Modeling Human Cognition Using a Transformational Knowledge Architecture

Stuart H. Rubin[1], Gordon Lee[2], Witold Pedrycz[3], and Shu-Ching Chen[4]

[1]SPAWAR - Systems Center
53560 Hull Street
San Diego, CA 92152

[2]Dept. of Electrical and Computer Engineering
San Diego State University
5500 Campanile Drive
San Diego, CA 92182

[3]Dept of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta,
Canada T6G 2R3

[4]School of Computing and Information Sciences
Florida Int'l University
11200 SW 8th St, ECS 362
Miami, FL 33199

Abstract - *While much research has been devoted to learning and machine intelligence, the field is still in its infancy. In particular, a technology that will allow for heuristic exploitation of information domain regularities to reduce the time required for knowledge acquisition while concomitantly resulting in an increase in the reliability of the acquired knowledge is still lacking. Unfortunately, contemporary learning mechanisms such as neural network architectures are inherently incapable of such performance. The objective of this paper is to present a new way of looking at learning and machine intelligence which has applicability in many fields such as in robotics, intelligent agents, data fusion, and cooperative sensing. In particular, we propose to construct a new architecture, that is, a transformational architecture for learning, intelligent fusion and transference of knowledge. A System of Systems (SoS) approach is used to realize machine intelligence. Random differences are learned by the system, generalized, and made available for subsequent replay in design transformations. Cross-domain symmetries can play a major role in design generation in particular and in the design of SoSs in general. The fundamental theory of randomization is the science, which underpins the practice. This strategy is employed in the design of the Knowledge Amplification by Structural Expert Randomization or KASER system.*

Keywords: intelligent learning systems, KASER, soft expert system

1 Introduction

Intelligence usually implies the capability of a system to adapt to changes and uncertainties and the ability to gather information in order to perform certain tasks. However, the development of learning algorithms for machine intelligence that increase the reliability of the induced knowledge while simultaneously reducing acquisition time is still a challenging problem. The Incompleteness Theorem and Randomization Problem dictate a need for such a new approach to information fusion, the backbone for machine intelligence. It follows from the *Unsolvability of the Randomization Problem* [1] that a capability for learning must be included if the system is to scale. *Randomization*, first coined by Gregory C. Chaitin in 1975 [1], infers that the computer does what it does best (serves as the search engine) while the user does that which he or she does best (serves as the 'conceptual engineer'). Such a human-machine symbiosis lies at the cutting edge of software engineering.

The opposite of randomness, in Chaitin's sense of the word, is *symmetry*. That is, if a pair of objects is not mutually random, then they are mutually symmetric. For example, two fruits such as an apple and an orange are mutually symmetric whereas, a fruit and a rock are mutually random in most contexts.

A network architecture can be employed to mine knowledge from information. Such knowledge can even be applied to the construction of extensible intelligent search engines, which further serve to put all this new knowledge at ones fingertips.

For example, consider the neural network. Feed-forward nets have been studied since the early days of the perceptron [2]. However, closed-loop nets, which emulate neural engrams, have not been studied as thoroughly as should be [3]. Such nets cannot be mathematically studied in detail as a consequence of the Incompleteness Theorem. Yet, even here there is randomization at work. Each neuron temporally and spacially sums thousands of inputs to yield one output signal, which is either excitatory or inhibitory. This many-to-one mapping defines a randomization, which necessarily occurs in a system of randomizations driven by feedback. Experiments have shown that such systems converge. One can thus compose an ensemble of networks where the conventional neural network weights are replaced by a symbolic set using a domain-specific modeling language.

978-1-4244-2173-2/08/$25.00 ©2008 IEEE

2 Randomization as a measure of intelligence

An intelligent software system interacts with the user in two principal ways. First, it requests random knowledge be supplied where necessary. Second, it asks the user (knowledge engineer) to confirm symmetric knowledge, where presented. Note that supplying a selection from a pull-down menu is partially random and partially symmetric in its component tasks.

Clearly, if a user can supply the requested random or symmetric knowledge, then it is through the application of acquired knowledge. It follows that if that knowledge can be captured in a knowledge-based system, then the requested tasks can be automated. What is being claimed here is that a network of cooperating grammar-based systems requests knowledge that is random in proportion to the size of its collective randomized knowledge bases.

As more and more knowledge bases are linked in a network, then the knowledge needed becomes more and more random. A consequence of Gödel's Incompleteness Theorem is that countably infinite truths are recursively enumerable, but not recursive [1].

Of course, the world is neither totally random, nor totally symmetric. Indeed, this follows from Gödel's Incompleteness Theorem. The degree of symmetry increases with scale. Were this not the case, then the universe would be random in the limit.

In other words, the degree of randomization possible is in proportion to the magnitude of the information, where there is no upper bound. Also, the processor time required to randomize such information is unbounded. Such absolute minimal entropy can of course never be achieved, for it would violate the Incompleteness Theorem. However, there is nothing to preclude the construction of randomizing systems of any desired level of utility. For the KASER, the utility of randomization is integral in learning to abstract design principles for a Systems of Systems (SoS).

3 Evolutionary transformation

The question as to which is better, or which came first, component randomization, or knowledge-based compilation is equivalent to the question: which is better bottom-up randomization or top-down randomization? It is clear that component randomization came first because just as was the case in evolution, there was no higher-level knowledge bootstrap initially available. Then, it follows that knowledge-based randomization evolved from component randomization. This means that components evolved in two simultaneous directions. First, they continued to evolve as domain-specific components (*horizontal* randomization). Second, some components mutated into transformative software (*vertical* randomization). Transformative software evolved just as enzymes evolved in the course of biological evolution. Such software can indeed be captured by the grammar. However, when active it acts like a rule or knowledge-base segment and applies domain-specific knowledge to further randomize the grammar.

This means that context-sensitive transformations of components can be made through spatial and/or temporal optimizations. Transformative components are saved in the single grammar in the form of rules, which are even capable of randomizing each other, just as they randomize their containing grammar! That is, the answer, which is sought is that the grammar must randomize components, some of which become active in the form of transformation rules -- again in the same grammar. These rules effect transformations on themselves. It follows that the Incompleteness Theorem applies, which implies that a countably infinite number of novel suggestions can be made by the grammar, which while ultimately true cannot be proven. Such a technique has promise from an engineering perspective.

4 Transformative components

Transformational components need to be represented in a context-free grammar in such a manner as to be context sensitive in their use. For example, one software design may require a quicksort component because it involves a large data set, while another may require an insertion sort because it does not. Clearly, the suggested component can never converge in the absence of context. Furthermore, it is only appropriate to substitute one component for its siblings when that component is provably better. Such is the case when, for example, one component has been run through an optimizing compiler and the other has not.

The user can *mark* components with far less effort than is required to program code. Such components then serve as foci for randomization. Interaction with the user can serve to further randomize this model. Randomization involves the substitution of existing component definitions into grammar strings. Clearly, this technique involves a heuristic search because the order of substitutions is critical.

5 The Inference Engine

Expert systems, and in particular blackboard architectures, provide a proven pragmatic framework for the capture and exercise of domain-specific knowledge.

Here, knowledge is captured by the rule base in the form of situation → action rules. The agenda

mechanism determines which rule to fire when. Typically, a "most-specific" first ordering is applied. In this manner, a more-specific rule can be acquired, which will override any more general rule – while the more general rule will remain to be fired where appropriate. This defines the inference engine. The inference engine typically runs in interpreted mode, or applies the Rete Algorithm where the rule base is relatively static. However, the rule base will not be static for the purpose investigated here.

The purpose of the inference engine is to select the next rule to be fired, if any, and update the blackboard, if applicable. Here is a high-level view of the applicable algorithm is as follows:

1) A most-specific match is sought for the context. Initially, this is done for the lowest subclass (i.e., the primitives). The most-specific question (i.e., if any), along with its previous response, is always first to be presented upon a match. The user or knowledge engineer may subsequently generalize this as necessary.

2) If no specific match can be found, then superclass (i.e., generalize) the instance. One can generalize the context from the right to the left; that is, from the most recent predicate to the least-recent predicate. Scan the entire grammar for a match before generalizing the next predicate. Again, seek a most specific match (i.e., longest predicate sequence).

3) Subclasses serve to delimit the search for a predicate when manually browsed. When automatically matching the context, they serve to delimit and otherwise order the space of functions that need be tested. Such processes could not be practically threaded in the absence of such constraints.

4) A special symbol(s) exists at the top of each generalization hierarchy, which provides the user or knowledge engineer with a means to insert knowledge (i.e., a new class or superclass) if nothing relevant is known. It is analogous in function to the idle process of an operating system.

6 Transformational grammars

The concept of randomization has previously been shown to reduce the effort required to write software (or generate high-level designs), while concomitantly improving the quality of the resulting code. Context-free grammars are inherently capable of randomizing, but only within the confines of a certain logical regimen. To achieve a greater degree of randomization, one must step outside of those confines. It is clear that when one 'steps outside' of these confines, one is dealing with randomizing the representational formalism itself -- not merely what is represented in that formalism.

If a component is capable of transforming other components, it is said to be *active*. The same component can be passive in one situation and active in another. Active components can also transform each other and even themselves. In fact, two interesting observations can be made at this point. In the case of biologically-based organisms, the definition of life is one such property. It used to be held that something was alive if it was capable of reproduction.

Grammars that consist entirely of passive components allow for the design of relatively inefficient components. This follows because the user may select a locally correct component definition in a pull-down menu; but doing so, the user is unaware that the resulting global design will be inefficient if that component is used in the local context. The inclusion of active components provides for the capture and reuse of the users expressed knowledge of optimization. Such optimizations can be applied to active components -- including, at least in theory, self-referential optimization. Thus, there need be no attendant inefficiency of scale if the grammar includes active components.

Clearly, randomization is ubiquitous. It must then be the case that the proper space-time tradeoff is defined by the domain to which the system is to be applied. Again, such systems are necessarily domain-specific in keeping with the dictates of the halting problem in computability theory. For example, a web address repeater should be composed entirely of passive components. On the other hand, a search for the deepest most random knowledge must be conducted by pure chance, strange though as it must sound. The great majority of other systems fall between these extremes.

7 Implementation of the KASER

To test the approach presented in this paper for machine learning, a KASER architecture was developed (see Figure 1) and is currently being implemented. "KASER" stands for "Knowledge Amplification by Structural Expert Randomization". A KASER facilitates reasoning using domain specific expert and commonsense knowledge. It accomplishes this through object-classed predicates and an associated novel inference engine. It addresses the high cost associated with the knowledge acquisition bottleneck. It also enables the entry of a basis of rules and provides for the automatic extension of that basis through domain symmetries.

KASER systems can be classified as Type I and Type II, depending on their characteristics. In a Type I KASER, words and phrases are entered through the pull-down menus. The user is not allowed to enter new words or phrases if an equivalent semantics already exists in the menu. In a Type II KASER, distinct syntax may be equated to yield the equivalent normalized semantics. The idea in a Type II KASER is to ameliorate the

inconvenience of using a data entry menu with scale. In a Type II KASER, selection lists are replaced with semantic equations from which the list problem is automatically solved.

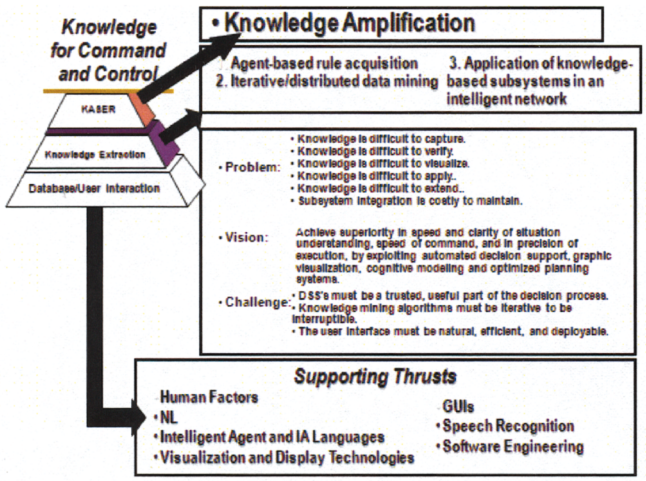

Figure 1: The KASER Architecture

An example of the need for KASERs in feature synthesis via feedback will serve to clarify their role. First, why the need for features? The answer is not that they help the system to perform better; rather, a mining/fusion system can not even begin to perform in the absence of features. Consider for example the mining of a chess game. If all that is recorded are the positions of the pieces on the board (i.e., a feature-less system), then any mined rule will necessarily be so specific (i.e., literal) that it will almost always fail to be matched or be replayed. However, if we mine features that suggest, say, the value of moving one's king to the center of the board, using depth-first search (DFS), or breadth-first search (BFS) as feature-based search paradigms to determine a proper move (i.e., to be contrasted with the literal replay of a saved move), then clearly the applicability problem will have been largely abated. It is empirically established that the need for a feature-based representation is essential [9].

Next, as one might imagine, it is not computationally easy to find effective features. There are just too many candidates and even supercomputers cannot possibly search through all combinations. Deduction fails too because it presumes that one has the most general feature set to begin with, which clearly is not the case. That leaves us with computational analogy. Domain-specific features such as this one are found through a combination of select domain representation [9] and domain-specific knowledge. Often the latter must evolve and the requirements for this evolution vary with the domain.

Observe though that a KASER can take a known domain-specific feature and suggest computationally similar features. These candidate features are then fed into the mining system to see if they are good predictors (i.e., in combination with other known or predicted features). A complex evolutionary loop can then be set up that can *predict future events with ever-increasing (but never absolute) accuracy*. The results can far outperform neural networks, genetic algorithms, case-based reasoners, and the like. Actual measurements of a KASER's capability for creating analogous knowledge (via the transformation of experiential knowledge) are given in Figure 2 for automotive diagnosis and are based on about 140 trials per vehicle – proceeding temporally in training from left to right [7].

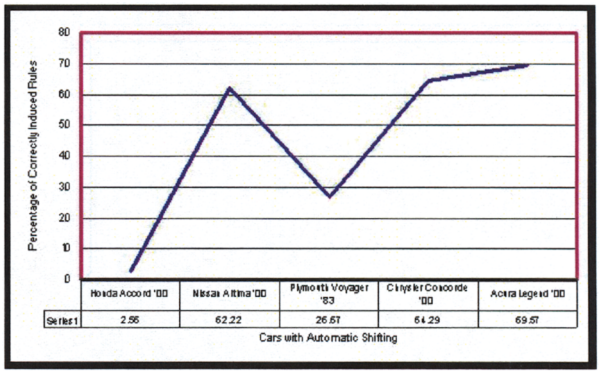

Figure 2: Knowledge Transference and Supra-Linear Learning in a KASER Applied to an Automotive Diagnostic Domain.

The observed "dip" is due to the fact that the 1983 Plymouth Voyager is diagnostically random to a relative degree (i.e., when first encountered) in comparison with its two predecessors. Note that the 70 percent asymptote seen here is domain dependent.

One of the major problems with other approaches to achieving the aforementioned goals is simply that they cannot be coordinated for execution on massively concurrent distributed processors. Our approach is predicated on randomization theory ([1] and [6], for example) and thus need not suffer this fate. Knowledge discovery and control algorithms that can fully utilize over a million concurrent processors have unrealized potential that far exceeds those that can at best run on super VonNeuman architectures.

Another major problem with other approaches is that they take the ill-fated *General Problem-Solver* (GPS) approach because they cannot effectively bring domain-specific knowledge to bear on the problem. Our system's approach has the human-in-the-loop who, at any time, can insert domain knowledge into the KASER for replay in the context of discovering analogical knowledge (i.e.,

knowledge that is open under deduction). As a practical matter, it is much easier and faster for the user to insert domain knowledge into a KASER than it is to reconfigure the afferent representations and necessarily retrain neural network models as an evolutionary paradigm. Again, knowledge, in part, takes the form of features and makes for *strong* learning by our large information system. The only way to avoid such intractability, in practice, is to encode high-level feature representations into the domain. Such techniques are closely aligned with evolutionary programming (EP). We can do much better, though, through proper evolution and inclusion of domain-specific features.

The complexity of searches in a KASER can be delimited through the use of squelches. The classical data-mining problem of over-fitting the data is greatly ameliorated through the mining of feature-based representations because spurious relations are not captured by the features. The greater the use of features, the less the possibility of over-fitting the data. In other words, the claimed superior predictive capability of our information systems is upheld.

Agents can perform automated data conditioning that humans find boring or inefficient, and tasks of vigilance that humans do not do very well. They are not subjected to human conditions, such as boredom or fatigue. In mixed-initiative systems, the operator/analyst can suggest features that either must occur simultaneously or concurrently. Agents can be deployed through the user-friendly interface to look for features using, say, a network-centric Single Integrated Picture (SIP) and report back when certain pre-specified criteria have been met. Over time, the operator/analyst with the help of agents can increase the content and accuracy of the knowledge base.

The results can provide automated alerts and operator oversight; data mining algorithms can automatically extract features from message-level databases that contain refined sensor and track data as well as various features from existing systems. These features sets may contain many dimensions – both physical and procedural, such as space, time, frequency, and constraints, together can constitute patterns of behavior. Patterns can be saved in a part of the KASER knowledge base designed especially to preserve salient features associated with situations of interest or threats. The operator/analyst can query the KASER regarding the relationships between features, their dependencies and constraints. The operator/analyst also can provide feedback and additional data to refine the knowledge base. Moreover, the operator/analyst can have control of the mining algorithms through a user-friendly interface.

8 Conclusions

The goal of machine learning is to tractably and reliably acquire specific situational and commonsense knowledge for use in problem-solving systems. KASER architectures can replace the conventional learning systems with a symbolic model using a domain-specific modeling language. The model captures features, which provide for accelerated learning, improved predictive accuracy, and explanative potential. The architecture has many advantages over conventional learning mechanisms – including the characteristics that it is self-organizing, is far less sensitive to noise and missing information, and computation is not serially bottlenecked. knowledge can be effectively transferred from one domain to another. The transference process will necessarily involve machine learning because it is inherently incomplete. However, simple learning on a case-by-case basis will make for an intractable learning system with scale (e.g., SoS). That is why it is necessary to introduce symmetry into the learning process – reflecting on previous design knowledge. That randomness and symmetry are both necessary was made clear by the above example and follows from the theory of randomization, which was first exemplified by the KASER [7]. Moreover, this paper should also make it clear that the issue of representation permeates any system design [9].

Randomization and symmetry are core AI concepts. Other papers have shown that they serve as a basis for much of machine learning [5]-[7]. In this paper, we have seen that machine learning, which is predicated on randomness and symmetry can serve well in the design of SoSs.

This paper provides the development of a computational intelligent architecture using randomization. It is well-known that the information revolution is destined to become a knowledge revolution, which in turn is pre-ordained to become a never-ending search for randomizations in the sense made clear herein. A metaphorical information-theoretic black hole is the ultimate result in keeping with the dictates of the Incompleteness Theorem. Given this, there is an inescapable conclusion, namely, the development of a computational intelligence is inevitable and has just barely begun. A transformational knowledge architecture may provide the characteristics needed for this development.

9 References

[1] G.J. Chaitin, "Randomness and Mathematical Proof," *Scientific American*, Vol. 232, 1975, pp. 47-52.

[2] R.L. Harvey, Neural Network Principles, NJ: Prentice-Hall Inc., 1994.

[3] J. Murthy and S.H. Rubin, "Multi-Sensor Fusion for Photonic Realization," to appear in the *Proc. 1st Ann. Infor. Integration and Reuse Conf.*, Atlanta, GA, 1999.

[4] S.H. Rubin, "A Typogenetic Application of Random Seeded Crystal Learning," *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Chicago, IL, 1992, pp. 283-289.

[5] S.H. Rubin, "A Heuristic Logic for Randomization in Fuzzy Mining," *J. Control and Intelligent Systems*, vol. 27, no. 1, pp. 26-39, 1999.

[6] S.H. Rubin, "On the Auto-Randomization of Knowledge," *Proc. IEEE Int. Conf. Info. Reuse and Integration*, Las Vegas, NV, pp. 308-313, 2004.

[7] S.H. Rubin, S.N.J. Murthy, M.H. Smith, and L. Trajkovic, "KASER: Knowledge Amplification by Structured Expert Randomization," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 34, no. 6, pp. 2317-2329, December 2004.

[8] Rubin, S. and Lee, G. "On the Use of Randomization for System of Systems (SoS) Design of Intelligent Machines", Proc. of the World Automation Congress, ISSCI, Budapest, 2006.

[9] S. Amarel, "On Representations of Problems of Reasoning about Actions," *Mach. Intelligence*, vol. 3, pp. 131-171, 1968.