KASER: A Qualitatively Fuzzy Object-Oriented Inference Engine

Stuart H. Rubin and Robert J. Rush, Jr.
Space and Naval Warfare Systems Center
San Diego, CA 92152 USA
{srubin, rushr}@spawar.navy.mil

Jayaram Murthy
Central Michigan University
Mt. Pleasant, MI 48859 USA
murthy@cps.cmich.edu

Michael H. Smith
University of California
Berkeley, CA 94720 USA
mhs@mining.ubc.ca

Ljiljana Trajkovic
Simon Fraser University
Burnaby, BC V5A 1S6 Canada
ljilja@cs.sfu.ca

Abstract

This paper describes a shell that has been developed for the purpose of fuzzy qualitative reasoning. The relation among object predicates is defined by object trees that are fully capable of dynamic growth and maintenance. The qualitatively fuzzy inference engine and the developed expert system can then acquire a virtual-rule space that is exponentially (subject to machine implementation constants) larger than the actual, declared-rule space and with a decreasing non-zero likelihood of error. This capability is called knowledge amplification, and the methodology is named KASER. KASER is an acronym for Knowledge Amplification by Structured Expert Randomization. It can handle the knowledge-acquisition bottleneck in expert systems. KASER represents an intelligent, creative system that fails softly, learns over a network, and has enormous potential for automated decision making. KASERs compute with words and phrases and possess capabilities for metaphorical explanations.

1. Introduction

First-generation expert systems are production systems where the knowledge base and inference engine are disjoint. Second-generation expert systems are defined as systems with a rudimentary capability for learning. Here, learning can be performed through the un-automated interpretation of grids or by user query. Third-generation expert systems go one step further [1]. They provide for learning by the rule-base through deductive and inductive processes.

Consider the theory of randomization [2]-[6] that entails reducing information to its simplest form through compression techniques. Deduction represents the most common form of information compression. In other words, why store all instances of a knowledge predicate when they are subsumed by a common generalization? Similarly, induction represents a fuzzy form of information compression. Here, while an instance of a knowledge predicate is saved, one tries to extend the scope of that instance through generalization operations. Unlike the case for deduction, error is inherent to inductive processes. Nevertheless, when errors are corrected, the effect propagates to include other predicates that build upon the corrected one [7].

Chaitin [2] first published the *theory of randomization*, which may be seen as a consequence of Gödel's Incompleteness Theorem [8]. Essential incompleteness precludes, in principle, the construction of a universal knowledge base that need employ no search other than referential search. Lin and Vitter [9] proved that to be tractable, learning must be domain specific. The fundamental need for domain-specific knowledge is consistent with Rubin's proof of the unsolvability of the randomization problem [5]. In this paper, we introduce the concept of knowledge amplification. Production rules are expressed in the form of situation-action pairs. When these rules are discovered to be in error, they are corrected through acquisition. Conventionally, a new rule must be acquired for each correction. This is called linear learning.

The acknowledged key to breakthroughs in the creation of intelligent software is overcoming the knowledge-acquisition bottleneck [10]. Learning "how to learn" depends fundamentally on representing the knowledge in the form of a "society of experts." Minsky's seminal work led to the development of intelligent-agent architectures [11]. Furthermore, Minsky [12] and Rubin [5] independently provided compelling evidence that the representational formalism itself must be included in the definition of domain-specific learning if it is to be scalable.

KASER stands for "knowledge amplification by structured expert randomization." KASER has a graphical

user interface (GUI) that features pull-down menus and modes for both end users and expert users. It is an object-oriented design. Unlike conventional intelligent systems, KASER is capable of accelerated learning in symmetric domains [2]. Symmetric domains are those that can be represented in more compact form.

Figure 1 shows plots of knowledge acquired by an intelligent system vs. the cost of acquisition. Conventional expert systems will generate the curve below the "break-even" line because with conventional expert systems cost increases with scale and is never better than linear. In comparison, the cost of KASER decreases with scale and is always better than linear unless the domain is random [2]. In practice, all non-trivial domains have some degree of randomness and some degree of symmetry [3]. The more symmetric the operational domain, the less the cost of knowledge acquisition and the higher the curve appears in the graph shown in Figure 1. The virtual-rule space (all possible rules) is always much larger than the declared or actual- rule space.

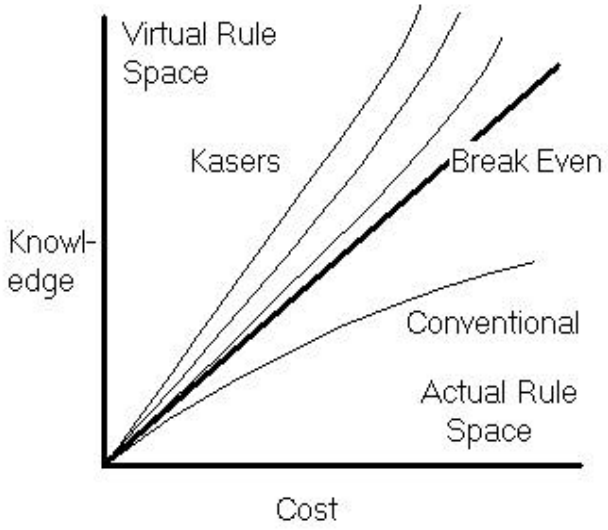

Figure 1. Comparative costs of knowledge acquisition.

The paper is organized as follows. In Section 2, we describe object-oriented design features. Section 3 explains the use of objects in the user interface. Section 4 addresses interest in future research and commercial applications. The paper is summarized in Section 5.

2. Declarative Object Trees

KASER operates in a manner similar to a conventional expert system, except for the representation of knowledge, the inference engine, and various other items that are affected as a consequence.

Figure 2 depicts an example of contextual specification for qualitatively fuzzy pattern matching in KASER. We depict here auto-repair knowledge. Nevertheless, because KASER is a shell, the operational domain has few limits. The more symmetric the domain of interest, the more creative the KASER tool can be. Thus, KASER possesses no inherent advantage when compared to a conventional expert system in the acquisition of random historical information, whereas, they can be far more applicable to symmetric domains such as mathematics or chess. In symmetric domains, the system is far more likely to induce correct new knowledge. Of course, no non-trivial domain is entirely random or entirely symmetric [3].

Declarative-object trees are used as follows. The antecedent tree presents instances of an object with increasing depth. The degree of instantiation or generalization is determined by a tree search that is limited by a settable squelch. Action-oriented knowledge is hierarchically arranged in the consequent tree. Additional knowledge, if available, will lead to an instance of that object. For example, the facts that the sky is cloudy and the barometric pressure is falling, will lead to the forecast that precipitation is likely. Then, given that the temperature is below the freezing point of water, the proper instance of precipitation will be snow.

The use of objects in the consequent tree provides for the maximal reuse of the contained information. Consequents may be selected as a sequence of actions. Contexts may be specified through the selection of a conjunction of objects from the antecedent menu (Figure 2). Although Figure 2 depicts the KASER tool using words and phrases, KASER also can run on object-oriented functions and procedures.

Declarative-object knowledge is acquired through interactions with a knowledge engineer. In general, this is an on-going process because a non-trivial knowledge base is never complete. Just as one can write a simple program and proceed to enhance it, so too can one evolve the declarative object trees. The more evolved the trees, the more accurate the induced knowledge for a given domain having a fixed symmetry factor. Thus, a highly developed tree can increase the dependability of KASER. For example, the conceptual object, "apple" may be evolved into at least two object subclasses: "red apple" and "green apple", depending on the application.

Our KASER tool contains tools for copying objects, moving objects, pasting (single objects), deep pasting (copying complete sub-trees), finding object nodes, and suggesting next nodes based on k = 1 level of look-ahead. For example, if one is developing an on-line pre-flight checklist, after entering "check the wing flaps," the system will look ahead one step and suggest checking the rudder. This helps prevent omissions in complex contextual specifications.

3. Object-Oriented Translation Menus

KASER requires that declarative knowledge be compiled dynamically in the form of object-oriented hierarchical

phrase-translation menus. Thus, each class (antecedent and consequent) of bipartite predicates can be interrelated through their relative positions in an object-oriented semantic tree. A declarative knowledge of such relationships provides a basis for common-sense reasoning. In this Section we describe the creation, maintenance, and use of the object-oriented trees.

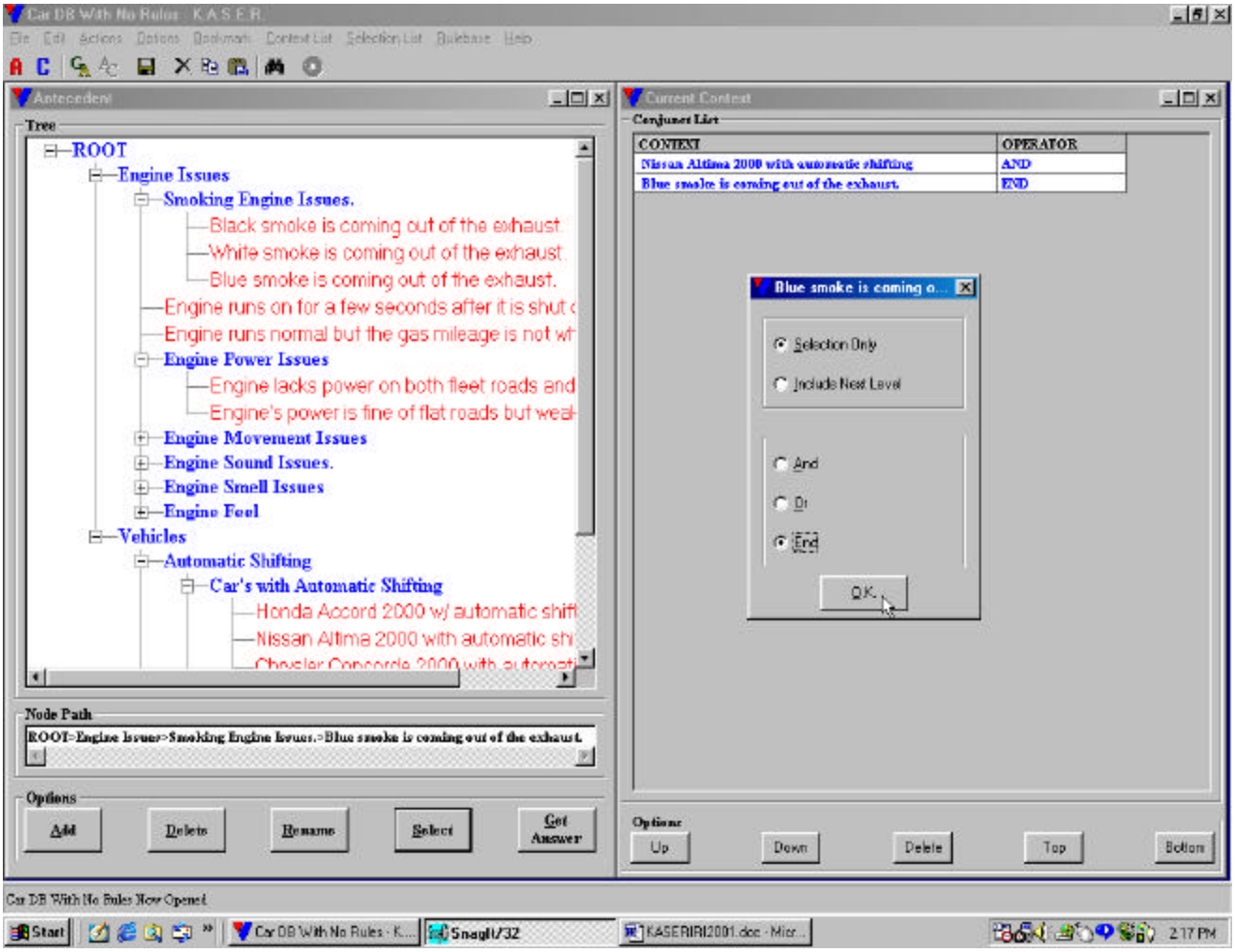

Figure 2. Contextual specification for a qualitatively fuzzy test.

1. The *phrase-translation menus* serve as an intermediate code (like in a complier), where English sentences can be compiled into menu commands using rule-based compiler bootstraps. KASERs can be arranged in a network configuration where each KASER can add to or delete from the context of another. This will expand considerably the intelligence of the network with scale, and is consistent with Minsky's, "Society of Mind" [11]. Furthermore, the very-high-level domain-specific language(s) used to define each predicate can be compiled through a network of expert compilers.

2. Each *antecedent or consequent phrase* can be associated with a textual explanation, Microsoft's Text-to-Speech engine (Version 4.0), an audio file, a photo, and/or a video file. Images may be photographs, screen captures, scanned, or drawings. They may also be annotated using arrows and numbers.

3. *Antecedents and consequents* can be captured using an object-oriented approach. The idea is to place descriptive phrases in an object-oriented hierarchy such that sub-classes inherit all of their properties from a unique superclass and may include additional properties as well. Menus tend to grow and this leads to the recursive extraction of sub-menus. New phrases can be acquired at any level. Each operational KASER is saved as a distinct file that consists of the antecedent and consequent trees, the associated rule base, and possibly the multimedia attachments

Consider the partial path: office supply, paper clip, and the partial path: conductor, paper clip. Here, any subclass

of paper clip will have very different constraints depending on its derivation. For example, anything true of paper clips in the context of their use as conductors must hold for every subclass of paper clips on that path. Unique antecedent integers can be setup to be triggered by external rules. Similarly, unique consequent integers can be programmed to fire external procedures. All we need do is facilitate such hooks for future expansion (e.g., a radar-mining application domain). Each project is saved as a distinct file, which consists of the antecedent and consequent trees, the associated rule base, and possibly the multimedia attachments.

4. An *acyclic graph* structure is used because the structure needs to facilitate dynamic acquisition (random phrases) and deletion, which cannot be accomplished in the presence of cycles due to side effects. Note that entering a new phrase on a menu implies that it is semantically distinct from the existing phrases, if any, in the same menu.

5. A *tree structure* is mapped to a context-free grammar, where the mapping process needs to be incremental because of the large size of the trees. Each node or phrase is assigned a unique number that uniquely identifies the path by which it is located.

6. Each *phrase* may be tagged with a help file that also serves the purposes of the explanation sub-system.

7. Each menu should be limited to one screen of items (21 lines, in Figure 2). Therefore, objects should be sorted dynamically into distinct classes. That is, new submenus can be created dynamically and objects moved to or from them.

8. Three contiguous *levels of hierarchy* should be displayed on the GUI at any time, if available.

9. A biologically inspired "marker gene" or *bookmark concept* allows the user to set mark points for navigational purposes.

10. A list of *recently visited menus* caches navigational paths for reuse.

11. A *global find* mechanism allows the user to enter a phrase and search the tree from the root or present location and find all matches for the phrase down to a pre-specified depth. The path, which includes the phrase, if matched, is returned.

12. Entered *phrases* (including pathnames) can be extrapolated automatically where possible. This intelligent feature facilitates keyboard entry. It can also assist with the extrapolation of path names to facilitate finding or entering a phrase. Path name components may be truncated to facilitate presentation.

13. A major problem in populating a tree structure is the amount of typing involved. Thus, copy, paste, edit, and delete functions are available to copy phrases from one or more menus to another using place-holding markers. Phrase sub-menus are not copied over. Note that the returned list of objects still needs to be edited manually for error and/or omissions. This follows from randomization theory and also maps well to natural-language translation.

14. Alternates in a menu serve as analogs and super-classes serve as generalizations for an explanation sub-system. In addition, help files and path names will serve for explanative purposes. An "intelligent-assistance" feature allows the system to predict the next node in a contextual, antecedent, or consequent tree. Each node in a tree locally stores the address (number) of the node to be visited next in sequence. If a node has not been trained, or if the address to which it points has been deleted without update, a text box stating, "*No Suggestion*" pops up. Otherwise, potentially three contiguous menus are brought up on the screen, where the rightmost menu contains the addressed node. Navigation is accomplished by clicking on a "*Suggest*" button. Otherwise, all navigation is performed manually by default. The user can hop from node to node using just the suggest button without registering an entry. The use of any form of manual navigation enables a "*Remember*" button immediately after the next term, if any, is entered. Clicking on this enabled button will set the address to which the *previously entered* node pointed, to that of the *newly entered* node. This operation overwrites the old pointer and allows for changing the item selected within the same menu. If a node (Toyota) is deleted, all pointers to it may be updated to the parent class (gas laser menu), and so on up the tree (vehicle type menu).

15. A pull-down menu will enable one of two options: (1) "Always remember" (by default) and (2) "Remember when told." The "Remember" button is not displayed under option (1). The system always starts at the root node of the relevant tree.

The system does not use more than one point of context in determining where to go next using the intelli-assist feature because we found that a simpler system is of greater utility in practice. That is, any increase in accuracy is more than offset by the extra training time and the extra space required, and the fact that to retrain reliably the patterns of visitation in response to a dynamic domain environment will take a relatively long time.

4. Future Research and Applications

A screen capture of a KASER for diagnosing faults in a jet engine is presented in Figure 3. Here, observe that the general and specific stochastics are both one. This means, in the case of the general stochastic, that KASER needed to use one level of inductive inference to arrive at the prescribed action. Similarly, the specific stochastic indicates that one level of deduction was necessarily employed to arrive at this prescribed action. Contemporary expert systems would have not been able to make a diagnosis and prescribe a course of action as they need to be explicitly programmed with the necessary details. In other words, KASER is offering a suggestion here that is open under deductive process. Simply put, it

created new and presumably correct knowledge. Two level-0 (valid) rules are supplied by the knowledge engineer (*R1* and *R2* below) and used in conjunction with the declarative object trees to arrive at the new knowledge *R4*:

R1: If exhaust smoky and sound low pitched then check fuel injector for carbonization.

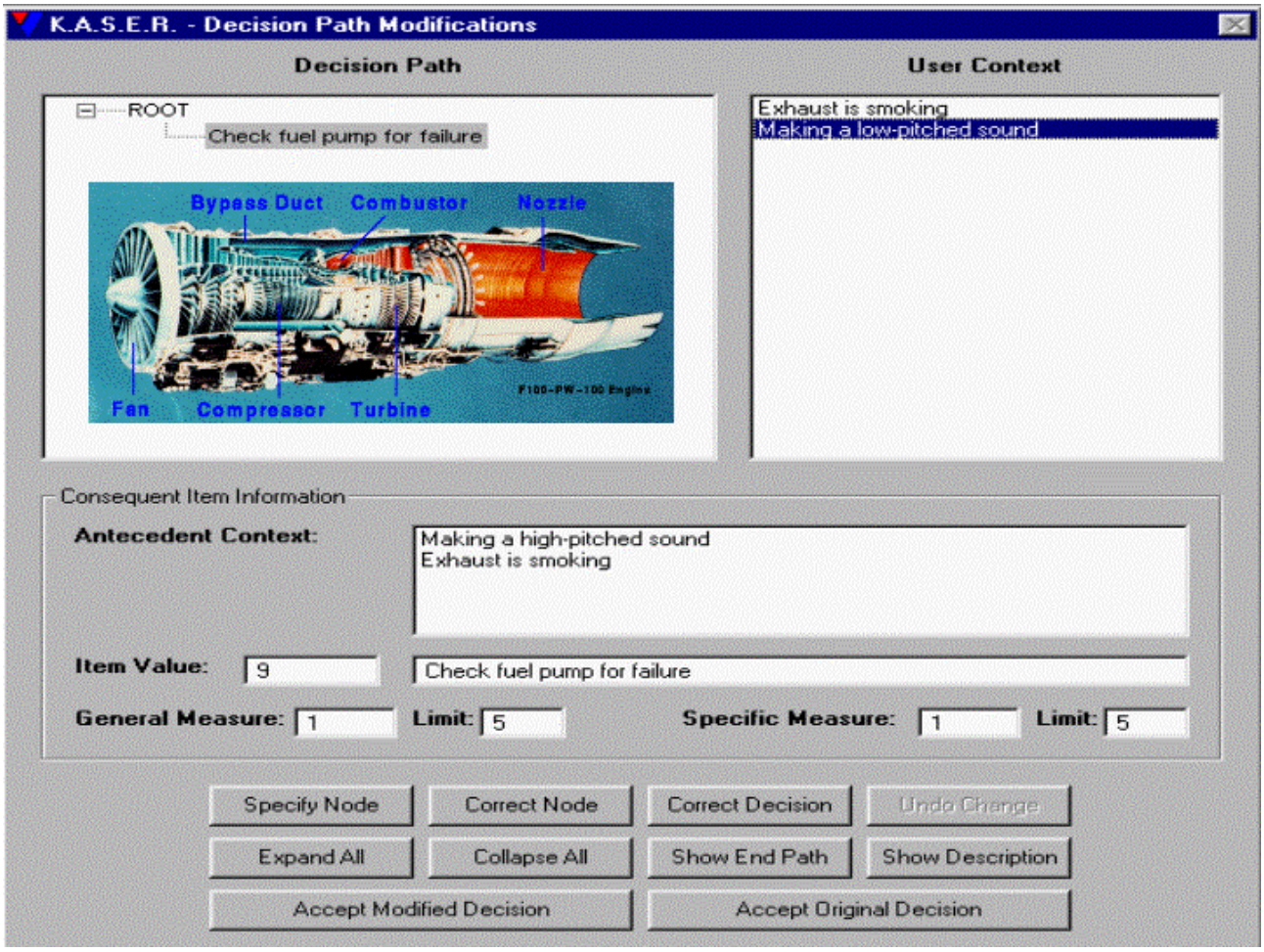

Figure 3. Screen capture of an operational KASER.

R2: If exhaust smoky and sound high pitched then check fuel pump for failure.

R3: If exhaust smoky and sound low pitched then check fuel pump for failure.

Upon confirmation of *R3*, *R2* and *R3* are unified:

R4: If exhaust smoky and sound not normal then check fuel pump for failure.

KASER finds declarative antecedent knowledge, which informs the system that the three sounds that an engine might make, subject to dynamic modification, are high-pitched, low-pitched, and normal sounds. By generalizing high-pitched sounds one level to the super-class *sounds* and then specializing it one level, one arrives at the first-level analogy – low-pitched sounds. This enables the user context to be matched and leads to the explicit instantiation of new knowledge. It generalizes rule consequents to maximize reusability. Object reuse may occur simultaneously at many levels, although this example depicts only one level for the sake of clarity. Many more algorithms, settings, and screens pertain to KASER than those detailed in this paper.

KASER's virtual-rule space automatically classifies the signatures and the generated stochastics provide an indication of reliability. KASER, having a virtual-rule space much larger than the declared-rule space, can

produce erroneous advice if the general stochastic is greater than zero. In this event, the user is requested to supply a corrective consequent(s) that may be "radioed" to the base computer for subsequent update on a daily basis, followed by uploading the more learned KASER. The main benefit here is that KASER can supply solutions to complex signature-identification problems that would not be cost effective to supply otherwise (Figure 1).

The agenda mechanism used in KASER differs substantially from that used in previous generations of expert systems. Future work can test tradeoffs between the number of antecedents matched to predicates and their level of generality. Also, since KASER cannot perform backward chaining due to an explosive virtual fuzzy space, work needs to be performed in the area of fuzzy programming forward chaining [7]. Such work would relax the level of specificity required of the predicate specifications, thus making rules easier to specify.

Work is underway to extend KASER capabilities through the use of associative memories and case-based reasoning. In brief, creative systems of ever-greater capability loom on the horizon as a boldly attainable goal. If successful, the impedance mismatch between the human and the machine will be further reduced.

5. Summary

The specific objectives of this project were to demonstrate: (1) a strong capability for symbolic learning, (2) an accelerating capability to learn, (3) conversational learning (learning by asking appropriate questions), (4) an analogical explanation subsystem, (5) probabilistically ranked alternative courses of action that can be fused to arrive at a consensus that is less sensitive to occasional errors in training, and (6) a capability to enunciate responses.

Randomization theory holds that the human should supply novel knowledge exactly once (random input) and the machine should extend that knowledge by capitalizing on domain symmetries (expert compilation). In the limit, novel knowledge can be furnished only by chance itself. Thus, future programming will become more creative and less detailed and, as a consequence, the cost per line of code will decrease rapidly.

In conclusion, the knowledge-acquisition bottleneck has been broken. KASER offers a new and more creative type of expert system. This third-generation expert system computes with words and employs qualitative fuzzy reasoning.

KASER may be said to fail softly and, for this reason, is not brittle. KASERs cannot be realized through the exclusive use of predicate logic because of their inductive component. Furthermore, KASERs can capture and otherwise represent virtually all of the verbally accessible domain knowledge possessed by a knowledge engineer. KASERs can predict their own likelihood of error, a feature that contributes to their dependability. They reason through a gradient descent or ordered search algorithm that minimizes this error.

KASER exhibits domain transference and supra-linear learning, in proportion to the inherent degree of domain symmetry. It introduces declarative object trees into expert systems. Such trees facilitate the capture of object-oriented semantic relations among rule predicates and thus serve the processes of analogical explanation as a consequence.

Acknowledgments

The authors thank the Office of Naval Research for their support of this research.

References

[1] S.H. Rubin, J. Murthy, M.G. Ceruti, M. Milanova, Z.C. Ziegler, and R.J. Rush Jr., "Third Generation Expert Systems," *Proc. Second Annual. Information Integration and Reuse Conf.*, pp. 27-34, 2000.

[2] G.J. Chaitin, "Randomness and Mathematical Proof," *Sci. Amer.*, vol. 232, no. 5, pp. 47-52, 1975.

[3] S.H. Rubin, "New Knowledge for Old Using the Crystal Learning Lamp," *Proc. 1993 IEEE Int. Conf. Syst., Man, Cybern.*, pp. 119-124, 1993.

[4] L.A. Zadeh, "Fuzzy Logic = Computing with Words," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 2, pp. 103-111, 1996.

[5] S.H. Rubin, "Computing with Words," *IEEE Trans. Syst. Man, Cybern.*, vol. 29, no. 4, pp. 518-524, 1999.

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

[7] S.H. Rubin, "On Knowledge Amplification by Structured Expert Randomization (KASER)," *Space and Naval Warfare Systems Center, San Diego Biennial Review*, 2001.

[8] V.A. Uspenskii, *Gödel's Incompleteness Theorem*, Translated from Russian. Moscow: Ves Mir Publishers, 1987.

[9] J.-H. Lin and J.S. Vitter, "Complexity Results on Learning by Neural Nets," *Mach. Learn.*, vol. 6, no. 3, pp. 211-230, 1991.

[10] E.A. Feigenbaum and P. McCorduck, *The Fifth Generation*. Reading, MA: Addison-Wesley Publishing Co., 1983.

[11] M. Minsky, *The Society of Mind*. New York, NY: Simon and Schuster, Inc., 1987.

[12] C.T. Clark, "Shoulders of a Giant," *Knowledge Management*, pp. 26-28, June 2000.