A Soft Expert System for the Creative Exploration of First Principles of Crystal-Laser Design

Stuart H. Rubin, Robert J. Rush Jr.,
Michael H. Smith, S.N. Jayaram Murthy, and Ljiljana Trajkovic[1]

ABSTRACT

A soft expert system is defined to be one that is qualitatively fuzzy. In this paper, we present such a system known as "KASER" which stands for 'Knowledge Amplification by Structural Expert Randomization". 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.

We demonstrate an application for KASER in the design of an intelligent tutoring system that teaches the basic science of crystal-laser design. It enables the student to experiment with various design concepts and receive feedback on the functionality of the proposed design. This is possible without a need to preprogram all possible scenarios.

Keywords: Soft Expert Systems, KASER, Knowledge Acquisition, Intelligent Tutoring Systems

1. INTRODUCTION

The theory of randomization was first published by Chaitin and Kolmogorov [1] in 1975. Their work may be seen as a consequence of Gödel's Incompleteness Theorem [2] in that it shows that were it not for essential incompleteness, then a universal knowledge base could, in principle, be constructed – one that need employ no search other than referential search. Lin and Vitter [3] proved that learning must be domain-specific to be tractable. The fundamental need for domain-specific knowledge is in keeping with the Unsolvability of the Randomization Problem [4]. This paper introduced the concept of knowledge amplification. Production rules are expressed in the form of situation $\rightarrow$ action. Such rules, once discovered to be in error, are corrected through acquisition. Conventionally, a new rule must be acquired for each correction. This is linear learning.

The acknowledged key to breakthroughs in the creation of intelligent software is resolving the knowledge acquisition bottleneck [5]. Learning how to learn is fundamentally dependent on representing the knowledge in the form of a society of experts. Minsky's seminal work here led to the development of intelligent agent architectures [6]. Furthermore, Minsky [7] and Rubin [4] 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.

A soft expert system is defined to be one that is qualitatively fuzzy. In this paper, we present such a system known as "KASER", which stands for 'Knowledge Amplification by Structural Expert Randomization". 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. Finally, the user supplies declarative knowledge in the form of a semantic tree using single inheritance. Unlike conventional intelligent systems, KASERs are capable of accelerated learning in symmetric domains.

In Section 2, a method is presented for a production system to induce property lists. Next, in Sections 3 and 4, randomization techniques are discussed. Then, in Section 5, an application of KASER to the design of an intelligent tutoring system is presented. Finally, in Sections 6 and 7, our conclusions and proposed future work are discussed.

2. INDUCING PROPERTY LISTS

We will define a production system as discussed in this paper as one which can automatically acquire a virtual rule space that is exponentially larger than the actual rule space with an exponentially decreasing non-zero likelihood of error. Moreover, the generalization mechanism will not only be bounded in its error, but other than for a straightforward user-

[1] SPAWARS SYSTEMS CENTER, 53560 Hull Street, San Diego, CA USA srubin@spawar.navy.mil, rushr@spawar.navy.mil. BISC Program, University of California, Berkeley, CA USA, m.h.smith@ieee.org. Department of Computer Science, Central Michigan University, Mt. Pleasant, MI USA murthy@cps.cmich.edu. School of Engineering Science, Simon Fraser University, Burnaby, BC Canada ljilja@cs.sfu.ca.

query process, it will operate without any a priori knowledge being supplied by the user.

To begin, define a production rule (e.g., using ANSI Common LISP) to be an ordered pair – the first member of which is a set of antecedent predicates and the second member of which is an ordered list of consequent predicates. Predicates can be numbers (e.g., [1..2] $\vee$ [10..20]) or words [8].

Previously unknown words or phrases can be recursively defined in terms of known ones. For example, the moves of a Queen in chess (i.e., unknown) can be defined in terms of the move for a Bishop (i.e., known) union those for a Rook (i.e., known). This is a union of property lists. Other basic set operations may likewise be used (e.g., intersection, difference, not, etc.). The use of fuzzy set operators here (e.g., "almost the same as") pertains to computing with words [8].

Two sample rules and their representation follow.

Hydrogen $\wedge$ Oxygen $\wedge$ Spark $\Rightarrow$ Steam
R1: ({Hydrogen, Oxygen, Spark} (Steam))

Hydrogen $\wedge$ Oxygen $\wedge$ Match $\Rightarrow$ Steam
R2: ({Hydrogen, Oxygen, Match} (Steam))

R1 and *R2* may be generalized, since the consequent predicates are identical (i.e., the right-hand sides (RHSs) are equivalent) and the antecedent terms differ in exactly one predicate. This is termed a level-1 generalization because it is one level removed from ground truth. In a level-*i* generalization, *i* is the maximum level of generalization for any antecedent predicate. The need for a generalization squelch arises because contexts may be presented for which there is no matching rule in the real space. Generalizations can be recursively defined.

The advocated approach captures an arbitrary rule's context – something that cannot be accomplished through the use of property lists alone. If veristic terms such as "Warm" are generalized to such terms as "Heat" for example, then qualitative fuzziness will be captured.

A1: ({Heat} {Spark, Match}(X001 Explosive-Gas-Igniter))

Generalization, *A1*, tells us that antecedent predicate, "Heat" is more general than either a Spark or a Match. We may also write this as Heat > {Spark, Match}. Note that the relation, ">>" is used to denote ancestral generalizations (and vice versa). The general predicate is initially specified as *X00i*, but this is replaced after interactive query with the user, where possible. Otherwise, the next-level expansions will need to be printed for the user to read. Redundant, at-least-as-specific rules are always expunged.

The common property list follows the set of instances. Here, it informs us that a spark or a match may be generalized to Heat because they share the property of being an Explosive-Gas-Igniter. Properties are dynamic. They must be capable of being hierarchically represented, augmented, and randomized. In addition, property lists are subject to set operations (e.g., intersection). Properties can be acquired by way of database and/or user query.

User-queries can be preprocessed by a companion veristic mining system. Similarly, system-generated queries can be post-processed by companion systems. Companion systems can also play a role in imparting tractability to the inference engine.

Consequent terms, being sequences, are taken to be immutable. The idea here is to automatically create a hierarchy of consequent definitions to maximize the potential for rule reuse. Begin by selecting a pair of rules having identical left-hand sides (LHSs), where possible. Consider:

R3: ({Hydrogen, Oxygen, Heat} (Steam))
R4: ({Hydrogen, Oxygen, Heat} (Light, Heat)}

Next, an attempt is made to generalize the consequent sequences with the following result.

C1: ((Energy) ((Steam) (Light Heat))(X002 Power-Source))

Here, the properties of Steam intersect those of Light and Heat to yield the property, Power-Source. Thus, a property of Energy, in the current context at least, is that it is a Power Source. Rules *R3* and *R4* are now replaced by their valid generalization, *R5*:

R5: ({Hydrogen, Oxygen, Heat} (Energy)}

A key concept is that further learning can serve to correct any latent errors. In addition, notice that as the level of randomization increases on the LHS and RHS, the potential for matching rules and thus inducing further generalizations increases by way of feedback. Consequent randomization brings the consequents into a normal form, which then serves to increase the possibility of getting antecedent generalizations, since more RHSs can be equated. Antecedent randomization is similar.

Next, consider *R5*, where *R6* is acquired and appears as follows after substitution using *C1*.

R6: ({Candle, Match} (Energy)}

The system always attempts to randomize the knowledge insofar as is possible. Using *A1* and *C1* leads to the level-1 conjecture, *R7*, which replaces *R6*.

R7: ({Candle, Heat} (Energy))

R7 is not to be generalized with *R6*. This is because {Match, Heat} is the same as {Match, Spark, Match}, which of course reduces to Heat and is already captured by *R7*.

At this point, learning by the system can be demonstrated. Suppose the user asks the system what will happen if a spark is applied to a candle. While a plausible method to light a candle, this will not usually be successful. Thus, the user must report to the system the correct consequent for this action:

R8: ({Candle, Spark} (No-Light))

R8 is a more-specific rule than is *R7* because the former is a level-1 generalization, while the latter is at level-0. Thus, *R8* will be preferentially fired when possible using a most-specific agenda mechanism. It too will be subject to subsequent generalization. Notice that the new consequent will protect against similar error.

The learning process has not completed. We still need to correct the properties list so that Matches and Sparks can be differentiated in the context of lighting a candle. The following property (i.e., LISP) list is obtained.

P1: (Match Explosive-Gas-Igniter Wick-Lighter)
P2: (Spark Explosive-Gas-Igniter)

Now, since Heat is a super class of Match, its property list is unioned with the new property(s) – Wick-Lighter. Suppose, at this point, the user poses the same question, "What will happen if a spark is applied to a candle?" Rule *R7* informs us that it will light; whereas, *R8* informs us that it will not. Again, the inference engine can readily select the appropriate rule to fire on account of specialization. However, here there is yet more to learn. Here is what is known: *R7* and *R8* differ on the LHS in exactly one predicate and $prop(Energy) \cap prop(No-Light) = \varnothing$. The reason that the candle lights for a match, but not for a spark can be delimited by computing, $prop(Match) - prop(Spark) =$ *prop(P1)* – *prop(P2)* = (Wick-Lighter). Rule *R7* is now replaced by *R7'*:

R7': ({Candle, (X003 Wick-Lighter)} (Energy))

That is, a property list named, X003, has been substituted for Heat. Notice that X003 is necessarily a subclass of Heat. Then, anything that has (all) the properties on the property list (i.e., X003) can presumably light a candle (e.g., a torch). Observe that the human-in-the-loop need not know why is a list of properties relevant, as the reasons will be automatically discovered. Notice that a Spark can no longer light a candle and only those items having at least Wick-Lighter in their property classes can light a candle. Observe the non-linear learning that has been enabled here!

Consider now the rule:

R9: ({Candle, Match} (Energy))

Clearly, this rule is correct as written. Candles do indeed produce steam, light, and heat. The utility of induction follows from the fact that the system has no knowledge that a candle is a hydrocarbon and hydrocarbons produce steam as a byproduct of combustion.

Antecedent predicate generalizations can be rendered more class-specific as necessary to correct overgeneralizations by increasing the number of levels of available generalization. The rule consequents will not be affected. For example:

A2: ({Car} {Ford, Fiat})

yields:

A3: ({Car} {Family-Car, Sports-Car})
A4: ({Family-Car} {Ford})
A5: ({Sports-Car} {Fiat})

Property lists can be automatically organized into a hierarchical configuration through the use of simple set operations. This means that rules can be generalized or specialized through the use of the disjunctive or conjunctive operators, respectively. Such property lists can be associatively retrieved through the use of a grammatical randomization process [9]. Moreover, matching operations then need to incorporate searching subclasses and super classes as necessary.

Finally, we note that this system can incorporate fuzzy programming [10]. Fuzzy programming will enable the system to explore a space of alternative contexts as delimited by optional consequent filters and ranked by the level of generalization used to obtain a contextual match (see below).

3. GRAMMATICAL RANDOMIZATION

Consider the following three property lists.

P3: (Ice A B C)
P4: (Water B C D)
P5: (Steam C E)

Here, ice, water, and steam share a common property, C, which for example might be that they are all composed of H_2O. Also, only ice has property A (e.g., frozen), only water has property D (e.g., liquid), and only steam has property E (e.g., gaseous). Observe that only ice and water share property B (e.g., heavier-than-air).

The use of a hierarchical object-representation is fundamental to the specification of property lists, antecedent sets, or consequent sequences. For example, when one specifies the object say, "aircraft carrier" one implicitly includes all of its capabilities, subsystems, and the like. One cannot and should not have to specify each subsystem individually. We proceed to develop a randomization for the sample property lists; although, it should be clear that the same approach will work equally well for the antecedent and consequent predicates. Perhaps the most relevant distinction is that one needs to distinguish object sequence dependence from independence in the notation. Of course, property lists are sequence-independent.

As the example stands now, to specify the properties of ice or water, one need state the three properties of each (in any order). This may not seem to be too difficult to accomplish, but this is only because the list-size is small. Consider now the randomized version of the property lists:

P3: (Ice A Precipitation)
P4: (Water Precipitation D)
P5: (Steam C E)
P6: (Precipitation B C)

Here, the property of precipitation has been randomized from the property data. Observe, that if the user states property B, then the system will offer the user exactly three choices (e.g., by way of a dynamic pull-down menu): B, Precipitation, or Random. The Random choice allows the user to complete the specification using arbitrary objects. In other words, an associative memory has been defined. Similarly, if the user selects Precipitation, then the system will offer the user exactly four choices (e.g., again by way of a dynamic pull-down menu): Precipitation, Ice, Water, or Random.

Suppose that in keeping with the previously described nomenclature conventions we had the following property list specifications.

P3': (X004 A Precipitation)
P4': (X005 Precipitation D)

In this case, if the user selects Precipitation, then the system will offer the user the following five choices – Precipitation, A, D, or Random. In other words, it attempts to pattern-match and extrapolate the set.

In practice, randomization is based on known classifications – not arbitrary ones. Thus, in the previous example, the randomization of *P3* and *P4* require that *P6* be known a priori. Again, this still allows for the use of integer identifiers.

Next, it can be seen that the utility of randomization is a function of its degree. The relevant question then pertains to

how to realize the maximal degree of randomization. First, recall that as rules are generalized, the possibilities for further predicate generalization are increased. This, in turn, implies that the substitution and subsequent refinement of property lists for predicates is increased. Finally, as a result, the virtual space of properly mapped contexts (i.e., conjectures) grows at a rapid rate. Experimental evidence to date indicates that this rate may be exponential for symmetric domains.

4. ACTIVE RANDOMIZATION

Active randomization is a symbiosis of property lists and grammatical randomization. Property lists are really just predicates that are subject to grammatical randomization. Moreover, randomized predicates allow the user to specify contexts and associated actions using minimal effort [9]. Next, suppose that we had:

(A B C D) (i.e., the properties of A are B, C, D)

Here, A is the randomization of B, C, D. Similarly, we have:

(B E F)

and the two rules (i.e., antecedent differentiation):

R10: A S --> W
R11: X S --> W

Then, we can create a randomization:

(Q A X)

which, since valid leads to the following replacement of *R10* and *R11*.

R12: Q S --> W

This replacement allows for the possibility of new rule pairings and the desired process then iterates. Thus, we have:

(Q: A $\cup$ X) {expanding A, X}

These are *active transforms* [9] in the sense that whenever A or X change their membership, the properties of Q may change. Evidently, this is a converging process. However, if subsequently we had:

R13: A S --> T
R14: X S --> G

where, T and G have no properties in common (i.e., neither is a subsequence of the other), then it becomes clear that A cannot substitute for X and vice versa. In other words:

R13: (A-X) S --> T
R14: (X-A) S --> G

Thus, we have:

(A: A - X) {contracting A}
(X: X - A) {contracting X}

These are active transforms and again this is a converging process. Next, suppose that T and G are such that G is a subsequence of T without loss of generality. Then, it follows that A is a subset of X and

(A: A $\cap$ X) {contracting A}
(X: X $\cup$ A) {expanding X}

These are active transforms. This is not however necessarily a converging process. That is not to say that it will diverge without bounds. It is just not stable. We do not view this as a problem. It is to be viewed as an oscillatory system that in some ways may mimic brain waves. The complexity of interaction will increase as the system is scaled up. The eventual need for high speed parallel/distributed processing is apparent. The case for consequent differentiation is similar. Here, one is processing sequences instead of sets though.

5. AN INTELLIGENT TUTOR

KASER, a soft expert system, computes with words and employs qualitative fuzzy reasoning [11]. It captures knowledge in the form of rules whose predicates may be hierarchically categorized. KASER introduces declarative object trees into expert systems. The graph-theoretic relation among predicates enables KASER to be flexible as opposed to brittle. It can capture and represent verbally accessible domain knowledge possessed by a knowledge engineer. KASER's inference engine utilizes a gradient descent algorithm to minimize error. The object-oriented semantic predicate relations allow KASER to offer a metaphorical-based explanation subsystem. The induction of symbolic metaphor makes KASER ideal as a tutoring system. Indeed, we learn by analogy and this modality is served by KASER.

One of the hallmarks of an effective tutoring system is the capability for intelligent instructional delivery. Students learn by exploring alternatives - alternatives of their own creation. Indeed, to the extent that (non-trivial) education depends on memorization, it is not learning and to the extent that education is synonymous with learning, it does not depend on (non-trivial) memorization. KASER-based tutors make use of inherent domain symmetries to provide the student with feedback on their creative processes. This is possible because, in the KASER, the virtual rule space in much larger than the actual rule space. The system learns to adjust its own creative processes through exercise of the model and interaction with a domain expert. Indeed, even the best of the brightest professors learn by way of informed trial and error. Furthermore, if one is not willing to endure the possibility of error, then one cannot learn. Not only does this serve as an argument for the KASER's relaxed approach to AI, but it serves as a definitive argument for the role of heuristics as well. Truly, formal methods for education and training need to be augmented with a softer instructional approach if true learning is the goal.

The KASER can be used in the design of intelligent tutoring systems where the following conditions are met:

- Multimedia is applicable to the educational experience.
- Domain knowledge can be codified readily in the form of production rules.
- Predicate knowledge can be programmed readily in the form of hierarchical object trees (a taxonomical basis).
- The educational domain has a high degree of symmetry.
- The domain is open and requires a very large number of rules to approximate it effectively.

- Domain knowledge is often fuzzy as opposed to precise (the temperature is warm – not 80.1^0 F.).
- The consequence of an incorrect answer is not a disaster as might be the case for example in the domain of clinical medicine.
- Instruction makes more use of metaphor than of memorization.
- The educational domain is stable – necessitating fewer software updates.
- Ideally, a touch-screen platform is available to facilitate human-computer interaction.

One of the tenets of randomization theory is 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.

Fig. 1 depicts an example of declarative object trees – including multimedia – that is used to capture knowledge in a KASER. Several pre-machined laser rods are shown. 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, chess, or crystal-laser design. In symmetric domains, the system is far more likely to induce correct new knowledge.

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 (Fig. 1). Although this figure depicts the KASER tool using words and phrases, KASER also can run on object-oriented functions and procedures.

No fewer than six predicates are possible, each having at least two possible alternative selections (Fig. 1). This implies that over 64 expert rules are needed to handle correctly every design contingency that the student might attempt. If each predicate has at least three possible alternative selections, then over 729 expert rules are needed to correctly handle every contingency. More realistically, if each predicate has at least ten possible alternatives, then over one million expert rules are needed.

The general measure indicates that KASER needed to use two levels of inductive inference to match a known valid rule.

Similarly, the specific measure indicates that two levels of deduction were necessarily employed to arrive at this match. Contemporary expert systems would not have been able to analyze this creative design, as they need to be programmed explicitly with the necessary details. KASER created new and presumably correct knowledge. Note that there is an 80 percent possibility that this new knowledge is valid.

In Fig. 3, the context depicted in Fig. 2 is run to conclusion. Here, the system correctly predicts the lasing parameters through the transformation of its basis knowledge. That is, an ytterbium-doped aluminum garnet is substituted for the ruby rod and liquid helium is substituted for liquid nitrogen, which we know will limit the heat dissipation and thus the power output. In Fig. 2, we deem that a medium level of output joules is proper because ionized argon gas produces a higher frequency pump than does xenon gas – making for a more efficient laser (less heat production). Otherwise, the output energy would indeed be low.

No fewer than six predicates are possible, each having at least two possible alternative selections. This implies that over 64 expert rules are needed to handle correctly every design contingency that the student might attempt. If each predicate has at least three possible alternative selections, then over 729 expert rules are needed to correctly handle every contingency. More realistically, if each predicate has at least ten possible alternatives, then over one million expert rules are needed.

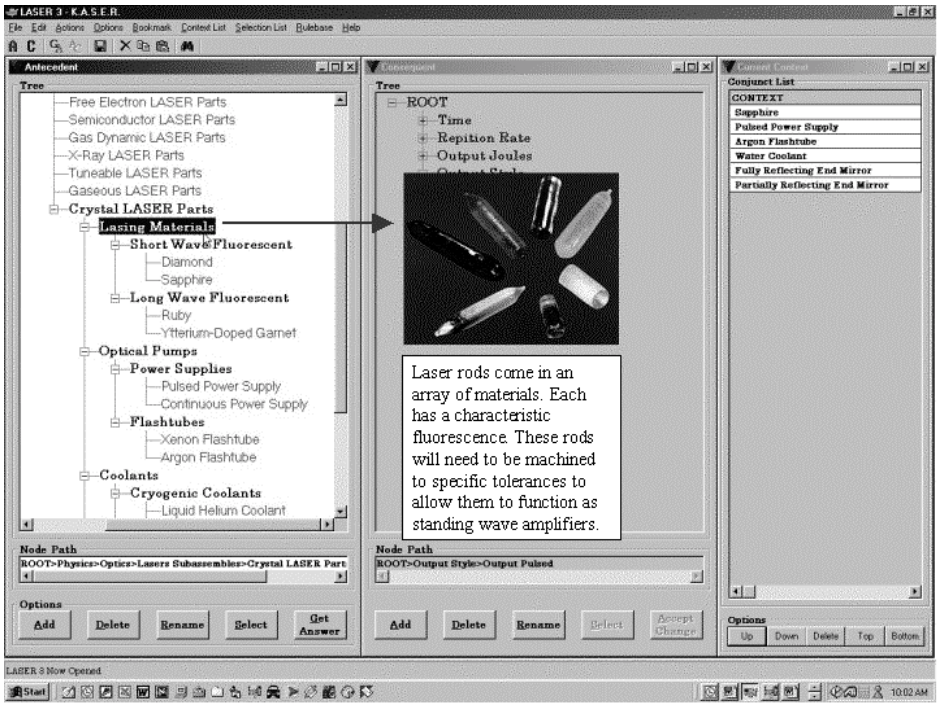

Fig. 1 Multimedia in the Antecedent and Consequent Declarative-Object Trees

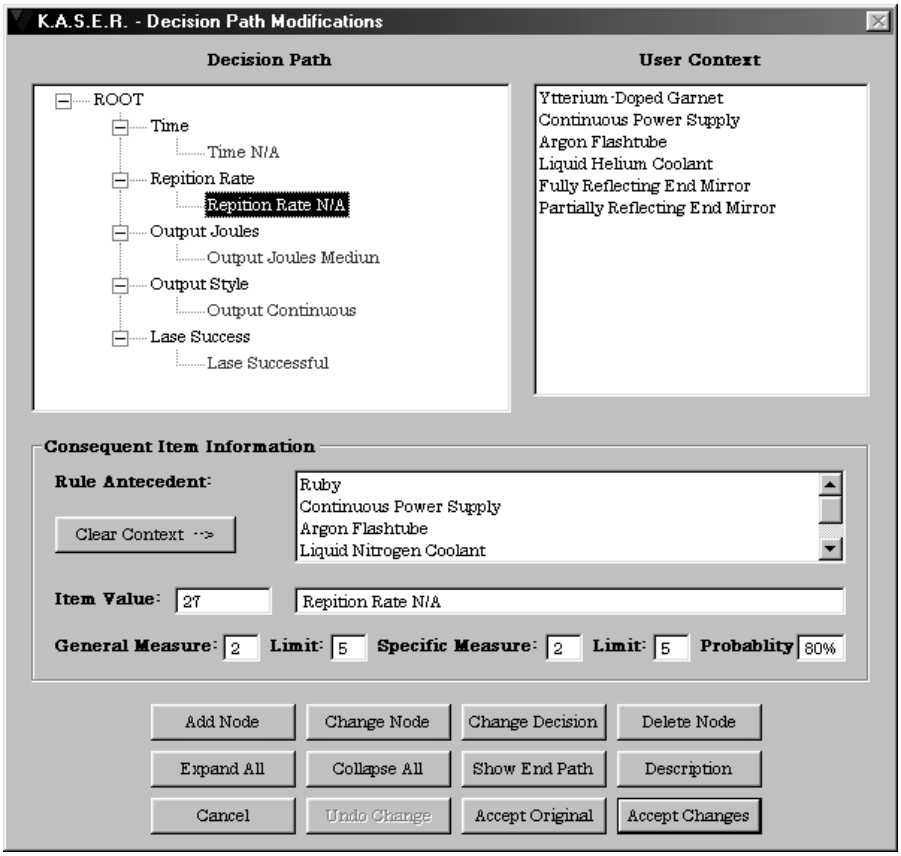

Fig. 2 Mapping the user context to the nearest-rule antecedent.

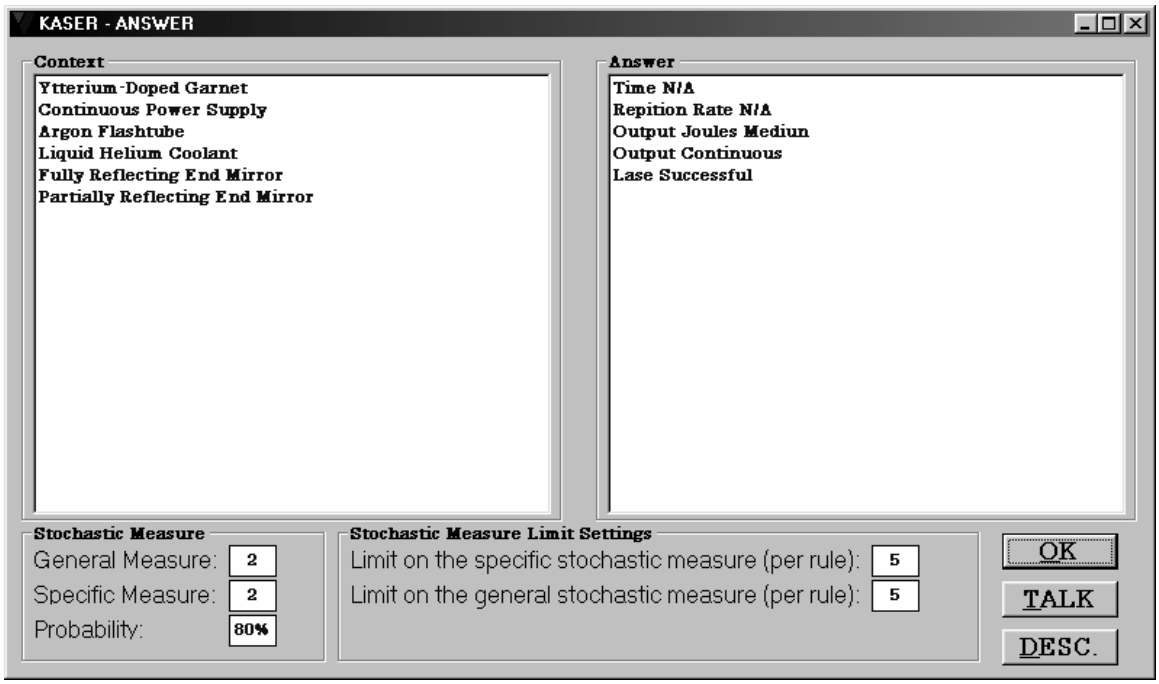

Fig. 3 Screen capture of a successful creative rule.

Clearly, it is impossible to capture a higher-level of intelligence using a conventional expert system. We observe in Fig. 3 that the generated answer is indeed correct. This is a fairly creative suggestion on the systems part – given a general stochastic measure of two. This is not to be attributed to chance; rather, it is a consequence of the high degree of domain symmetry exhibited by the crystal laser domain. Only twelve basis rules were initially supplied. More or less occasional errors can be repaired through rule acquisition (which will also lead to refined probability estimates) and/or tree adjustments.

The benefit provided by KASER operating in a symmetric domain is not that it is correct all of the time. Should that be, one needs to program large conventional expert systems. Rather, the KASER amplifies whatever knowledge is supplied to learn more from what it is told than is possible using conventional technology. Thus, KASER represents an important advance in artificial intelligence technology.

6. CONCLUSION

KASER is an example of a new and more creative type of soft expert system. This third-generation expert system computes with words and employs qualitative fuzzy reasoning [11]. It also may be said to fail softly and, for this reason, is not brittle. KASER cannot be realized through the exclusive use of predicate logic because of their inductive component. Furthermore, KASER is designed to capture and represent verbally accessible domain knowledge possessed by a knowledge engineer. KASER 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 what psychologists term domain transference and supra-linear learning (as previously discussed), 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 (e.g., a Ford is analogous to a Honda because they are both instances of car).

7. FUTURE WORK

The agenda mechanism 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 done in the area of fuzzy programming forward chaining [10]. Such work would relax the level of specificity required of the predicate specifications, thus making rules easier to specify.

KASER facilitates a high-level interaction between communicating sub-systems. That is, it is impractical to build intelligent networks using the brittle knowledge bases of contemporary expert systems. Moreover, success using the KASER forces one to revisit the role of representational formalisms (ontologies) in the capture and formal extension of knowledge. Clearly, representation is an important facet of information-theoretic randomization [1], [4], [10], [11]. It is expected that the development of the science of domain representation will lead to KASERs of ever-greater utility. It is evident that KASERs will open up a new avenue of scientific inquiry leading to an improved capability to represent knowledge. A generalized improved capability for formal representation will serve to fuse diverging methodologies in AI.

We are working on extending KASER capabilities through the use of associative memories and new representational formalisms. Creative systems of ever-greater capability loom on the horizon as a boldly attainable goal. If successful, the mismatch between the human and the machine will be reduced further. Finally, because the functionality of KASER improves with increasing domain symmetry, collecting statistics of user interactions with the KASER in known domains could pave the way for future research to characterize and measure the degree of symmetry in various domains. This will serve to instruct us how to become better professors as a consequence.

8. REFERENCES

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

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

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

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

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

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

[7] C.T. Clark, “An Interview with Marvin Minsky,” *Knowledge Management*, pp. 26-28, Jun. 2000.

[8] L.A. Zadeh, “From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions,” *IEEE Trans. Ckt. and Systems*, vol. 45, no. 1, pp. 105-119, 1999.

[9] S.H. Rubin, “The Role of Computational Intelligence in the New Millenium,” *Plenary Speech for the 3d World Multiconference on Systemics, Cybernetics, and Informatics (SCI'99) and 5th International Conference on Information Systems Analysis and Synthesis (ISAS'99)*, Orlando, FL, pp. 3-13, Jul. 1999.

[10] S. H. Rubin, “On Knowledge Amplification by Structured Expert Randomization (KASER),” *Space and Naval Warfare Systems Center, San Diego, Biennial Review, 2001*, SPAWARSYSCEN TD 3117, Aug. 2001.

[11] S.H. Rubin, R.J. Rush Jr., J. Murthy, M.H. Smith, and L. Trajkovic, “KASER: A Qualitatively Fuzzy Object-oriented Inference Engine”, *NAFIPS-FLINT 2002, IEEE Intern. Conf on Fuzzy Logic for Internet Applications and Fuzzy Information Processing, New Orleans, LA, 27-29 Jun., 2002* .