KNOWLEDGE–BASED CONTROL

by
John H. Painter
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843–3128
409, 845–7441
painter@zadok.tamu.edu

''If we could first know where we are, and whither we are tending, we could better judge what to do, and how to do it.''
--A. Lincoln; June 16, 1858.

''The heart of the prudent getteth knowledge and the ear of the wise seeketh knowledge.'' [Proverbs 18:15].
--King Solomon; circa 1,000 B.C..

ABSTRACT

Knowledge–based Control is here defined as the management of dynamic systems whose states admit qualitative modeling. That is, dynamic systems are modeled by assigning qualitative (linguistic) descriptors, such as operating modes, to partitions of a direct–product space of system states. The state–variables may include non–numerical as well as numerical variables. By "management" is meant the identification (decision) of current system qualitative state, followed by formulation of system inputs (control) to move the state to a desired next state. These decision and control processes are implemented in the presence of uncertainty in the qualitative state (mode) definitions.

This paper examines contributions from several disparate disciplines, such as artificial intelligence, the decision sciences, and the emerging area of fuzzy control. An aeronautical application is used as a means to obtain visibility of the concepts examined. Two levels of architecture are presented for implementing qualitative decision and control for an aircraft. A geometric approach, rather than algebraic, is taken to the knowledge–based control problem.

INFERENCE AND CONTROL

INFERENCE EMBEDDED IN CONTROL.

In 1858, Abraham Lincoln pretty well made the case for Knowledge–based Control with his statement about knowing first and doing second. Down through the centuries, it has been appreciated that knowledge is the precursor to prudent action. So it is, with the merging area which is the subject of the present paper.
knowledge is the precursor to prudent action. So it is, with the emerging area which is the subject of the present paper.

Knowledge–based Control, also known as Intelligent Control, comes from the *Systems and Control* discipline, as a branch which is traceable back to a seminal paper by K. S. Fu[1]. Twenty years ago, he recognized that automatic control should benefit from developments in *Artificial Intelligence* (AI), if not vice versa. That these benefits have been so long in coming, is a comment upon the difficulties inherent in AI development and in transfer between two very different technologies. It is one purpose of the present paper to point out dualities between AI and Control which make the transfer easier.

Since the 1950's, it has been recognized that inference is an embedded element of modern control. Bellman's[2] development of dynamic programming employed explicit inference, in the form of decision, as an integral step of control. Kalman's[3] famous contribution also showed inference, specifically prediction, to be inherent in (state) control. Both examples of inference embedded in control illustrate what is formally codified in the now famous *Separation Theorem*[4] of stochastic control. The theorem states that, under fairly general conditions, a *reasonable* closed–loop control system may be formally partitioned into three parts; comprising the system (plant) under control, an inferential estimator, and a memoryless controller.

This work was supported, in part, by the National Aeronautics and Space Administration, under NASA Grants NAG1–1066 (Langley Research Center) and NGT–50401 (NASA Headquarters).

The difference between traditional *Modern Control* and Knowledge–based Control is that the embedded inference and memoryless control algorithms now contain qualitative, as well as numerical, processing elements. That is, the inferences and resulting controls are focused on qualitative performance measures, as well as numerical. As a dual to the usual required state–space modeling of the plant, now there is the required modeling of *qualitative states*.

THE PARADIGM.

The general pattern in knowledge–based control of a dynamic system is as follows: First, the usual numerical state–variable modeling of the plant takes place, with a preference given to states which are available as sensor readings, or are, at least, easily derivable (observable) therefrom. Next, a state–space is constructed, which is the *Cartesian product* of the states. (Actually, it is a direct–product space, since the door is left open to measurable states which are inherently qualitative, and not numerical.) Then, and most importantly, qualitative states are defined as partitions of the space. These qualitative states are most generally "*operating modes,*" as defined by a human operator. Such modes are not directly measurable, but are observed by (human) inference processing of the available sensor measurements. Control is then exerted to maintain or modify these inferred states (modes).

As an example, consider the management of the flight of a transport–type aircraft. By *management* is meant an automation of the control activities traditionally performed by the pilot and/or flight engineer. It is assumed that the usual numerical automatic flight control system (AFCS) is present. Thus, the pilot concerns himself with evaluation of flight performance and formulation of inputs to the autopilot, to accomplish the goals of the flight.

The usual sensor readings are assumed to be available, as in Figure–1, below.

SPEED	ALT.	FORW_∠	LAT_∠	HEADING	POWER	RESOURC.	AUX_DEV.
IAS Mach.	Alt.	Pitch	Bank	Mag. Comp. (TH)	EPR EGT N(rpm) Eng–#	Fuel_Qty.	Flaps Slats Gear Spd_Brk

Figure 1. Raw Numerical Data List.

Based on these available readings, and others derivable from them, qualitative operating modes are defined, as in Figure–2. Given the sensor readings and the qualitative definitions, mode is inferred and control exerted, based on knowledge of the aircraft and of the rules of flight.

The list of operating modes is assumed finite, with *mode_unknown* covering the rest of the possibilities. Thus, the required inference is decision, rather than estimation. That is, given the sensor measurements, it is to be decided into which partition of the direct–product space the observed "*point*" fits. Given that decision, an input to the AFCS is then formulated, based on known goals and characteristics of the aircraft.

ANTECEDENTS.

Although the above paradigm sounds simple (and it is accomplished routinely by human pilots) the automation thereof is not trivial for several reasons. First, in the definition of operating modes, the partitioning of the state space, is not clean. That is, the operating modes are not unique, in terms of the sensor measurements. Second, the decision inference method is not obvious. This is because reasonable limiting of the number of operating modes may cause some sensor data to apparently contradict a mode that is "*essentially true*". The decision method must accommodate these anomalies. Finally, the method for synthesizing qualitative commands, accompanied by numerical prescriptions, based on qualitative performance interpretations, it also not obvious. Thus, is examined the *information and control* discipline, as well as the *artificial intelligence* discipline to determine what priori results might apply to the present problem.

DISCRETE–EVENT DYNAMIC SYSTEMS.

A particular branch of control theory which has been active for twenty years or more is that of discrete–event dynamic systems (DEDS). This discipline has attempted to create, for dynamic systems exhibiting a finite (or at least countable) number of states, a theoretical basis parallel to that ex-

MODE	STATUS	CONTROL	MANUVER	CONSTRAINTS
Takeoff Clean_up Climb_out Climb_on_course Cruise Descend_on_course Initial_approach Final_approach Go_around Land Unknown	Normal Emergency Alarms: Systems, Fuel.	Pilot in Command. Air Traffic Control Other	Pilot Computer/ Autopilot/ Flight Control system.	Manuver Power Fuel Systems Weather

Figure 2. Flight Operations List.

isting for continuous–state dynamic systems. Upon these discrete state spaces, "*events*" are defined as the causes of the abrupt transitions of the state–vector between its discrete values. A recent paper by Ramage and Wonham[5] provides a clear summary of the application and methods of DEDS. Effort has focused on *admissible event trajectories* through the state space (sequences of events). In the inference context, effort is to determine if, given a certain sequence property, an observed sequence exhibits that property. In the control context, a trajectory is controlled to have the desired property. Methods of analysis have been *logical*, if time is not a consideration, or *timed*. Among the timed approaches are non–stochastic (Petri nets) or stochastic (Markov processes). Generally, the analytical approaches have been arithmetic. That is, they have not been geometric.

In the paradigm previously described (aircraft example), a finite set of events is defined on a continuous state–space. In the decision context, the events are not the causes of transitions from one region of the space to another, but are the partitions of the space, itself. Interest, here, is not in a sequence of operational modes, but in identifying the individual modes. In the control context, interest is not in some average control of a sequence of modes, but in just moving very deterministically to a succeeding mode. Finally, modeling and processing methods are desired, which give maximum insight (visibility) into the nature of the problem, itself. Thus, the search through antecedents is continued.

EXPERT SYSTEMS: FIRST GENERATION.

Next is that branch of AI, known as Expert Systems. Therein are programs particularly constructed to perform qualitative inference. It is natural to inquire whether such software might be employed to perform the embedded inference required in Knowledge–based Control.

The success of early Expert System work devolved from the development of the "General Problem Solver (GPS)" by Newall and Simon[6], in the 1960's, which provided the foundation of much of the subsequent development in Expert Systems. According to Giarratano and Riley[7], GPS produced the now commonly accepted architecture for an expert system, comprising i)–long–term memory (rules, ii)–short–term memory (working memory), and iii)–cognitive processor (inference engine). GPS was founded upon the assumption that much machine *reasoning* could be done, based on IF–THEN types of rules. This assumption was shown by subsequent events to be true.

Winston[8] related that GPS focused on a state description of the problem to be solved. In particular, it was goal driven, to move the current state of the problem to some goal state. A distance measure was employed upon current and goal states to measure progress toward a solution. This procedure is dual, of course, to feedback control theory, wherein an error between input and output drives the system. Since the state of the GPS problem was not numerical, the problem control was by search through a tree, to move current state to goal state. In modern expert system parlance, this was *forward–chaining, depth–first search.*

The original Newell and Simon approach dominated expert system work for twenty or more years. And, this approach was marked by a preference for algebraic, logical processing over what might be termed the geometric approach, the latter based on appreciation of natural organizations of knowledge–rich domain–specific problems. Thus, expert systems tools and shells were created, to which the knowledge representation and inference procedures

for diverse domain–specific problems must needs be conformed. This paradigm came to be known as the "First Generation" of expert systems. It reached a limit to growth, a barrier characterized by ***brittleness***, the inability to respond to increasing problem sophistication with a complementary sophistication of inference. Thus, the characterization by Giarratano and Riley[7] as "the eternal beginner."

THE DECISION SCIENCES.

There is another discipline, known as ***Decision Science*** (DS), interested in problems very similar to that of Intelligent Control. It owes its existence to ancestors in industrial engineering, operations research, econometrics, cybernetics, etc. A close examination of the contributions of Expert Systems to problems of interest in the Decision Sciences community has been made by Sutherland[9]. Because Sutherland's formulation of the decision problem is so congruent to the problem of knowledge-based control, his precepts are briefly reviewed here.

Sutherland first defined the decision problem as being conjunctive. That is, it had two sequential parts, being, first, the recognition of an event (e), and, second, the formulation of a response (r). Both event and response are characterizable in terms of an abstract ***state***. Moreover, for an action–oriented problem class, there existed a ***function*** (i.e., a mapping), f:(e,r), relating response to recognized event. He identified sixteen possible cases, being all (e,r) combinations resulting from four degrees of ***uncertainty*** in either event or response. The four degrees of uncertainty (from the stochastic control viewpoint, not Sutherland's) ranged from both event and response being purely deterministic to both wholly stochastic.

Sutherland showed that Decision Science and Computer Science (AI) had not (circa 1985) been working on the same problems. Specifically, AI had been focused on either event–recognition or response–formulation, while DS has focused on the conjunction of the two. His bottom–line assessment was that (first–generation) expert systems had not evolved to the point of being able to handle problems which are characterized by a large measure of uncertainty in the data. His reasoning was that the purely logical (arithmetic) algorithms of the first–generation inference engines would have to yield to facilities capable of dealing mathematically with "***graded qualitative valuations.***" He prophesied that AI would have to embrace certain elements of "***fuzzy technology.***"

FUZZY METHODS.

Fuzzy Set Theory was created by Zadeh[10] twenty–five years ago. Within five years, its use in decision theory was being examined[11]. Within nine years, its use in control as being promoted[12] Later (after 1986), it was implemented in practical control systems[13].

Fuzzy Set Theory is a dual of Bayesian Conditional Probability Theory, in the sense that ***events*** may be modeled by set functions whose range is the positive reals. Moreover, these functions may be manipulated to create similar positive real functions defined on subsets of the original domain. Therefore, in both cases, the set functions may be used to calculate the solutions of decision problems.

As a case upon which to focus, consider the airplane example. There the real–number domain might be the sensor reading, "Indicated Air Speed," IAS. On this set of positive numbers are defined two events, being "Takeoff" and "Landing." Now, the decision problem is to take an airspeed reading and to make the (hard) decision on whether the aircraft is taking off or landing.

In the Fuzzy context, a prior knowledge about the aircraft is used to create two functions. One is representative of certainty about what airspeeds should be observable during takeoff. The other models landing. These functions are called "***Membership Functions,***" according to the idea that the observed airspeed has membership in the subsets supporting the two disparate event definitions. These two functions are unimodal (by design) and are arbitrarily scaled to have unity maximum value. The decision is obtained by choosing the event whose membership function is greatest at the value of the sensed airspeed. See Figure–3, below, for an illustration.

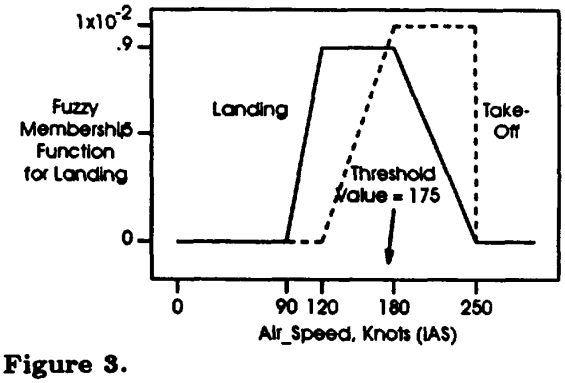

Figure 3.
Membership Functions for Takeoff/Landing.

In the Bayesian context, the functions formed are so–called "*a priori conditional probability density functions.*" They are formed in exactly the same manner and have the same shape, as for the fuzzy membership functions. However, they are not arbitrarily scaled, but have unit area. The reason for the unit area requirement, is that, to obtain functions for other events defined on the domain, the combining algorithms do not admit arbitrarily scaled functions. The same decision rule is employed, which is called "Maximum–Likelihood Decision Rule."

Now, it is seen that it doesn't matter what the functions are called, so long as the same decision rule is used. Scaling, however, does matter. It can be seen from Figure–3 that while the Bayes decision threshold is 175 knots, the Fuzzy decision threshold is 180. Thus, there is a relative decision bias of 5 knots between the two. For an indicated air speed of 177 knots, the two methods yield different decision. Figure–3 plots the membership functions with the required Bayes scaling.

Fuzzy Control[14] is decision–directed[15] control. By that is meant that controls implemented from a fuzzy theory basis result as the conjunctive solution of two decision problems. First, based on measured sensor data, it is decided what event has taken place. Second, given that event, it is decided what control to exert.

Fuzzy Control requires two sets of membership functions. The first relates sensor data to event. The second relates event to control action. A joint, two–dimensional membership function may then be synthesized, relating measured sensor data directly to control. The result is then a "*fuzzy set of control*"[14], which must be "*defuzzified*", in order to obtain a unique control effort.

In manipulating the various fuzzy membership functions, there are two different sets of combining algorithms which may be used. The first, and currently most popular, employs the "*conjunction*" and "*disjunction*", characterized by taking pointwise minima and maxima, respectively, of two membership functions. These are described as the "*hard logical AND*" and "*hard logical OR*", respectively. The other set is just the usual "*soft*" logical connectives[11]. It is worthwhile to note that if the soft connectives are used with unit area membership functions, there results just the usual Bayes conditional probability density functions for subsets of the domain set. Subset membership functions derived using the two differing sets of connectives may or may not have the same shape (ignoring scaling).

It is concluded that the general ideas of fuzzy control may be applied to the present problem, even though specific implementations may use the standard (soft) Bayes manipulations. What is important is that there exists support for qualitative inference and control which is essentially geometric, rather than algebraic. If, now, expert systems may admit these geometric approaches to modeling, qualitative inference, and control, a solution to knowledge–based control will be synthesizable therefrom.

EXPERT SYSTEMS:

SECOND GENERATION.

The first generation of expert systems was characterized by logic–based implementations. A high–level computer language (PROLOG) was even developed to support such implementations. However, use of that language required that the problem be formulated in the first–order predicate calculus. Thus, the design of expert systems was driven by the programming necessary to embody the paradigm descended from the pioneering efforts of Newell and Simon[6]. The problem definition had necessarily to be shaped into the form required by existing expert system tools. Thus, was born a need for *Knowledge Engineers*, whose task it was to translate the natural problem requirements into forms suitable to programming with the extant tools. Expert system development became characterized by programmers learning enough about the problem domain to develop efficient expert systems.

With the advent of the second generation, the so–called *domain experts* (engineers, in the present context), become the expert system developers, with programming methodology no longer dominating the effort. This is because of a shift in focus on the task of translating problem requirements into programmable form. The formulation is done at a much higher level than previously. Whereas, the problem was abstracted at the programming level, now the problem is abstracted at a level dealing with a generic combination of problem, knowledge representation, and inference strategy[16]. Now, domain experts (in the information and control sciences) may learn enough about symbolic computing to specify the inference engine and knowledge representation down to the (object–oriented) programming level.

Bylander and Chandrasekaran[16] draw attention to the fact that knowledge representation, if unconstrained by a priori programming requirements, is strongly influenced by the combination of problem nature and inference strategy. Thus, there is

a knowledge representation which is natural to that combination. "*Natural knowledge representation*" is referred to in [16] as the "*interaction problem.*" There, Minsky[17] is cited as having proposed *frames* as a knowledge representation suitable to the interaction problem. They[16] note that the emphasis in frame representations is on describing the conceptual structure of the domain.

It is intuitive to an engineer in the information and control sciences to characterize a problem by its *functional flow.* That is, the information processing functions being performed are specifically isolated and defined, from the overall problem. Then, the flow of data through these functions is diagramed, according to the (problem–based) natural sequence of operations. Then, the functionality implied by the diagram is mapped (sometimes directly) into hardware and/or software. In computer science, this is called the "*data--flow*"[18] architecture. As an example, the top–level data–flow diagram for the knowledge–based aircraft management problem is as shown below in Figure–4.

of inference functions, suitably generic to cover the domain of dynamic systems. Such a set of useful functions has been the subject of much work by Chandrasekaran[19] and his associates[16] [20] during the last several years. The refinement of the generic function approach shall well serve the present purposes.

A SPECIFIC APPROACH.

MODEL–BASED KNOWLEDGE REPRESENTATION.

Traditional first–generation expert systems typically consisted of a knowledge base and an inference engine, separately. Two standard approaches to knowledge representation were Production Rules and Frames[21]. Such representations must, in general, provide both syntax and semantics, defining symbols to be used and specifying how meaning is to be attached to the arrangement of symbols[8]. Such a representation may be described as a *seman-*

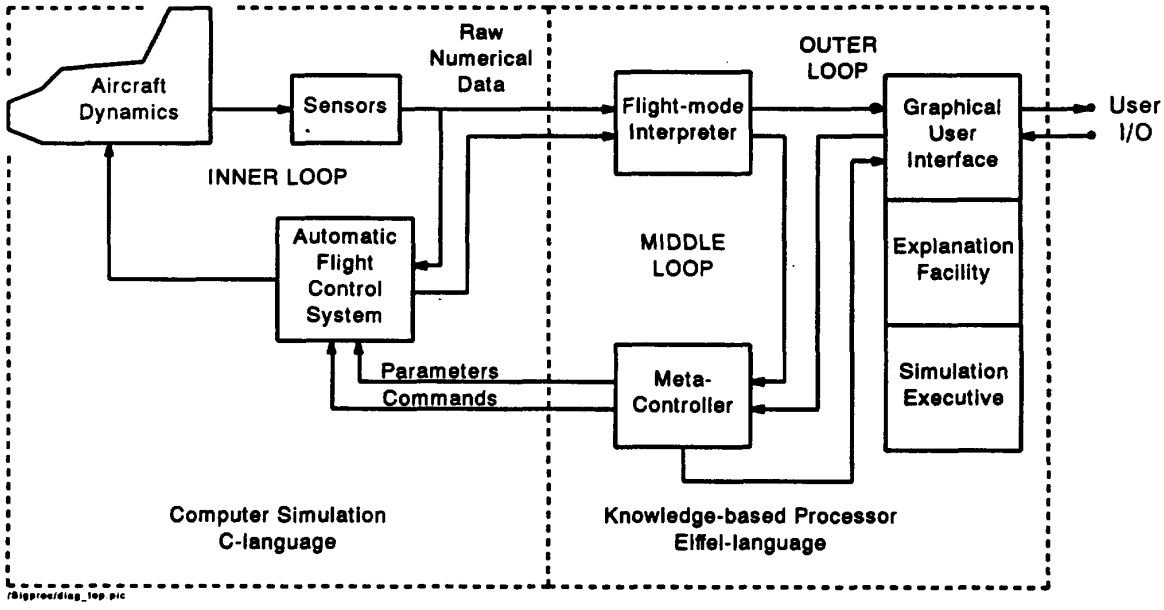

Figure 4. Top–level Data–flow Diagram.

The natural abstraction of the knowledge–based control problem requires the isolation and definition of the functions (or tasks) to be performed by the inference engines. Note that a complete knowledge representation is not obtained without consideration of the inference process.

It has been noted that knowledge–based control is decision–based, requiring a formal dealing with uncertainty. What is needed is definition of a set

tic net, comprised of nodes and links. The nodes are data objects and the links are relations, following the nomenclature common to object–oriented modeling and programming[22].

What may be generally viewed as a semantic net may, in the dynamic system domain, better be viewed as a hierarchical inference tree. Therein is defined a hierarchical set of decision hypotheses, with the most general at the top. The most *general*

hypotheses are the most generally defined operating modes. These are defined in terms of linguistic variables. The least general, at the bottom of the tree, are the sensor numerical data corresponding to the linguistic operating modes.

The decision inference is inductive, proceeding up the tree. Control inference is deductive, proceeding down the tree. For a strictly rule–based implementation, these two would correspond to forward–chaining and backward–chaining, respectively.

An instance of such an inference tree, for aircraft, is set forth as Figure–5, below.

Another refinement is added, which further justifies the frames choice. And, that is that each frame (–level) is treated as an object in an object oriented programming environment. Then, procedures (methods) are attached to each slot in each frame, implementing the inference (decision) process from level to level. The inheritance mechanism may be used to advantage, here, to produce efficient, readable code.

For this knowledge representation, a complementary inference procedure is now specified.

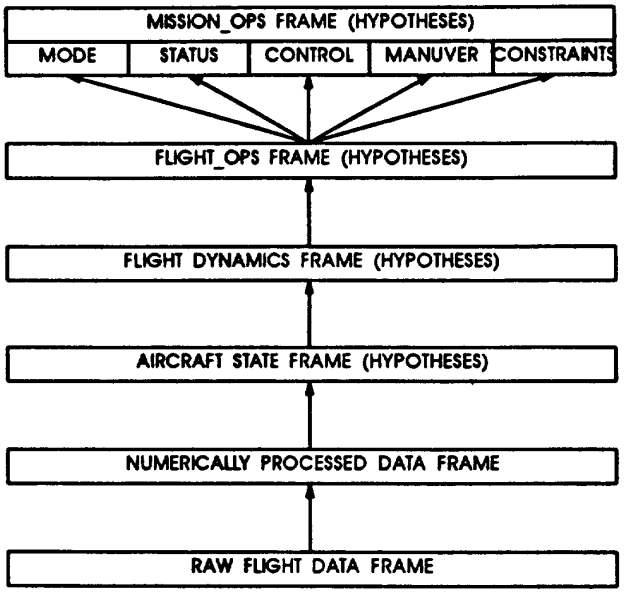

Figure 5. Inference–Task, Data Representation.

The figure shows four levels of hypotheses. These are compound hypotheses, made up of more elementary hypotheses, having names like "MODE," "STATUS," "CONTROL," "MANUEVER," and "CONSTRAINTS." The highest level concerns the mission of the dynamic system. The next level concerns its operation as a system. Then is a level concerning its dynamics. The next concerns its individual qualitative states. Then is a level of description (which is not a decision hypothesis), being its numerical states, not raw, but after numerical processing. Finally, is the level of raw numerical sensor (and other non–hypothetical) data.

Viewing compound hypotheses as sets, as in decision science, each level of description has the appearance of a direct–product space of linguistic variables. As an analyst thinks of a vector space for numerical variables, a programmer may then think of a data frame for linguistic variables.

ABDUCTIVE INFERENCE.

In fuzzy inference, (Bayes) ***Maximum Likelihood*** decision is used. This algorithm computes the (soft) product of the various membership function. The algorithm works properly, provided there is no conflicting evidence present. However, suppose that some part of the dynamic system or operating procedure fails, not badly enough to throw the system into some other mode, but badly enough to yield data in conflict with the actual operating mode.

For example, consider an aircraft on final approach to landing, which is processing sensor measurement as indicated air speed (IAS), altitude (ALT), rate of climb (ROC), flight path angle (FPA), engine power (EPR), FLAPS, and GEAR, in order to make the interpreted decision "MODE_FLT_OPS

= APPR_FNL". The variable, GEAR, will enter the computation as P(GEAR/LAND)=0, where $P(*|*)$ denotes membership function. Thus, the entire ML–product will be reduced to zero, and the computation will fail to yield a decision. Now, it is clear that the airplane is still on final approach, but the gear just hasn't come down (for whatever reason). What is needed is a decision computation which will still yield "MODE_FLT_OPS = APPR_FNL," but will also yield "STATUS_FLT_OPS = ALARM_GEAR."

This example shows that rudimentary (Bayes or Fuzzy) decision won't suffice. What is needed is a modified decision rule, such that the fuzzy foundation is retained, but conflicting evidence is accommodated. A decision framework which admits this modification is that of ABDUCTIVE Inference, as formulated in the Chandrasekaran group and clearly reported by Punch[20], et. al. Figure–6 shows a functional flow diagram for ABDUCTIVE Inference.

The ***consistency*** requirement is that the elementary hypotheses must be compatible with each other. That is, that, pairwise, they are not mutually exclusive. The ***completeness***, or coverage, requirement is that the compound hypothesis contains all the plausible elementary hypotheses. This definition of ***covers*** is compatible with the set theoretic definition, considering the compound hypothesis as a direct–product. This means that no plausible elementary hypothesis may be ignored. The last requirement, on ***plausibility***, is to select the compound hypothesis which has the highest plausibility score. If a plausibility threshold is used to remove ***implausible*** elementary hypotheses from the computation, then the Bayes–ML algorithm can be used.

The key to designing the Abductive Inference Engine lies with its internal control. Punch[20], et. al., showed a tree–like goal structure (their Fig.–5) which formed the basis for internal control of

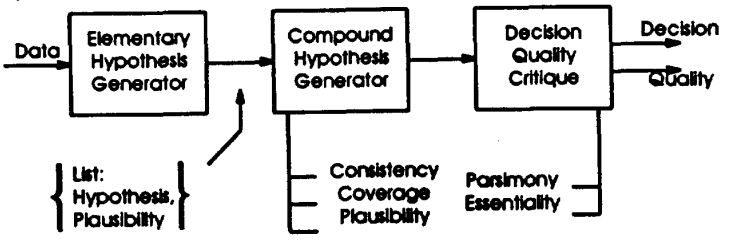

Figure 6. ABDUCTIVE Inference Functional Diagram

The first functional element is an Elementary Hypothesis Generator. This produces a list, with each elementary hypothesis tagged with its corresponding ***plausibility***. The plausibility is a numerical measure of the likelihood of the elementary hypothesis. It may also be called a ***Confidence Factor***. For the airplane, an elementary hypothesis is $\{LAND|IAS\}$. A plausible hypothesis is then one whose numerical plausibility value exceeds some predefined threshold.

The next functional element in abductive (decision) inference is a Compound Hypothesis Assembler. Its task is to assemble, from the elementary hypotheses, the ***most plausible*** compound hypothesis. This is also what Bayes–ML does. However, the Abductive Assembler does this in a constrained way. It satisfied internal requirements of 1). Consistency, 2). Completeness, and 3). Plausibility.

the inference engine. Their control was dubbed "Selector–Sponsor," in which a Selector would select from among various Sponsors, each one of which sponsored a particular method, or task. That is, the Selector was a global controller, choosing from among the various subtasks for Abductive Inference. The Sponsor then evaluated it method's appropriateness to be the next task.

The control strategy for the dynamic systems problem is differentiated between the Interpretation task (inductive inference) and the System Control task (deductive inference). The concept of a community of distributed experts is employed, which is a parallel processing concept. Interpretation starts at the bottom of the frame structure (See Fig.–5), which the occurrence of new sensor measurements. Each measurement slot contains its own ***expert***. Each expert evaluates its owned membership functions. Each expert then selects its plausible hypotheses. Each expert then passes its data, as mes-

sages, to the higher levels of the framework, according to rules. Every slot in the framework functions in this same way.

Chandrasekaran's[19] list of generic functions is now augmented to deal with temporal (dynamic) systems. System Operational Modes, if ideally defined, are mutually exclusive and sequential. That is, they occur naturally in sequence. Therefore, if the last mode is known, a priori information is available about the present mode. Thus, is needed another generic task, that of ***History Formatting and Processing***. This is the dual of exploiting correlation in stochastic decision/estimation (e.g. Kalman filtering).

From all of the above, results the second–level data–flow architecture shown in Figure–7, below.

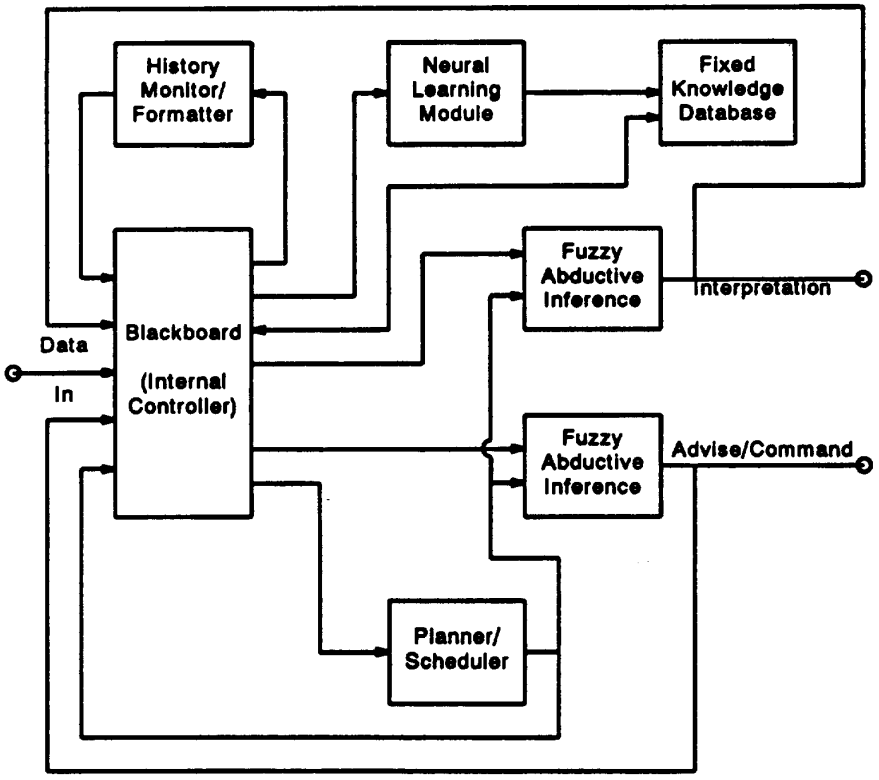

Figure 7. Symbolic Processor.

CONCLUSION.

What has been related here, is that Knowledge–based Control (or Management) of (complex) Dynamic Systems may be approached in a way that is very natural to practitioners of the information and control sciences. This approach is a synthesis of results evolving in several heretofore separate disciplines. Modeling and processing methods are favored which support visualization. Dualities are exploited, which exist between Artificial Intelligence and Systems Theory (Communication, Control, and Signal Processing). Based on research in progress, this approach appears to the author to hold great practical promise in applications like aircraft flight management.

BIBLIOGRAPHY.

[1] K. S. Fu, "Learning Control Systems and Knowledge–based Control Systems: an Intersection of Artificial Intelligence and Automatic Control, IEEE Transactions on Automatic Control, vol. AC–16, no. 1, 1971, pp. 70–72.

[2] R. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ., 1957.

[3] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Transactions of the ASME, Journal of Basic Engineering, vol. 82D, March, 1960, pp. 34–45.

[4] F. C. Schweppe, Uncertain Dynamic Systems, Prentice–Hall, 1973, pp. 479–489.

[5] P. J. G. Ramadge and W. M. Wonham, "The Control of Discrete Event Systems," Proceedings of the IEEE, vol. 77, no. 1, Jan. 1989, pp. 81–98,

[6] A. Newell and H. A. Simon, Expert Systems, Principles and Programming, PWS-KENT Publ. Co., Boston, 1989, pp. 11-15.

[7] J. Giarratano and G. Riley, Human Problem Solving, Prentice-Hall, 1972.

[8] P. H. Winston, Artificial Intelligence, Second Edition; Addison-Wesley, 1984, pp. 146-157.

[9] John W. Sutherland, "Assessing the Artificial Intelligence Contribution to Decision Technology," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-16, #1, Jan/Feb., 1986, pp. 3-20.

[10] L. A. Zadeh, "Fuzzy Sets," Information and Control, no. 8, 1965; pp. 338-353.

[11] R. E. Bellman and L. A. Zadeh, "Decision-making in a Fuzzy Environment," Management Science, no. 4, 1970, pp. 141-164.

[12] L. A. Zadeh, "A Rationale for Fuzzy Control," Transactions of the ASME, Series G (USA), no. 94, 1974, pp. 3-4/

[13] K. J. Astrom, J. G. Anton and K. E. Arzen, "Expert Control," Automatica, no. 13, 1986, pp. 277-286.

[14] W. Pedrycz, Fuzzy Control and Fuzzy Systems, John Wiley & Sons, Inc., 1989, pp. 61-80. (See p. 69 for "fuzzy set of control."

[15] J. H. Painter and S. K. Jones, "Results on Discrete-time, Decision-directed, Integrated Detection, Estimation, and Identification," IEEE Transactions on Communications, vol. COM-25, no. 7, July, 1977.

[16] Tom Bylander and B. Chandrasekaran, "Generic Tasks for Knowledge-based Reasoning: The "Right" Level of Abstraction for Knowledge Acquisition," International Journal of Man-Machine Studies, #26, 1987, pp. 231-243.

[17] M. Minsky, "A Framework for Representing Knowledge," The Psychology of Computer Vision, McGraw-Hill, 1975, pp. 211-277.

[18] Tom DeMarco, Structured Analysis and System Specification, Yourdon Press, 1978.

[19] B. Chandrasekaran, "Generic Tasks in Knowledge-based Reasoning: High-Level Building Blocks for Expert System Design," IEEE Expert, Fall 1986, pp. 23-30.

[20] W. F. Punch, M. C. Tanner, J. R. Joshephson and J. W. Smith, "Pierce, A Tool for Experimenting with Abduction," IEEE Expert, October, 1990, pp. 34-44.

[21] L. E. Widman, K. A. Loparo and N. R. Nielsen, Artificial Intelligence, Simulation, & Modeling, Wiley, 1989, pp. 8-10.

[22] S. Shlaer and S. J. Mellor, Object-Oriented Systems Analysis, Prentice-Hall (Yourdan Press), 1988.