commonly available that have the capacity to control their forces of contact, acquire and manipulate parts in uncertain locations and poses, and use trained neural networks to accomplish a predefined goal.

### Acknowledgements:

We thank NIST for providing the FRAPA Joint Venture with the ATP award. Matt Collins of Perception for his work in 3-D vision. David Kozlowski from Sandia National Lab. The other members of the FRAPA Steering Committee: Connie Philips of the NCMS; Bob Stoughton of MicroDexterity Systems; Mike Yoshonis of PICO; Valerie Bolhouse, Frank Maslar, Steven Sparkes of Ford Motor Company. Contributors at CWRU are Ravi Hebbar and Dan Morris on low level controls, Yonghong Zhao, Craig Birkhimer and Prof. Yoh-Han Pao on force-guided strategies, and Prof. Michael Branicky, Siddharth Chhatpar, Yuesong Wang, and Ittichote Chuckpaiwong.

## References

1. Robotic Industries Assoc. 1999
2. Whitney, D. 1987, "Historical Perspective and State of the Art in Robot Force Control," *International Journal of Robotics Research*, pp. 3-14, vol. 6, no.1.
3. Newman, W. S. and Mathewson, B.B., "Integration of Force Strategies and Natural admittance Control", Proceedings from 1994 International Mechanical Engineering Congress and Exposition, Vol. 1 of 2, Nov. 1994
4. Newman, W.S. and Y. Zhang, "Stable Interaction Control and Coulomb Friction Compensation Using Natural Admittance Control". *Journal of Robotic Systems*, 1994. 11(1).
5. Newman, W.S., "Stability and Performance Limits of Interaction Controllers," *ASME J. Dynamic Systems, Measurement and Control*, 1992. 114(4): p. 563-570.
6. Pipitone, Frank and Adams, William, "Rapid Recognition of Freeform Objects in Noisy Range Images Using Tripod Operators", Navy Center for Applied Research and Artificial Intelligence (NCARAI), AIC-93-037, 1993