SYMBOLIC DIAGNOSIS for INTELLIGENT CONTROL

Steven Jowers and John H. Painter
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843-3128

ABSTRACT

This paper reports the results of research to create a symbolic diagnostician to support intelligent control of numerical processors and/or processes. Example applications include 'real-time' signal processors, industrial automation, and aerospace power systems. The approach is to create a generic, symbolic inference engine to interpret data from 'real-time' numerical processes. The interpreted data is then utilized by companion symbolic and numeric modules resulting in a dynamic, intelligent 'real-time' control archectictere. General results are obtained while focusing research efforts on an initial target application — a software-intensive radio receiver/processor. Object-oriented programming techniques are used due to ease of knowledge engineering and potential parallels to hardware implementation.

INTRODUCTION

Intelligent Control is loosely defined here as the use of symbolic processing for the purpose of controlling numerical (procedural) systems. An 'Intelligent Controller' embodies functions of inference as well as control, based on symbolic processing. When such symbolic inference deals with more than just the 'state' of the process being controlled, then it is prudent to separate inference from control, both functionally, and architecturally. This paper examines symbolic inference in support of intelligent control and also as a source of valuable output information from an intelligently controlled process.

The concept of Intelligent Controls was first proposed by K.S. Fu to link Artificial Intelligence (AI) and Automatic Control [1]. Today, Intelligent Control is applied to industrial automation, where machines (robots) function without complete *a priori* knowledge of the work environment. Such implementations are rudimentary in terms of some previous promises of AI [2]. This paper's approach is pragmatic, however, following the suggestion of [3]:

> "... treating AI as an engineering discipline having a new set of software tools and techniques that can create more powerful and natural systems, regardless of any similarity to human solution techniques." [3]

In the present work, a specific application has been chosen as a target around which to develop general results. The particular target is a 'software-intensive radio,' which is envisioned as being digitally implemented. Symbolic inference and control for managing the radio is demonstrated via a computer emulation (Monte Carlo). The symbolic control architecture so developed is held to be generic and not dependent on the target application, per se.

This research was supported, in part, by NASA, under Grant No. NAGW-1194

SYMBOLIC INFERENCE AND CONTROL FOR NUMERIC PROCESSES

ARCHITECTURE

The architecture of the integrated symbolic/numeric processing system is shown in Figure 1., below, and is similar to one postulated by Saridis [4]. Saridis' view was that such an architecture may be viewed as a hierarchy according to a tenet he described as the "Principle of Increasing Precision with Decreasing Intelligence."

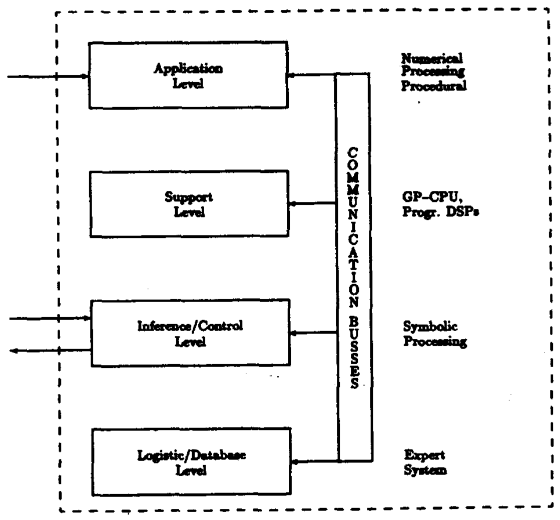

Figure 1. Total System Architecture.

The architecture dealt with in this paper is four-level and is inverted from Saridis' view, in that the upper levels are high precision (numerical processing), corresponding to the target application. The data (sensory) input to the application is at the top level. The top two levels comprise numerical processing, with the topmost level being the prime 'Application' level and the next level down being used for numerical 'Support.' Note that modules at this level support the top level or the next level down. The third level contains symbolic processing modules whose functions are to draw inferences and to provide control (generalized) for the top levels. The final, fourth level comprises the 'Data-base', or library, which supports the 'Inference/Control' level.

It is with the third level, the 'Inference/Control' level that this paper is concerned. It is envisioned that this level generally has two different types of modules serving different purposes. The purpose of one type is restricted to real-time ('decision-directed') control and management of the numerical processor and is herein called the "Symbolic Controller." It implements control tactics which may not be compatible with an inflexible, 'hard-wired,' direct encoding of algorithms in the numerical processing program. Such control tactics are, for example, those depending on heuristic 'rules,' such as might be used by a human operator. The second type of symbolic module at this level is that which interprets, not the state of the numerical processor, but the data flowing through the processor. It is this second type of symbolic inference module which is dealt with in the present paper.

Functionally, the inference processes conceived here are to be used for diagnostic purposes. This is different from the inference function internal to the Symbolic Controller which manages the numerical processor. In the controller, inference is restricted to just determining the nominal operational 'state' of the processor. (See the companion paper by Painter, Lin, and Glass [5].) Here, inference is used to 'diagnose' data which is flowing in the numerical processor. The results of this diagnosis yield inferences about conditions external to the numerical processor and also about the level of performance (tuning) of the numerical processor. Therefore, we will explicitly name this kind of inference engine, "Symbolic Diagnostician."

As in the case of the intelligent controller, it turns out that the symbolic diagnostic processor is supported by numerical processing, expressly dedicated to diagnosis. That is, at the support level in , there are one or more numerical modules which support the symbolic diagnostic processing on the third level of the architecture. Likewise, at the fourth level, there are knowledge bases which are associated specifically with the diagnostician. This partitioning on the three lower levels is shown explicitly in Figure 2., below.

GENERALITY OF APPROACH

The purpose of the diagnostician is two-fold. First, it provides information to the Symbolic Controller about the level of performance of the numerical processor. That is, the diagnostician does not determine what the numerical processor is doing, but, rather, how well it is doing it. With this information, the controller may then 'tune' the numerical processor's algorithmic parameters. Such diagnostic information may be provided routinely. Alternately, the controller may request the diagnostician to run tests.

The second diagnostic purpose is to infer conditions external to the numerical processor. That is, the diagnostician may serve to determine general characteristics of the external environment. These might be characteristics for which the numerical processor was not initially optimized. For instance, in the case of a dynamically maneuvering vehicle under intelligent control, the diagnostician might be tasked to characterize the environment of the sensors providing guidance information for the vehicle. Such information is at a higher level of abstraction than that for which the guidance system was initially optimized. This information might then be passed to the intelligent controller as meta-knowledge [6] for use in reoptimizing the guidance algorithms.

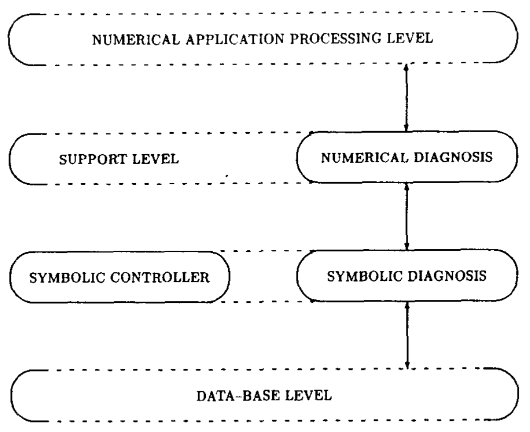

Figure 2. Control/Diagnostician Partitioning.

Figure 2. shows a general architecture, applicable to any numerical system application. The detailed implementation of the diagnostician, at both the symbolic and numerical processing levels, depends on the specific targeted numerical application. The specific implementation depends on the kinds of raw data available to the diagnostician, within the application system.

The types of applications envisioned here are those where the numerical processor is engaged in control or signal processing for dynamic systems. Examples are signal processors and/or controllers for vehicles, robots, aerospace power systems, and radio or image processors, to name a few. In these types of applications, the numerical processor deals with sampled-data waveforms. Therefore, the data available to the diagnostician is waveform data, or is derived from waveform data, after intermediate processing. For such an application, the diagnostician can be expected to examine internal processor data in two domains: 1) Frequency-domain; and 2) Time-domain.

In the time domain, the numerical modules of the diagnostician may compute moments of waveforms occurring at various locations in the application processor. Such locations of interest would include the feedback error points of various tracking loops. Other locations might include 'channels,' wherein various signals and noise are flowing. Moment calculations then yield measures of 'signal power' or 'noise power,' as well as other averages. Another time domain measurement might include the 'envelope' of a waveform, or short-term time-averaged r.m.s. value. In the frequency domain, spectra of signals are of interest. These are derived from various numerical transformations, such as the Fast Fourier Transform (FFT) or Wigner Transform [7].

With the numerical diagnostician modules producing time-domain and frequency-domain measures of signal characteristics within the application processor, the diagnostician's symbolic modules then accept these numerically-derived data and process them symbolically. Functioning as an "Expert System," the Symbolic Diagnostician compares the numerical data to stored data and draws inferences therefrom. For data which lends itself to graphical interpretation, such as frequency spectra and time-domain envelopes, symbolic 'image interpretation' is performed [8,9].

For applications such as robots, vehicles, and other dynamically controlled systems, the diagnostician is concerned mainly with supporting the Symbolic Controller. However, for applications such as radio or image processing, the diagnostician is a valuable adjunct to the numerical processor, for the purpose of interpreting the external data, itself. It is such an application that provides a focus for the present work.

SOFTWARE-INTENSIVE RADIO APPLICATION

In order to provide a 'real-world' challenge for the development of the diagnostician, and to keep the effort well focused, a specific application has been chosen for the target numerical processor. It is a radio processor, which is highly digitalized in implementation. In the present effort, the radio is emulated in software. Such a radio application yields a rich environment within which to develop the intelligent control structure and elements of the Symbolic Diagnostician. The radio provides a multi-loop control regime, as well as a requirement for interpretation of the external signal environment. The system has strenuous real-time processing requirements.

APPLICATION ARCHITECTURE

The application architecture for the radio system is shown below in Figure 3., including the emulated signal generator.

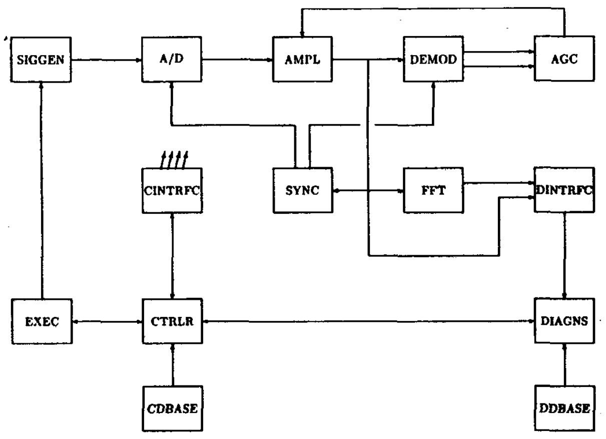

Figure 3. Radio Application Architecture.

This is a minimum configuration, suitable for development of the diagnostician. The total application system architecture has more structure than shown here. (Parenthetically, one of the nice things about basing development on an emulated application is that the application architecture may be pared down to just that necessary to support development.)

In the figure, 'signal' flows roughly from the upper left, across and down to the right. The upper (application-level) modules include an A/D converter (emulated), a signal amplifier (or scaler), a demodulator, and an automatic gain control module. On the second (numerical support) level are a synchronizing module for synchronizing both the A/D and the demodulator. Other modules on this level are an FFT (batch) transformer and a pair of numerical interface modules, one supporting the Symbolic Controller and one supporting the diagnostician. On the third (symbolic processing) level are the controller, diagnostician, and an executive for running the simulation. The fourth level contains the data (rule and parameter) bases for controller and diagnostician.

The signal generator produces pseudo-random, white, gaussian noise samples, plus a selection of radio signals (emulated), which, latter, may be modulated or unmodulated, bearing digital data or other messages. The present development effort for the diagnostician focuses on two signal modulation types, binary phase shift keying (BPSK) and amplitude modulation (AM). The diagnostician is presently being developed to examine the time-domain 'channel,' at the application level of the radio (actually an intermediate frequency channel) and also the output of the FFT.

FUNCTIONALITY

The diagnostician is required to make several determinations. First is the determination whether there is signal present in the channel, or just noise. If there is signal, the determination is made as to the presence of modulation. If modulated, next is the determination as to whether the modulation contains digital data. Next, a discrimination between known modulation types is attempted. Also, a determination is attempted as to whether multiple signals (possible interference) occupy the channel. At the present level of development, the diagnostician is not tasked to determine quality of performance of the radio modules, themselves.

As will be described below in detail, the diagnostician utilizes symbolic interpretation of the FFT spectra to make some of the determinations listed above. Other determinations also require examination of the signal 'envelope.' Thus, the total characterization of the radio channel requires, in general, examination of time-domain and frequency-domain, both numerically and symbolically.

SYMBOLIC DIAGNOSTICIAN

In this section is detailed the initial design of the symbolic diagnostician, as tasked for interpreting the radio channel. Algorithms used will be briefly mentioned. Accordingly, discussion starts with the higher level inference process and structures and moves to lower, specific algorithms. In describing this design, comments are made concerning its portability to other problem domains.

The Symbolic Diagnostician is built upon two fundamental concepts: 1) feature extraction, and 2) inference as a function of feature. Feature extraction is accomplished via appropriate numeric and symbolic algorithms while the inference process uses this data to provide an intelligent description of the waveform. This information is subsequently utilized by the Symbolic Controller and other modules to implement the intelligent control paradigm [5].

EXPERT SYSTEM DESIGN

In [10], Hueschen and McManus report the design of an AI-based aircraft guidance and control system. Their design partitions the problem space such that multiple experts, each efficient for a subspace are used, with inferences reported to a general coordinating body. J.B. Cruz and Stubberud [11] discuss a similar idea. They used a Coordinator to manage the activities of multiple controllers. The present approach uses design concept similar to [10], but incorporates the more explicit design of [11]. Here, the Coordinator/controller paradigm is modified such that a Diagnostic Controller, DIAGNS CTRLR (Figure 4.), interacts with a collection of embedded Local Area Experts (LAE). The Controller dialogues with each LAE to gather opinions concerning the received signal data. Each LAE indicates a confidence measure of its opinion and the completeness of its analysis at the time of confidence calculations. The Controller then makes a decision using this knowledge and any history deemed relevant to the problem. A Symbolic Controller, described in a companion paper [5], is advised of this decision.

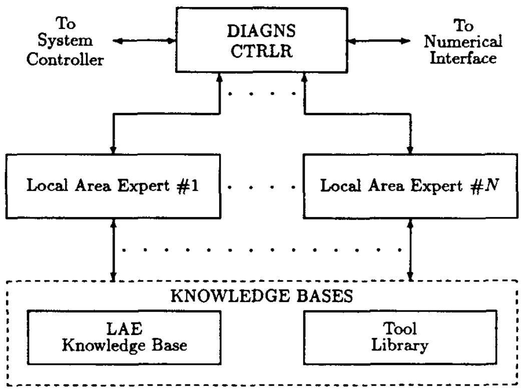

Figure 4. Spectral Diagnostician

LOCAL AREA EXPERTS

As mentioned above, the purpose is not to develop a single, real-time, all encompassing pattern recognition expert, but rather, to develop local experts, all of which simultaneously process data and submit opinions to the Diagnostic Controller, which then makes a selection and advises the Symbolic Controller accordingly (note that the Diagnostician may advise that the signal is of an "unknown-type."). The partitioning of the problem domain minimizes the overhead of each LAE and allows for faster processing of the data at the local level. Tools optimal in the local problem area are used, rather than generic, all-powerful, and time consuming tools needed if only one global expert were used.

These local experts, for any specific application, have fundamental actions and attributes. First they will access a knowledge base to obtain the 'expertise' for their problem area. Second, tools needed to suitably process information in a manner specific for their expertise are loaded from a tool library. Third, as data is processed, there is visible to the Diagnostic Controller a confidence value, reflecting the confidence of the LAE at that instant; and fourth, a status value is also visible which reflects the completeness of the LAE's analysis. In particular, as with specialized tools for a local problem area, the confidence and status values are also customized to reflect the needs of the LAE (i.e. actual computation strategies are a function of the area of expertise and are not required to be homogeneous across the collection of all experts).

KNOWLEDGE BASES

At system startup or reset, all LAEs access knowledge bases and use their respective names to extract knowledge stored there. For the software radio application, two knowledge bases are used; an LAE Knowledge Base and a Tool Library. The LAE Knowledge Base contains AND/OR graphs representing available modulation schemes and a list of tools tailored to that modulation type. A second knowledge base, a Tool Library, houses all tools needed in the analysis process. These tools are: 1) numeric, such as the FFT; 2) symbolic, such as a program to process waveform attributes; and 3) algorithms incorporating both symbolic/numeric functions, such as a program to process waveform data into primitives suitable for symbolic processing ([12] for example).

Designing the knowledge bases in this manner allows for data to be logically collected. For example, when a new modulation scheme is added to the knowledge base, there may be a need to recognize new features. All that need be done is to design a new tool, place it in the Tool Library, and add the AND/OR graph plus the corresponding tool list to the LAE Knowledge Base. If other modulation schemes can make use of this tool, modify the appropriate AND/OR graphs and add the new tool to the tool lists. If a new tool algorithm is found to be more efficient than the one currently in the Tool Library, simply replace the tool's code with new code and retain the tool name. The LAE Knowledge Base needs no altering.

Once startup is completed, the Symbolic Diagnostician is ready to process data. As new data is sent to the diagnostician, the Controller allows each LAE to copy the raw data into its environment and subsequently begin its expert analysis. Note that while this copy operation may be seemingly time consuming on a sequential machine, it preserves the integrity of the raw data. In future implementations, parallel processing is envisioned, thus necessitating the copying of data to other computation environments.

CONFIDENCE

In the AND/OR graphs, the confidence computation is affected by the type of connection between branches. For example, an AND connector requires substantial evidence on all branches for that connector to return a high confidence value. Any branch having low confidence dominates and causes the connector to return a correspondingly low value. In contrast, the OR connector returns a high confidence value if only one of the branches is strongly supported. Further, if a high value is returned for an OR connector, then searching other branches associated with that connector is suspended. Later, if time permits, the expert may return to these unexamined branches in the hopes of bettering its confidence. Hence, the AND and OR connectors are functions which combine branch confidences according to some specified scheme. The actual combining algorithms can vary from application to application. Possible calculation methodologies include those derived from Bayesian theory [13], from the Demptster-Shafer mathematical theory of evidence [14], fuzzy sets [15,16], or simply ad-hoc rules of thumb.

EXAMPLE LAEs

As an example, contrast possible LAEs which infer BPSK or AM modulation schemes. Figure 5.shows representations needed to distinguish these modulation schemes. BPSK has a frequency spectrum shape of $\frac{sin^2(x)}{x^2}$ ("structured symmetry") around the center frequency while AM has a delta function with "simple symmetry" around it. In the mnemonic FSC is used to mean "Find Shape's Center" and corresponds to the center frequency of the radio signal.

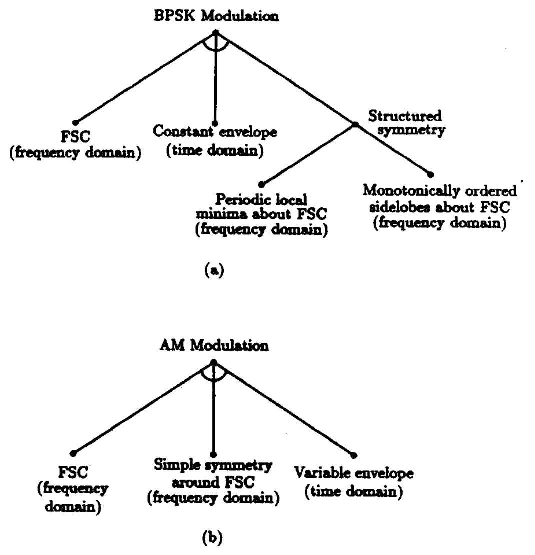
BPSK Modulation
FSC (frequency domain)
Constant envelope (time domain)
Structured symmetry
Periodic local minima about FSC (frequency domain)
Monotonically ordered sidelobes about FSC (frequency domain)
(a)
AM Modulation
FSC (frequency domain)
Simple symmetry around FSC (frequency domain)
Variable envelope (time domain)
(b)

Figure 5. AND/OR graphs for BPSK and AM modulation schemes

At startup, *LAE-BPSK* and *LAE-AM* access the LAE Knowledge Base and, using their names AM or BPSK as keys, load a symbolic description of figure 5a or 5b into their local environments. Note that both use frequency-domain and time-domain data. LAE-BPSK uses algorithms to extract features from the FFT data and to confirm 'constant envelope' in the time domain. LAE-AM uses algorithms to extract a different set of features from the FFT and to confirm 'variable envelope' in the time domain. The terminal nodes on the graphs denote specific tools used to support or deny the validity of each branch.

Next, the LAE examines the list of tools needed to process each branch of its respective AND/OR graph and thus detect the modulation scheme. The LAE ports these tools into its environment and begins analysis as the Controller directs. Dialogue continues between the Controller and each LAE until an opinion is ready to submit to the Symbolic Controller. Important parameters are also submitted to the controller, such as center frequency, bit rate, etc.

DESIGN IMPLEMENTATION

SOFTWARE DESIGN

The Symbolic Diagnostician's design relies heavily on the use of recent developments in the programming sciences. Specifically used is a programming technique known as Object-Oriented Programming (OOP), since it affords some options not readily available in structured programming [17]. Inheritance and the Message Passing communication technique [18,19] are two such options making it desirable. Inheritance here allows the logical partitioning of modulation schemes into classes and subclasses in a heiracharial style such that a particular modulation type has all the characteristics of the class or subclass and one or more unique characteristics used to distinguish it from other members of the same class. Message passing, on the other hand, allows a collection of objects, instantiations of a class, to dialogue with other objects through a vocabulary of defined words known as "methods." The internal code delineating the method is hidden -only object behavior is visible. It is the method's execution which causes the behavior of the object. This behavior includes the sending of messages, an altering of its state, or the issuing of function calls. The behavior patterns of many objects are thus used to build the desired behavior of the larger system.

The Symbolic Diagnostician instantiates two types of objects; the Diagnostic Controller and the Local Area Experts, which have the modulation type incorporated into their names. These objects dialogue by passing messages, some of which cause the the LAE's to execute numeric and/or symbolic methods and return an appropriate response, such as an opinion and analysis status.

Another reason for the use of OOP is for analogies which arise from potential hardware implementations. For a system to run real-time, it is desirable to have any algorithms which lend themselves to parallel computation to be executed in like manner. With the inference scheme proposed, it is natural to realize this design in hardware through parallel processing techniques. Individual modulation schemes can be designed as objects during preliminary developments and subsequently realized through the allocation of a sequential machine per object. The complete diagnostician is then a parallel implementation and can use the concepts of distributed processing systems [20,21,22].

As final comment concerning the OOP nature of this diagnostician, production-rule systems can be implemented at the LAE level if the need is present for that type of system (the current implementation does not use one per se). It is not the intent of the proposed design to preclude larger, more intelligent experts use but rather to encourage effective use of any technique in its appropriate domain such that cost/time constraints can be optimized. The final diagnostician structure may use a collage of symbolic paradigms resident within multiple LAEs.

The need for symbolic and numeric algorithms working in tandem requires an environment which supports appropriate language types. Current development is being done on a VAX single CPU machine using a version of Common LISP developed by LUCID which will interface with numeric algorithms written in FORTRAN or C. Embedded in this version of Common Lisp is an object coding system known as "Flavors" which is being used to design the Symbolic Diagnostician. Symbolic algorithms used by various objects are written in LISP while the numeric algorithms are written in FORTRAN. Knowledge bases are currently implemented in LISP as 'Hash Tables.'

REAL-TIME CONSTRAINT

A final comment needs to be made concerning the real-time constraint, as this is of fundamental concern. The present design allows for the Controller to take the advice of the LAE with the highest confidence at a given moment. If the need of the moment is for quick answers (i.e. sometimes something is better than nothing), the local experts may not be able to engage in complete data analysis. Therefore, they are designed so that the current confidence, initially set low, is updated as a branch of the AND/OR graph finishes its calculations. These "running confidences" are visible to the Diagnostic Controller which can, at a moment's notice, choose the modulation scheme with the highest instantaneous confidence value or, if the Controller is using a "history" of the previous schemes detected, choose some other scheme. Local experts may continue to process data and update their confidence values, and the new values used by the Controller at a later time if it so chooses.

CONCLUSION

This paper has described a general intelligent control paradigm with the intent of detailing one specific aspect of the proposed structure, that of symbolic diagnosis. A target application, that of a software radio, has been used to stimulate development of this diagnostician since many of the constraints imposed by such an application are found in the conceived generic control paradigm. Its design is such that the an expert may be implemented through the partitioning of the problem space into suitable subspaces where a local area expert can use tools optimal for the subspace. In like manner, confidence values are calculated according to an algorithm appropriate for the local problem space. Knowledge bases are designed so that relevant data can be grouped logically, allowing easier maintenance of the data bases. The reported effort uses Object-Oriented Programming techniques to support design of the Symbolic Diagnostician as well as for the potential hardware implementations allowing for potential solution of the real-time constraint.

REFERENCES

[1] K.S. Fu, "Learning Control Systems and Intelligent Control System: An Intersection of Artificial Intelligence and Automatic Control," IEEE Transactions on Automatic Control, Vol. AC-16, no. 1, pp. 70-72, 1971.

[2] H. Anderson, "Why Artificial Intelligence Isn't (Yet)," AI EXPERT, vol. 2, no. 7, pp. 36-44, July, 1987.

[3] R. H. Anderson and R. B. Greenberg, "UNIX and AI, A Beautiful Marriage," UNIX WORLD, pp. 26-33, August, 1986.

[4] G. N. Saridis, "Knowledge Implementation: Structures of Intelligent Control Systems," Proceedings of the IEEE International Symposium on Intelligent Control, Philadelphia, Penn., Jan., 1987.

[5] J. H. Painter, S. K. Lin, and E. Glass, "A Knowledge-based Control Paradigm for Real-time Systems," Third IEEE International Symposium on Intelligent Control, Arlington, Va., Aug. 24-26, 1988.

[6] R. Davis and B. G. Buchanan, "Meta-Level Knowledge: Overview and Applications," The 5th International Joint Conference On Artificial Intelligence, Cambridge. MA., August, 1977.

[7] T. A. C. M. Claasen and W. F. G.Mechlenbrauker, "The Wigner Distribution - A Tool for Time-Frequency Signal Analysis," Phillips Journal of Research, Vol. 35, pp. 217-250, 1980.

[8] G. C. Stockman and L. N. Kanal, "Problem Reduction and Representation for the Linguistic Analysis of Waveforms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-5, no.3, pp. 287-298, May, 1983.

[9] Y-C. Cheng and S-Y Lu, "Waveform Correlation by Tree Matching," IEEE Transactions. Pattern Analysis and Machine Intelligence, vol. PAMI-7, no. 3, pp. 299-305, May, 1985.

[10] R. M. Hueschen, and J. W. McManus, "Application of AI Methods to Aircraft Guidence and Control," American Control Conference, Atlanta, Ga., June 15-17, 1988.

[11] J. B. Cruz, and A. R. Stubberud, "Knowledge-Based Approach to Multiple Control Coordination in Complex Systems," Proceedings of the IEEE International Symposium on Intelligent Control Philadelphia, Penn., Jan., 1987.

[12] Kuo,"Well Log Correlation Using Artificial Intelligence." Dissertation, Department of Petroleum Engineering, Texas A&M University, August, 1986.

[13] J. Ihara, "Extension of Conditional Probability and Measures of Belief and Disbelief in a Hypothesis Based on Uncertain Evidence," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-9, no. 4, pp.561-568, July, 1987.

[14] R. R. Yager, "Arithmetic and Other Operations on Dempster-Shafer Structures, " International Journal Man-Machine Studies, vol. 25, no. 4, pp.357-366, October, 1986.

[15] L. A. Zadeh, "Fuzzy Sets as a Basis for a Theory of Possibility," Fuzzy Sets and Systems, vol. 1, pp.3-28, 1978.

[16] D. Dubois and H. Prade, "The Treatment of Uncertainty in Knowledge-Based Systems Using Fuzzy Sets and Possibility Theory," International Journal of Intelligence Systems, vol. 3, no.2, pp.141-65, Summer, 1988.

[17] J. Stein, "Object-Oriented Programming and Databases", Dr. Dobb's Journal, vol. 13, no. 3, pp.18-34, March, 1988.

[18] M. Stefik and D. G. Bobrow, "Object-Oriented Programming: Themes and Variations," The AI Magazine, vol. 6, no. 4, pp. 40-62, Winter, 1986.

[19] J. A. Stankovic, "Software Communication Mechanisms: Procedure Calls Versus Messages," Computer, vol. 15, no. 4, pp. 19-25, April, 1982.

[20] S. M. Shatz, "Communication Mechanisms of Programming Distributed Systems," Computer, vol. 17, no. 6, pp. 21-27, June, 1984.

[21] P. B. Hansen, "Distributed Process: A Concurrent Programming Concept," Communications of the ACM, vol. 21, no. 11, pp. 934-941, November, 1978.

[22] A. Z. Spector, "Performing Remote Operations Efficiently on a Local Computer Network," Communications of the ACM, vol. 25, no. 4, pp246-260, April, 1982.