Geodesic Search and Retrieval of Semi-Structured Databases

Stuart H. Rubin and Shu-Ching Chen, *Senior Members, IEEE*

***Abstract*—This paper addresses the learning of search-control knowledge expressed in semi-structured natural language. That knowledge is mined from semi-structured databases and applied to search the same. As is the case with most self-referential systems, this process necessarily addresses the issues of randomization, representation, machine learning, and evolutionary programming. Machine learning becomes pervasive at all levels of representation. That is, the randomization of knowledge necessarily includes that of its representation. Application to the search and retrieval of multimedia databases is suggested.**

I. INTRODUCTION

REPRESENTATION is key to the learning process of humans (e.g., visual, auditory, kinesthetic – i.e., multimedia). It is likewise central to the learning process of computational machines [1]. This paper presents a new science for the computational evolution of semantics through the use of an unstructured natural language; the structured generalization of semantics, as well as the analogous construction of semantics through the use of a production system using a most-specific inference engine. In theory, the capability for the semantic translation of natural language allows for the bootstrapping of knowledge bases of ever-higher levels of generality, where the information-theoretic density of knowledge cannot be bounded above.

A fundamental component of ISR technologies pertains to all levels of data fusion and extrapolation [2]. The greater need here is for methods that can parse natural language queries, map them to their semantic normalization, and retrieve information associatively tagged with the normalization in a seamless heterogeneous architecture. Retrieved information can be passive in the sense that it is limited to the data level or active in the sense that it may be a method for computing the desired information. The scientific goal here is to make literal and latent information alike, which may be imbued in a semi-structured database, available for reuse and subsequent integration. While this is a computationally intensive process, it has the advantage of being maximally amenable to execution on fine-grained processors. Unlike the case for neural network applications, a consequence of semantic retrieval is that resulting knowledge can be explained to the user by way of metaphor. Moreover, the KASER, (U.S. Patent No. 7,047,226) can generate analogous features from a feature set [3]. Given that the proposed semantic retrieval methodology can automatically learn to extract relevant phrases (i.e., features) and their sequence from a supplied query, the system will converge on ever-better sets of features and heterogeneous rules expressed in terms of those features for purposes of fusion and prediction. Features are evolved along geodesic lines to minimize evolutionary time, but are necessarily annealed to insure diversity [4].

Manuscript received February 26, 2007. This work was supported in part by the U.S. Office of Naval Research (ONR) under an ILIR Grant and in part by NSF EIA-0220562, NSF HRD-0317692, and by a Florida Hurricane Alliance Research Program sponsored by the National Oceanic and Atmospheric Administration.

S. H. Rubin is with SPAWAR Systems Center, San Diego, CA USA (Phone: 619-553-3554; fax: 619-553-1130; e-mail: stuart.rubin@navy.mil).

Shu-Ching Chen is with the Florida International University, School of Computing and Information Sciences, Miami, FL 33199 USA (Phone: 305-348-3480; fax: 305-348-3549; e-mail: chens@cs.fiu.edu).

II. NEED AND BACKGROUND

The product of Dr. Rubin's 2004 ONR-funded research effort – the *semantic normalizer* can be easily trained by a bilingual and otherwise ordinary user to translate natural languages (e.g., to backend COTS Arabic to English translators). This project also resulted in a novel learning algorithm for *message summarization* for use by various naval reporting agencies. This effort led to an algorithm for effective metaphorical learning and a novel algorithm for mining data to be tested for efficacy against a COTS neural network program (results forthcoming). It was also ascertained that potential transitional customers for the semantic normalizer wanted a product that they did not have to train. As a result, we embarked upon developing a hardwired natural language interface for a relational database query language (SQL). Then, as a result of related-thinking on how best to apply our developing product to IED detection, a novel data mining capability was discovered, which is currently being built/tested and which serves as the starting point for this paper.

Application domains such as battle management, logistics, signal analysis, targeting and tracking, counter-insurgency, as well as the development of intelligent auto pilots for UAV (swarms), among others, can, it is claimed, be better addressed by the methodologies purported in this paper than by any combination of conventional methodologies (e.g., case-based reasoning, expert systems, genetic algorithms, neural networks, support-vector machines, *et al.*) [2].

III. APPROACH

The semantic search and retrieval of semi-structured database information (including multimedia objects and events) can perhaps best be introduced by way of example. That is, suppose that one entered the natural language query,

"Where is the nearest place for treating victims of gas explosions?" We want our information systems to be capable of replying with say, "I have three hospitals in New Orleans with burn units in order from nearest to furthest." Of course, we also want it to be able to further query the user to recursively disambiguate the last query, as necessary. For example, the system might query, "What are victims of gas explosions?" The user could reply, "They are burned, in shock, and/or not breathing." The system would learn this association and not repeat it twice in the same context. Observe that the mapping from "victims of gas explosions" to "burn units" in this case is strictly semantic, where the semantic association is acquired through the disambiguation of context. A methodology for machine learning, as applied to contextual disambiguation, serves as the focus for the scientific research overviewed in this paper.

Context-sensitive learning has been used by the KASER for the generation and necessarily limited checking of metaphorical knowledge [3]. The approach, to be defined below, is predicated on randomization theory [5], [6], [7], which allows it to be coordinated for execution on massively concurrent distributed processors, which is of course necessary for scalable realization. That is, the process of semantic association and normalization provide for information reuse, which allows for an exponential reduction in the search space. Now the process for performing semantic association and normalization requires domain-specific knowledge and the evolutionary acquisition of such knowledge will be discussed below.

Here, we will generalize that process and consider the *triangle inequality* for evolutionary functional instantiation. Let, f_i represent an arbitrary total computable program having m articulation points (i.e., formal parameters). Let us state, without loss of generality that each articulation point has on average n values (i.e., value parameters) that it can assume. The task then is to find a mapping for each articulation point such that the mapping

$$f_i((In_0, Out_0), (In_1, Out_1), ..., (In_{p-1}, Out_{p-1}), (In_p, Out_p)) \qquad (1)$$

is satisfied. That is, f_i must simultaneously map p I/O pairings. Fuzzy mapping is not enjoined and in any case the complexity of conducting a random search here is $O(n^m)$. We note that a consequence of randomization theory is that the minimal value for p cannot be proven in the general case [5], [6], [7]. Nevertheless, random testing theory provides us with a mechanics for comparing the efficacy of two alternative test sequences [4]. For example, when testing a simple sort routine, (((1 2 3) (1 2 3)) ((3 2 1) (1 2 3)) ((3 1 2) (1 2 3)) ((0) (0))) will do a better job, on average, of covering the execution paths of the sort routine than will the test of the same magnitude, (((1) (1)) ((2 1) (1 2)) ((3 2 1) (1 2 3)) ((4 3 2 1) (1 2 3 4))). The reason here is that the latter can be randomized into a theory; namely, ((*n, n-1, ...2, 1*) (*1, 2, ..., n-1, n*)), which the former test sequence cannot (i.e., it is more or less of a fixed point). Ideally, test sequences are random, which is defined by the point at which the size of the test sequence in bits approximates the size of the minimal program (i.e., theory) that captures it [6]. Next, consider breaking f_i into q schemas - f_j, f_k, …, f_u such that each total computable function has approximately $m' = \left\lceil m/q \right\rceil$ articulation points. Moreover, a reduction in the number of articulation points implies that $n' << n$ because there will be less formal parameters that one needs to provide a range for. The complexity of conducting a random search here and in general is $O(n^{\lceil m/q \rceil})$. Clearly, this is an exponential reduction in the search space as a function of q, as claimed. What we see here is that a little knowledge, represented as a schema as opposed to rules, frames, cases, etc. [1] can have an enormous impact on that which is tractably computable. This means that the process of semantic association and normalization has the potential to "exponentially" improve the quality and quantity of reuse in any pattern-matching search and retrieval system. Again, this is consistent with our scientific purpose here.

A change in the representation of knowledge can have a critical impact on that which is and is not solvable [1]. Humans tend to excel at finding better and better representations for knowledge. On the other hand, computers tend to far exceed human capabilities for number crunching. What is needed is a symbiosis of the two paradigms – ideally the human will do what (s)he does best and the computer will follow suite. Given this, we will incorporate a form of Evolutionary Programming (EP) to find sentential features, where we will determine the best learning algorithm and let the computer crunch it.

Compare and contrast this with genetic algorithms or neural networks that are *NP-hard* [8] in their learning and are thus theoretically reducible to bit-level representations. One way to avoid such intractability, in practice, is to encode high-level static feature representations into the domain. We can do much better though through the proper evolution and inclusion of domain-specific features. The *geodesic* principle implies that all subsystems are mutually dependent. Thus, the computationally costly evolution of the best feature sets benefits from the co-evolution of its guiding heuristics and vice versa as will be seen below.

A. Heuristic Evolutionary Feature Decomposition

The evolution of parsing (i.e., breakpoint) knowledge is based on the concept of contextual reduction. Again, consider our example query, "Where is the nearest place for treating victims of gas explosions?" It follows from the triangle inequality for evolutionary functional instantiation that the query needs to be reduced to its most basic set of features in order to minimize the computational complexity of normalization. For example, such a set is properly defined by, ((where is the) (nearest place for) (treating victims of) (gas explosions)). At first glance, it might appear that one can further simplify these parenthesized features by removing prepositions such as, "the", "A", etc. However, this deceptively simple approach might for example also transform (Vitamin A) into (Vitamin). Rubin *et al*. have proposed a field-effect approach to natural language semantic

mapping that is based on the *iterative randomization of sentential semantics* [9]. In this paper, a domain-specific algebra is introduced for acquiring sentential knowledge that is capable of transforming such features as, (the man bit the dog) into semantically equivalent, but syntactically simplified ones, such as (man bit dog).

Using a set-based methodology, our example query would be further randomized to become ((where) (nearest place) (treating victims) (gas explosions)). (This methodology can be rewritten and adapted to the context of the current problem domain.) Features can be thus normalized in linear time on a concurrent architecture, or in quadratic time as a function of sentential length on a serial processor. Features are then associatively hashed using a symbol table. Of course, the rate of growth in the symbol table decreases exponentially with scale. Again, this is in keeping with a scalable design. The initial query has been reduced to a set of integer tokens and that set is then mapped to the user-defined proper response – be that a machine-generated query to elicit further details for purposes of disambiguation, an effective procedure for generating a reply, and/or as defined above, the literal, "I have three hospitals in New Orleans with burn units in order from nearest to furthest." Notice that while the semantic mapping here is one to one, syntactic mapping will be many to one. Note too that effective procedures for generating replies can execute queries formulated in local languages for the retrieval of semi-structured database information.

Having detailed the refinement and semantic mapping of features above, the question remains as to how one can best determine initial feature breakpoints. That issue will be addressed next and is resolvable through the use of randomness and symmetry [6]. More formally, given a fixed sequence of words, $w_0, w_1, w_2, ..., w_n$, we want to identify that set of up to n+1 features, which are defined by, $w_i, ..., w_j$, $i \le j$, such that the feature set is properly mapped to its semantic association. Then, breakpoints are defined by a set of up to n+1 ordered pairs that consist of, (w_i and whether it's the *start*, *inclusion,* and/or *end* of a feature). Thus, there are 4^{n+1} possible breakpoints (i.e., start; inclusion; end; start and end). That would be several million breakpoints for a typical query. It follows that a form of Evolutionary Programming (EP) is properly applied to the approximation of the best loci for the breakpoints.

At the outset, breakpoints are evolved through the use of pure chance. Subsequently, known features are iteratively applied to reduce the complexity of search. This is the symmetric step and while it never completely precludes the random step (i.e., for purposes of annealing), it should be noted that the computational complexity of solving for the (near) best breakpoints, on average, greatly decreases with scale until it becomes linear. There is a well-known need to order the training instances that stems from Winston's work on the conceptual definition of an arch in Blocks World [10] and Mitchell's work on Version Spaces [11]. The results of this early work allow us to "kick-start" the random-symmetric learning mechanism much in the same manner as a child acquires a facility with a natural language by bootstrapping from simple to complex constructs.

For example, if say we have previously acquired two normalized features, (where) and (gas explosions), then the exponent in the number of possible breakpoints has been reduced by precisely this factor. If nothing else, this alone would allow EP to do a better job of optimization with the incursion of far less complexity. However, features may be embedded in other features; although, this allowance is precluded by our algorithm. An example of embedding is provided by the feature, (natural (gas explosions)). Symmetric reduction here would preclude the discovery of the best feature, (natural gas explosions). In view of this constraint and in keeping with Rubin's previous work on randomization [7], it is necessary that random search and symmetric reduction proceed concomitantly on separate threads. Theoretically speaking, that is to say that the effective discovery of the (near) best breakpoints for feature discovery is *recursively enumerable*, but not *recursive* [12]. The balance of computer resources to be dedicated to random vs. symmetric discovery at any given time is, as it turns out, not that difficult a problem. One simply sets up a feedback loop that measures the degree of success in feature randomization as a function of the degree of randomness and symmetry and applies evolutionary search to optimize the mixture for successful randomization over the time march. One theoretical result here is that in any non-trivial (i.e., scaled) system, random and symmetric discovery must both be present [4].

The remaining question, in the present context, pertains of course to how one puts such feedback on a metric basis. Were the discovery process strictly symmetric, then feedback would consist of matching rule antecedents stored in a knowledge base against the context consisting of the symmetrically-reduced features. Then, a count of rules that would be properly fired vs. improperly fired would yield the desired metric for comparison purposes. However, as we have just witnessed, the discovery process is inherently random too. This means that many unknown features will be created – features that can have no match among the existing base of rule antecedents at the time of creation (i.e., via diagonalization arguments). The solution is to delay feedback until such time as the random features sufficiently populate the base of rule antecedents to themselves become more or less transformed into symmetric features. An ever higher-quality rule base is evolved by expunging those rules and their associated features (i.e., over the entire rule base) having minimal metrics. While this no doubt sounds complicated, it does reduce to a simple solution as follows.

Rules are moved to the logical head of the base when fired and correct (see below). A rule is said to be *properly matched* if the case antecedent, which is associated with the rule is deemed by the user to cover the semantics of the rule antecedent. Whenever domain feedback leads to the

contradiction of a properly matched given fired rule, that rule and its associated case are expunged without any deletions being made to the symmetric lookup table; and, a new case will be acquired. Otherwise, the improperly matched rule is not expunged; but, any features that co-occur in this rule and are present in the context are expunged from the symmetric lookup table. (The fact that good features may be eliminated because they occur in the context of bad ones also has a positive side, where "single words" will be disassociated from the feature space to enable the formation of longer features. Cases associated with erroneous rules are not saved because new cases may have new features and because when space is at a premium, these cases have been shown to induce rules that are least reliable for the current context.) All other rules in the base, which contain these expunged features are "re-expanded" about these features and subsequently re-reduced using the updated symmetric lookup table. Remaining word sequences, if any, may become maximal-length features after normalization. Using our previous example, if ((where) (nearest place) (treating victims) (gas explosions)) were improperly triggered by the context about (treating victims) and (gas explosions), then assuming that (victims gas) were known by the symmetric lookup table, the updated result would be ((where) (nearest place) (treating) (victims gas) (explosions)). Note that (treating) and (explosions) have been subsequently normalized with possible removal as a consequence. In this manner, reductions, over time, are limited to those that prove themselves to be truly symmetric. Relatively infrequently-fired rules will eventually "fall off the bottom" and be replaced by ones containing genuine symmetric features. This statistical mechanical approach insures that on average, random features will not survive to become persistent symmetric ones. There need be a statistical correlation between features, or sets of features, and proper actions (i.e., rule consequents).

The context must be reduced in order of non-increasing size of the reduction entries in the symmetric lookup table (i.e., reducing the most-specific features first). Using an associative or hashing mechanism to parse the word sequence of length n, we find that there are at most $\sum_{r=1}^{n}\binom{n}{r} = 2^n - 1 = O(2^n)$ hashes to be made, which occurs where the symmetric lookup table is either empty or inapplicable. This shows that the associative parsing mechanism is quite tractable for the typical query length of say ten words or so (i.e., even with subsequent normalization). Actually, since features may not embed other features, the complexity of associatively parsing symmetrically reduced queries, which is the general case, is actually much less, or $O(2^{n-k/c})$, where c is the number of such reductions and k is their average length.

B. Case-Based Feature Discovery and Fusion System

Having completed our discussion of heuristic evolutionary feature decomposition above, there remains the need to reduce an arbitrary feature space to a more (most) salient feature space. Simply put, the less the inclusion of extraneous features, the better the pattern matcher can do in finding proper antecedents for matching the supplied context. In this section, we present an outline as to how to best accomplish this using a dynamic weight vector approach. In Section D below, an improved algorithm is presented, which replaces the dynamic weight vector approach with one for randomizing the feature sets themselves.

Here, a supplied context is compared to get a metric match for each case in the base. Expunged features have their associated case-column set to "—" in preparation for reassignment.

Example 1:

feature: f_0 f_1 f_2 f_3	(the four features having the greatest weights)
context: 1 1 1 1	
case: 1 0 1 0 → A	
match: 1 -1 1 -1	1 = match; -1 = not match;
W_0: .25 .25 .25 .25	score = .25(1) + .25(-1) + .25(1) + .25(-1) = 0;
W_1: .20 .20 .40 .20	score = .2(1) + .2(-1) + .4(1) + .2(-1) = 0.2 (better);

Example 2:

feature: f_0 f_1 f_2 f_3 f_4	
context: 1 1 1 1 1	
case: 1 0 1 0 -- → A	(Cases may be lacking one or more features)
match: 1 -1 1 -1 0	1 = match; -1 = not match; 0 = omitted;
W_0: .20 .20 .20 .20 .20	score = .2(1) + .2(-1) + .2(1) +.2(-1) + .2(0) = 0;
W_1: .17 .17 .33 .17 .17	score = .17(1) + .17(-1) + .33(1) + .17(-1) + .17(0) = 0.167 (better);

When evaluating a W_i, each row in the range, where there must be at least two rows in the range having the same consequent, will in turn have its antecedent, $a_{i,j}$, serve as a context, c_j. This context will be compared against every row excepting that from which it was derived. The score of the ith row is given by $\sum_{j=1}^{n} w_j(c_j - a_{i,j})$, where $\forall_j \mid w_j \geq 0$ and $\sum w_j = 1$. In the case of Boolean functions, define $(c_j - a_{i,j})$ $= \left\{ \begin{matrix} +1, & a_{i,j} = c_j; \\ -1, & a_{i,j} = \overline{c_j}; \\ 0, & otherwise. \end{matrix} \right\}$. Here, if the row having the *maximum* score has the correct consequent, award +1; otherwise, -1. In the case of real-valued functions, define $(c_j - a_{i,j})$ $= |c_j - a_{i,j}|$, which is always defined. Here, if the row having the *minimum* score has the correct consequent, award +1; otherwise, -1. Thus, the higher the score, the better the W_i,

where a perfect score is defined to be the number of rows in the range – the number of singleton classes there.

Example 2 is reworked below for the situation where there is continuous variation (Δ). By design, there will never be any omitted features here.

Example 2':
feature: f_0 f_1 f_2 f_3 f_4
context: 1 1 2 4 1 (can be phrases, rays, track prediction, etc.)
case: 3 1 3 3 1 → A
Δ : 2 0 1 1 0 perfect match = 0;
W_0: .33 .17 .17 .17 .17 score = .33(2) + .17(0) + .17(1) + .17(1) + .17(0) = 1.0 (can be larger);
W_1: .20 .20 .20 .20 .20 score = .2(2) + .2(0) + .2(1) + .2(1) + .2(0) = 0.8 (better);

The advantage provided by such a weighted vector approach is that it allows one to hill-climb an optimal solution using sigmoid functions. However, as will be seen in Section D below, hill-climbing can take the form of iterative symbolic improvement as well. The relative advantage here is best made clear by way of example. The inherent problem with the sigmoid approach is that one can map say cases to chess boards, but unless an exact match is to be had, there is almost nothing to be gained by finding that the current board is *almost* the same as the saved board. This is because, in chess and many other domains of practical import, variation in a single degree of freedom (e.g., the color of the square a piece sits on) makes all the difference. This argument naturally extends to procedural consequents as well. This will not be a problem using the approach presented in Section D because that approach seeks to symbolically and iteratively remove extraneous features. It then climbs from working rule sets to more general rule sets that remain relatively valid. That is principally why this second approach is to be preferred over this "neural" one. Indeed, Rubin [13] and Zadeh [14] provide compelling evidence that when the numerical basis for a fuzzy logic is relaxed, one can attain a capability to compute with words using self-referential symbolic transformations [13], or *protoform* schemas [14].

C. On Randomization and Discovery

In his theoretical paper, Rubin [4] proves the Semantic Randomization Theorem (SRT) (i.e., $\exists k \mid \forall c \geq k > 0,\ \varphi_r = \varphi_i^{t*c}(l) = \varphi_i^{t*k}(l),\ \mid \varphi_r \mid < \mid (\varphi_u \parallel \varphi_v) \mid$). First however, he proves that in general, randomization is inherently a heuristic process. Indeed, the salient arguments presented by Rubin [13] and Zadeh [14] are captured by the *Unsolvability of the Randomization Problem* [4]. This theorem serves to vindicate the inherent need for heuristics in symbolic randomization.

Theorem (unsolvability of the randomization problem): There is no algorithm, which when presented with indices *i* and *j* of arbitrary computable functions $\varphi_i : N \to N$ and $\varphi_j : N \to N$ can decide whether φ_i is a randomization of φ_j. Thus, there is no algorithm, which when presented with the index *j* of an arbitrary computable function $\varphi_j : N \to N$, $j \in N$, can randomize that function (i.e., transform it into φ_i, where *i* indexes an arbitrary randomized, or random, function).

It follows as a consequence of the SRT [4] that the complexity (density) of knowledge is unbounded in the limit. While intelligence is in every case constrained by the operant laws of space and time, every non-trivial (i.e., self-referential) finite realization of intelligence involving randomization is necessarily domain specific and not recursively enumerable (i.e., inherently heuristic) as a consequence of this theory [4]. The methodology used to prove the SRT allows us to define a heuristic mechanics in the novel algorithm for randomizing feature sets for use in the geodesic search and retrieval of semi-structured databases.

D. Geodesic Randomization-Based Feature Discovery and Fusion System

An improved innovative algorithm for iteratively generalizing the feature space is depicted in Fig. 1. Our previous example, as currently written, has:

((where) (nearest place) (treating victims) (gas explosions)) → "I have three hospitals in New Orleans with burn units in order from nearest to furthest." (2)

One candidate randomized rule would be:

((where) (treating victims) (gas explosions)) → "I have three hospitals in New Orleans with burn units in order from nearest to furthest." (3)

A *geodesic randomization* is defined over the entire rule base, since each randomization can potentially affect that which can be randomized (i.e., because of the introduction of contradictions). In other words, the randomization space is *co-dependent*. An overview of geodesic randomization follows in Fig. 1.

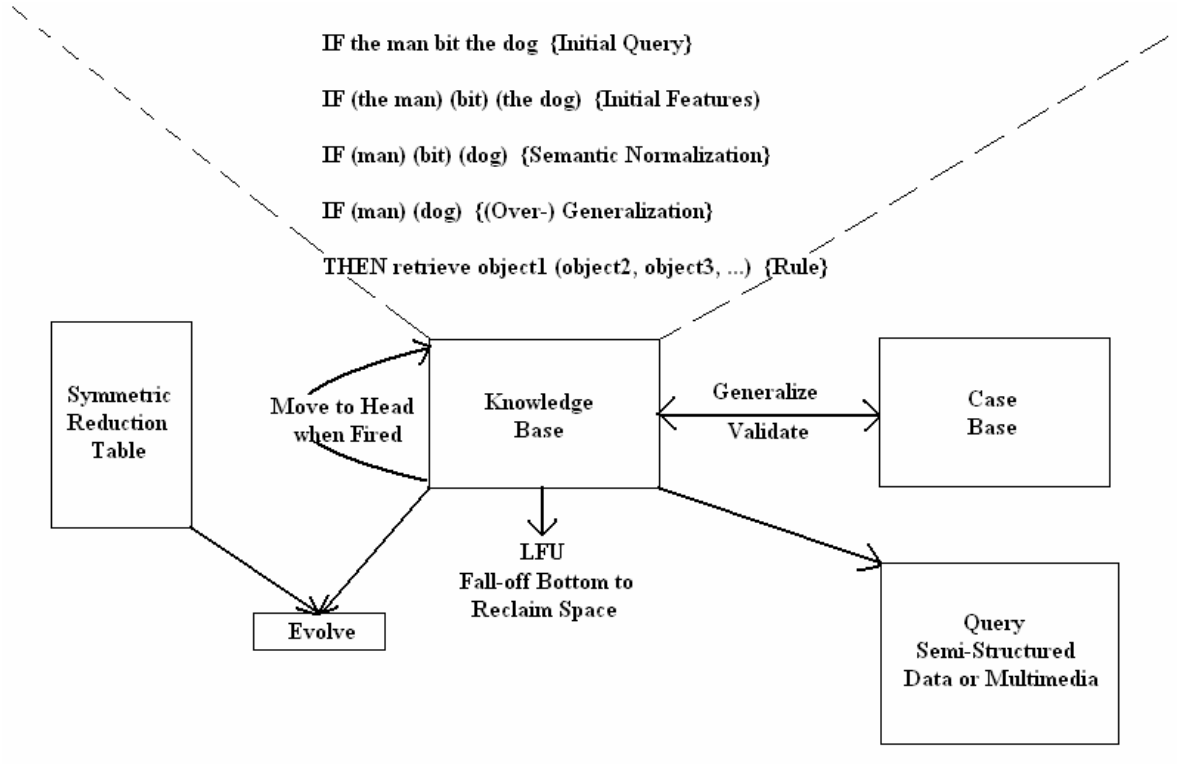

Fig. 1. An overview of geodesic search and retrieval of semi-structured data or multimedia

IV. Conclusion

Evolutionary programming, randomization, machine learning, and their associated representations are endemic to the search and retrieval of large databases. Furthermore, the acquisition of knowledge facilitates the further acquisition of search-control knowledge, which after all defines randomization [6]. Application to the search and retrieval of multimedia databases, in particular, will be examined in our next paper.

Acknowledgment

This work was produced in part by a U.S. government employee as part of his official duties and is not subject to copyright. It is approved for public release with an unlimited distribution.

References

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

[2] Y. Dote and S.J. Ovaska, "Industrial applications of soft computing: a review," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1243-1265, 2001.

[3] 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.

[4] S.H. Rubin, "On Randomization and Discovery," *Information Science*, vol. 177, iss. 1, pp. 170-191, 2007.

[5] G.J. Chaitin, "Information-Theoretic Limitations of Formal Systems," *J. ACM*, vol. 21, pp. 403-424, 1974.

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

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

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

[9] S.H. Rubin, S.C. Chen, and M.L. Shyu, "Field-Effect Natural Language Semantic Mapping," *Proc. 2003 IEEE Int. Conf. Syst. Man, Cybern*., Washington, DC, pp. 2483-2487, 2003.

[10] P.H. Winston, *Artificial Intelligence*, first ed., Addison-Wesley Pub. Co., 1977.

[11] T.M. Mitchell, *Version Spaces: An Approach to Concept Learning*, Ph.D. thesis, Stanford University, 1978.

[12] A.J. Kfoury, R.N. Moll and M.A. Arbib, *A Programming Approach to Computability*, New York, NY: Springer-Verlag Inc., 1982.

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

[14] 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.