IJRCS
International Journal of Robotics and Control Systems
Vol. 2, No. 1, 2022, pp. 201-229
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs
ASCEE

Methodologies and Applications of Artificial Intelligence in Systems Engineering

Awatef K. Ali [a,1], Magdi S. Mahmoud [b,2,*]

[a] Electronics Department, National Telecommunications Institute, Nasr City, Cairo-Egypt
[b] Control and Instrumentation Engineering Department, KFUPM, PO Box 5067, Dhahran 3126, Saudi Arabia
[1] 3watef@gmail.com; [2] msmahmoud@kfupm.edu.sa
[*] Corresponding Author

ARTICLE INFO

Article History
Received December 02, 2021
Revised December 26, 2021
Accepted March 04, 2022

Keywords
Knowledge-Based Systems;
Artificial Neural Networks;
Fuzzy Logic and Systems

ABSTRACT

This paper presents an overview of the methodologies and applications of artificially intelligent systems (AIS) in different engineering disciplines with the objective of unifying the basic information and outlining the main features. These are knowledge-based systems (KBS), artificial neural networks (ANN), and fuzzy logic and systems (FLS). To illustrate the concepts, merits, and demerits, a typical application is given from each methodology. The relationship between ANN and FLS is emphasized. Two recent developments are finally presented: one is intelligent and autonomous systems (IAS) with particular emphasis on intelligent vehicle and highway systems, and the other is the very large scale integration (VLSI) systems design, verification, and testing.

1. Introduction

Over the past four decades, numerous terms borrowed from psychology and biology such as adaptation, learning, pattern recognition, self-organization, and artificial intelligence have been introduced into the systems literature. We now reached the edge that these terms are necessarily imprecise and the ideas cannot compress into simple statements without vital loss content. However, we need such terms because they help us to communicate and advance the progress of our science Intelligence is a loaded word because of its strong association with human ability. Here, we adopt the practical view that an artificially intelligent object is one that employs some sort of self-adaptation (learning) in its functioning to face the external environment. Some interesting views on artificial intelligence (AI) are expressed in [1]. In the early theoretical development of artificial intelligence, there were strong ties to information, computer, and system sciences. One objective of this paper is to trace the development of Al and outline some related issues. Another objective is to bring together three methodologies that revolve around the concepts of Al and by their applications we obtain artificially intelligent systems.

The paper is organized as follows. Section 2 provides a brief exposition of background material in terms of notations and key elements. In Section 3, the subject of artificial intelligence (AI) is discussed and many of the common ideas are laid down. Section 4 (knowledge-based systems) and Section 5 (application 1) discuss the concept of knowledge processing and its use in typical system design application, (application 2) the artificially intelligent planning system AIPS, and (application 3) the use and implementation of AI in VLSI design and verification, fault diagnosis, and test generation. Section 6 (artificial neural networks) and Section 7 (application 4) jointly discuss the concept of neuro-modeling and computing as well as their use in typical system identification, and Section 8 discusses the concept of fuzzy modeling, analysis, and implementation together with their use in control system design. In Section 9 (application 5), the basic elements of the recently developed intelligent and autonomous systems (IASs) are presented with particular emphasis on intelligent vehicle and highway systems. Section 10 outlines the relationship between artificial neural networks and fuzzy systems. Finally, Section 11 concludes the paper.

2. Background

In order to appreciate the key role and importance of an artificially intelligent system (AIS), we provide in this section some information related to common terms and parts of AIS. Initially, we put some emphasis on the concept of decision-making and the associated decision analysis. We record the fact that decision making constitutes a key portion of human life and ranges from the routine and swift to the complex and time consuming. In typical cases, one would rely on systematic approach to analysis and solution. Decision analysis is a matter routinely faced by virtually any discipline whether or not those in that discipline have been exposed to any formal methods in of decision making. Needless to stress that we normally do not in support make decisions through decision analysis. Rather, we use the information provided by decision analysis to assist us in, hopefully, making better decisions. Thus, the purpose of decision analysis is to provide the decision maker with information for use in the support of the decision-support process, where such information has been derived through a logical, scientific, and systematic process. It is then up to the decision maker to decide how to interpret and use this supporting information in the determination of the ultimate decision.

Taking a systems standpoint, decision analysis might reasonably be viewed as a process that involves the transformation of data into (useful) information in support of the decision-making processes. This clarifies the common practice that data alone are of little benefit and to have value, it must be transformed into a format from which we can perceive such useful information. More specifically, one fundamental rule concerning data is that to be of value, data must be in the right form, in the right place, at the right time.

2.1. Management Information Systems (MIS)

Simply stated, MIS can be viewed as relevant knowledge produced by the output of data processing operations and acquired to achieve specific purposes. Such information is thus the basis connecting information network is vital to MIS. Typically, the concept of MIS involves real-time information processing that yields organized results.

2.2. Decision Support Systems (DSS)

Despite the basic role of MIS, they are relatively passive entities. While they remove the drudgery of data processing and development of visual aids and substantially decrease the time required to obtain such information, they still play a limited role in decision making. This is particularly relevant in view of the vast development of certain analytical methods for decision analysis including mathematical programming, queuing theory, marginal analysis, input-output analysis, and project scheduling. In turn, this paves the way to the realization of a decision support system (DSS) which can be viewed as a combination of an MIS and an analytical tool. Thus one conception of a DSS is that of a computerized system for accessing and processing data, developing managerial displays, and providing recommended courses of action. Using this definition, a block diagram of a general decision support system is depicted in Fig. 1. Note that, everything above the dashed line is assumed to contain within the DSS, in particular within the computers and computer networks employed by the DSS. It should be stressed that a DSS is a far more active participant in the decision-making procedure than either data processing (DP) or MIS. Finally, we remark that the analytical methods of DSS normally invoke the use of algorithms for the derivation of solutions for the particular class of mathematical models under consideration.

2.3. Heuristic Programming (HP)

Heuristic rules, or heuristics for short, are rules that are developed through intuition, experience, and judgment. Typically, they do not represent our knowledge of the design of, or interrelationships, within a system; rather they represent guidelines through which a system may be operated. Heuristics, sometimes called rules of thumb, do not necessarily result in the best, or optimal, result. However, in virtually any situation expertise involves the development and use of heuristics. One of the general characteristics of many heuristics is their focus on screening, filtering, or pruning. When one or more heuristics are combined with a procedure for deriving a solution from these rules, we have a heuristic program. Based thereon, heuristic programming is a procedure for finding the solution to a model consisting of heuristic rules. In this case, the solution sought is not optimal but rather acceptable or one that satisfies our predetermined aspirations.

Broadly speaking, we are in support of the idea that heuristics and heuristic programming, when and where appropriate, may enhance the decision-making procedure.

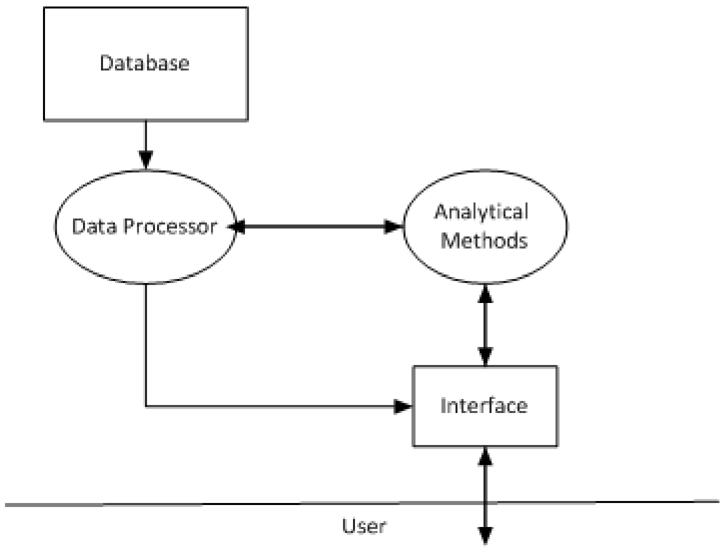


Fig. 1. A Generic Decision Support System

3. Artificial Intelligence: A Brief Account

It has been along standing argument among early users and designers of digital computers whether or not every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. The term artificial intelligence (AI) was coined some four decades ago to designate efforts along this new scientific direction [2]. By now, AI has grown and is becoming a technology achieving practicality that has recently attracted considerable publicity.

Adopting one viewpoint, AI is concerned with designing computer software to render the task of computing smarter. Thus, research in AI applications is focused on developing computational methodologies to intelligent behavior. By focusing on the computational aspects, a comparison between AI and conventional programming (CP) is presented in Table 1.

Another characteristic of AI programs is the extensive use of domain knowledge. For further

Table 1. Artificial Intelligence (AI) vs Conventional Programming (CP)

	AI	**CP**
1	Primarily Symbolic	Often Primarily Numeric
2	Heuristic Search (solution steps are implicit)	Algorithmic (solution steps are explicit)
3	Decision Structure Separate from Domain Knowledge	Information and Decision Integrated together
4	Easy to Modify, Update and Enlarge	Difficult to Modify
5	Some Incorrect Answers are Often Tolerable	Correct Answers are Required
6	Satisfactory Answers are Usually Acceptable	Best Possible Solution is Usually Sought
7	Exploratory Programming	Structured Programming

demonstration, basic elements of AI are displayed in Fig. 2, in which the inner ring depicts the main ingredients and from which the applications drawn in the second ring are composed.

AI problem solving can often be viewed as a search among alternatives choices [3]. It is thus possible to represent the resulting search space as a hierarchical structure called a tree and seek appropriate rules to guide the search procedure. An excellent exposition of these search techniques can be found in [3], [4]

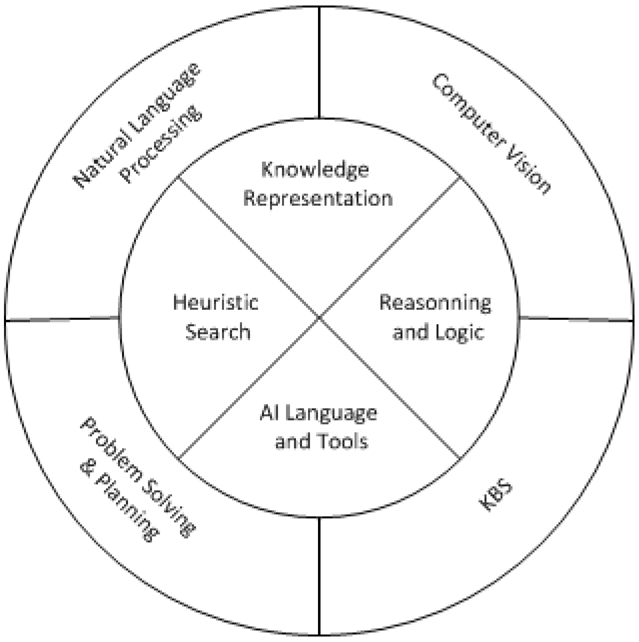


Fig. 2. Basic Elements of AI

Natural language processing is concerned with natural language front ends to computer programs, computer-based speech understanding, text generation and understanding and related applications [5].

Computer vision is concerned with enabling a computer see, to identify or understand what it sees, to locate what it is looking for, to verify the correctness of manufacture, etc. The link between computer vision and robotics is strong and yields useful applications [6].

Along another front, AI is concerned with precisely the problem that DSS and HP are concerned with, that is, decision making. One fundamental difference is that the objective of those in the AI community is considerably more ambitious than that of the DSS sector. Thus the purpose. of AI is not simply to support decision making; rather, the ultimate goal of AI is to develop an intelligent machine (synthetic being) that will itself make decisions. In particular, such a machine will be able to learn through experience, to recognize the limitations of its knowledge and to exhibit true creativity.

One difficulty of this goal is that there is no agreement as to precisely how one define, with much less measures, intelligence.

Looking at the side of applications, we record that most of our systems are complex and interconnected, like power systems, transportation networks, chemical plants, to name a few. This motivates the search for new tools of modeling, information processing and decision-making. Since AI can be viewed as the science of automating intelligent behavior [7], research interests were focused on merging techniques of AI and systems engineering. Among the fruitful results, three important methodologies are worth mentioning. These are:

1. Knowledge-Based Systems (KBSs)
2. Artificial Neural Networks (ANNs)
3. Fuzzy Logic and Systems (FLSs)

KBS are constructed (often in the form of a computer code) by obtaining the domain-specific knowledge (that is, knowledge of medicine from a medical doctor) and coding it into a form that a computer may apply to similar problems. In traditional KBS, uncertainty is handled by probability theory and reasoning is often done by probabilistic methods- probabilistic reasoning (PR). Neural networks, on the other hand, represent models of the human brain. In neural networks, the emphasis is on the brain's learning process, while traditional AI knowledge-based systems try to model the physical aspects of the brain [8]. Fuzzy logic, which stems from fuzzy set theory [9], deals with the vagueness, imprecision, and linguistic approach to human reasoning. Fuzzy logic and systems deal with a measure of vagueness as predicate logic deals with a measure of randomness.

Our purpose in this paper is to provide a modest coverage of the three forgoing methodologies and to present some of their potential engineering applications.

4. Knowledge-Based Systems (KBSs)

With emphasis on computational support, we formally define a knowledge-based system [10] as a computer program that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert.

In fact, this definition is typically used in a fair number of texts on KBSs. When properly interpreted, this definition serves to capture, but only to a certain degree, the expert systems concept. Alternatively, with focus on the development of quantitative (mathematical) models of the available knowledge we can introduce a knowledge-based system as a model and associated procedure that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert.

In this regard, the model is the representation of the knowledge base of the human expert. To proceed further, one possible representation of a knowledge-based system is depicted in Fig. 3, where the components above the dashed line are those within the computer. Below this line, access capabilities for two types of human users are noted. The first is a knowledge engineer is the person responsible for placing the knowledge into the knowledge-base of the KBS through the interface and rule adjuster. The second type is simply anyone who will be using KBS as a decision-making aid.

The interface handles all input to the computer and controls and formats all output. The inference engine (knowledge processor) is employed to perform two primary tasks. First, it examines the status of the knowledge base and working memory so as to determine what facts are known at any given time, and add any new facts that become available. Second, it provides for the control over the order in which the inferences are made. As the knowledge processing element of an KBS, the inference engine serves to merge facts with rules to develop, or infer, new facts.

The knowledge base is the very heart of any KBS. It typically contains two types of knowledge, that is, facts and rules. The facts within a knowledge base represents various aspects of al specific domain that are known prior to the implementation of an KBS. The rules within the knowledge base are simply heuristics of the type discussed earlier. If the knowledge base has been constructed through interaction with a human expert, these rules represent the knowledge engineer's perceptron of the heuristics that are employed by the expert in decision-making.

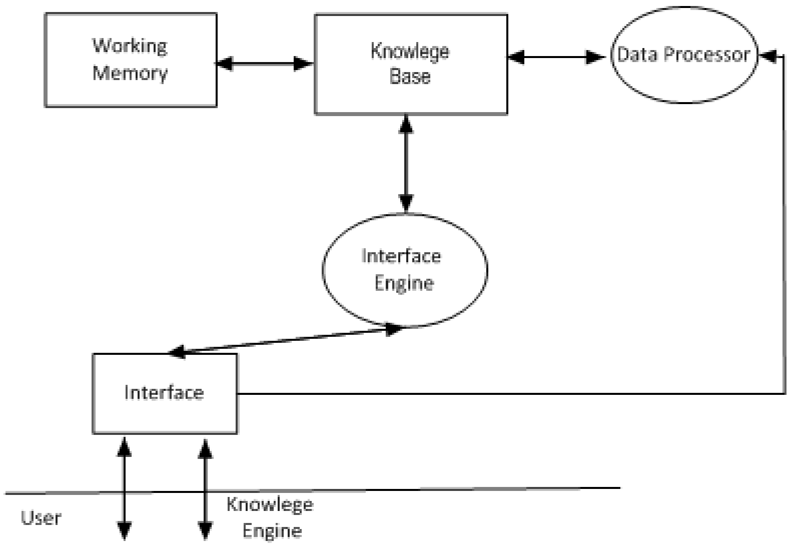


Fig. 3. A Generic Knowledge-Based System

The working memory of an KBS changes according to the specific problem at hand. The contents of the working memory consist of facts. However, unlike the facts within the knowledge base, these facts are those that have been determined for the specific problem under consideration during (and at the conclusion) of the implementation phase. More specifically, the results of the inference process are new facts and these facts are stored in the working memory.

The final module to be discussed is the rule adjuster. It merely serves as a rule editor, that is, it enters the rules specified by the knowledge engineer into the knowledge base during the development phase of KBS. It may also allow for various checks on these rules including consistency, completeness. It may also incorporate learning into the process.

By employing an additional module the explanation facility, we obtain an elaborate structure of KBS as displayed in Fig. 4. This module has the purpose of explaining to the user the reasoning behind any particular problem solution. In the same way that we ascribe credibility to human experts on the basis of their ability to explain their reasoning, we look an KBS to exhibit the very important quality of transparency, that is, of making their chain of reasoning explicit so that the user may judge its plausibility.

Recalling the architecture of a generic DSS, as depicted in Fig. 1, it should be readily seen that. a DSS may employ heuristics or, in many cases, heuristic programming. By combining KBS and DSS, we obtain an embedded knowledge-based system (EKBS) or hybrid DSS (HDSS) as depicted in Fig. 5. In this way, the combined system (be EKBS or HDSS) would seem to provide a logical extension as well as an enhancement to either the DSS concept or the KBS methodology. It allows for the choice between analytical tools (mathematical models and algorithms) and expert heuristics (knowledge bases and inference strategies), depending on the specific characteristics of the problem at hand.

Recently, there has been growing research interests on applying KBS to almost all engineering disciplines. With the rapid expansion in very-large scale integration (VLSI) technology and the increase in the propagation speed from computer science to engineering, we are witnessing a much wider spread in KBS packages [11]. The major development tools are summarized in Fig. 6 while the classification of knowledge-representation schemes employed by these systems are displayed in Fig. 7.

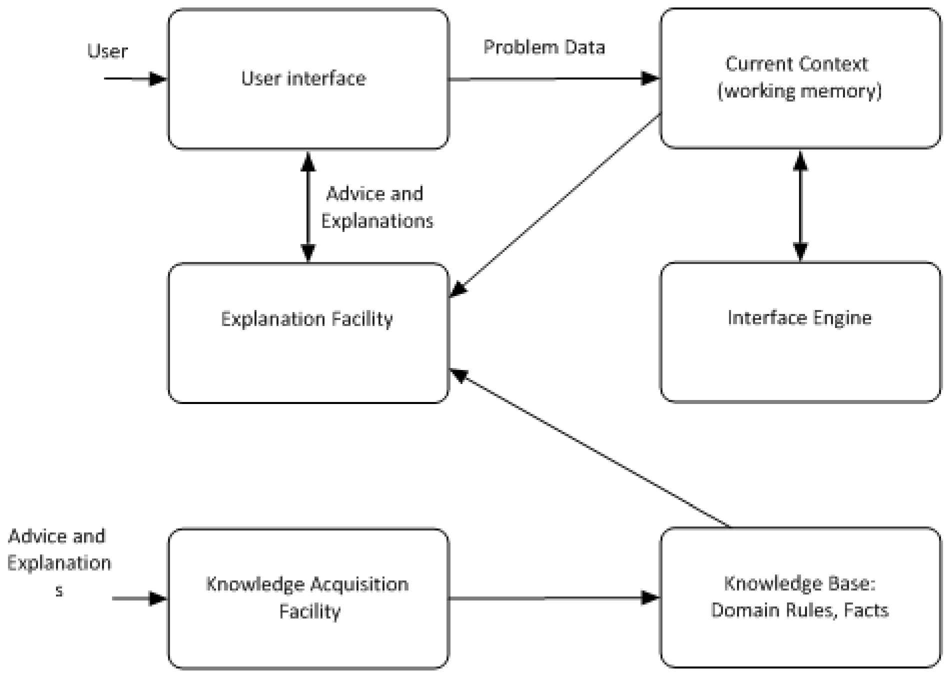


Fig. 4. The Components of a More Elaborate KBS

5. Knowledge-Based Applications

In this section, we introduce two types of knowledge-based applications. First, we provide a short description of a knowledge-based control system which eventually illustrates how to use a KBS methodology as a controller. Second, we highlight ideas on how to use artificially intelligent planning system for control.

5.1. Application 1: A Knowledge Based Control System KBCS

Historically, there are two main approaches to the design and implementation of KBSs. The first approach considers that general problem solvers are often desirable and the second approach. builds specific systems for specific tasks. The synthesis of the two approaches essentially takes the middle ground. The idea is that many tasks have requirements in common, and that these requirements can be met by an expert system shell, to which we add knowledge about particular tasks. Typical shells are EMYCIN [38] and OPS5 [39]. There are variations on this synthesis approach, one of which is to provide a toolkit containing many of the methods used in the various expert system shells [40], [41], [42]. More recently, it appears important that such kits be packaged as open systems; in the sense that the underlying programming language is accessible and the linking of new tools with the ones provided. We note that most of the experimental work in all three phases has been done in AI languages mainly; LISP (LIST Processing) and PROLOG (PROgramming in LOGIc).

A class of KBCSs has been recently developed for the control system analysis and design [43], [44]. One of the basic ingredients of this class of KBCSs is the systematic use of frames. Simply stated, frames are complex data structures for representing stereo-typed objects, events or situations.

A frame has slots for objects and relations that would be appropriate to the situation. Attached. to each frame is information such as

How to use the frame, What to do if something unexpected happens, and Default values for slots. Frames can also include procedural (refers to action) as well declarative (refers to facts and assertions) knowledge.

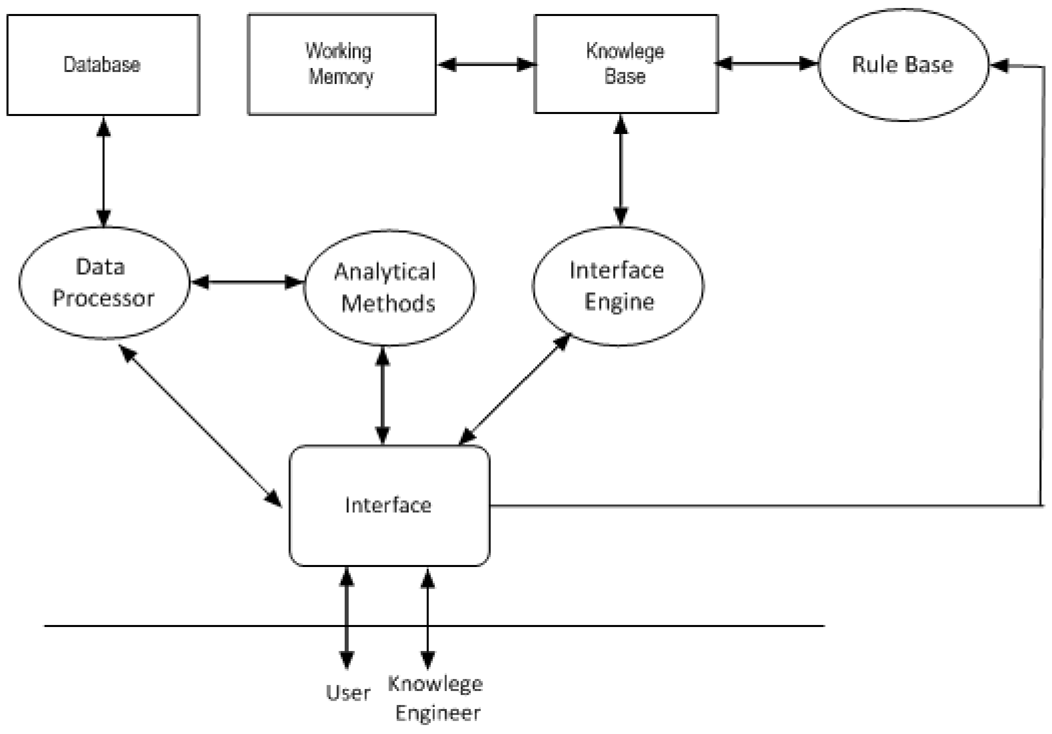


Fig. 5. The Combined DSS and Expert System

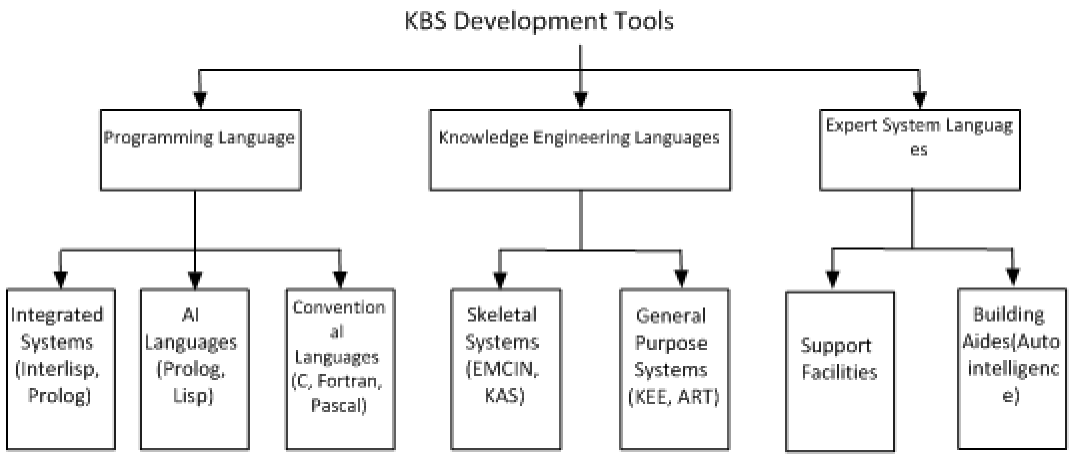

KBS Development Tools
Programming Language
Knowledge Engineering Languages
Expert System Languag es
Integrated Systems (Interlisp, Prolog)
AI Languages (Prolog, Lisp)
Convention al Languages (C, Fortran, Pascal)
Skeletal Systems (EMCIN, KAS)
General Purpose Systems (KEE, ART)
Support Facilities
Building Aides(Auto intelligenc e)

Fig. 6. KBS Development Tools

For example, it can combine (a feedback control systems has a feedback path) as declarative-type information and (.... the next step is to evaluate the eigenvalues of the system matrix.... Then) as procedural-type information. It is important to realize that frames facilitate expectation-driven processing-reasoning based on seeking confirmation of expectations by filling in the slots. More interestingly, frames organize knowledge in a way that directs attention and facilitates recall and

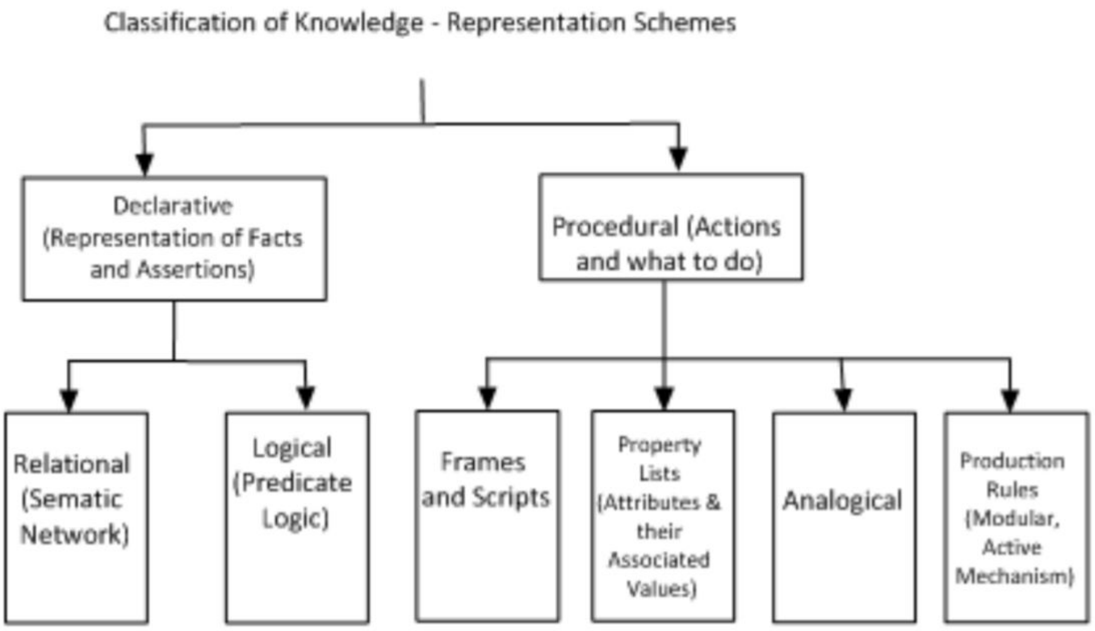

Classification of Knowledge - Representation Schemes
Declarative (Representation of Facts and Assertions)
Procedural (Actions and what to do)
Relational (Sematic Network)
Logical (Predicate Logic)
Frames and Scripts
Property Lists (Attributes & their Associated Values)
Analogical
Production Rules (Modular, Active Mechanism)

Fig. 7. Classification of knowledge-Representation Schemes

inference.

For a typical control system, a generic frame might have knowledge about:

System Types, Stability, Performance, Controllability, Observability along with a procedure associated with each slot to determine values attached to the slot such as System is unstable, System gain margin in db. A candidate example would be

Candidate CONTROL SYSTEN FRAME
Self: A LINKAR_SISO_SYSTEM
A NONLINEAR_S1S0_SYSTEM
Type: TYPE_0_SYSTEM
a TYPE_1_SYSTEM
Stability: a STABLE_SYSTEM
an UNSTABLE_SYSTEM
Stability_Performance: a GAIN_MARGIN
(if_Needed Find a GAIN_MARGIN with value=ok)
Bandwidth_Performance: CLOSED_LOOP_BANDWIDTH
Response_Performance_0: LOW_FREQUENCY_GAIN Response performance, 1:a VELOCITY_CONSTANT

TYPICAL CONTROL_SYSTEM_FRAME
Self: a LINEAR_SISO_SYSTEM
Type: TYPE_0
Stability: a STABLE_SYSTEM
Stability_Performance:-
3db<=EXCESS_GM_VALUE<=3DB
Bandwidth_Performance:-
10%<=EXCESS_BW_VALUE<=10%
Response_Performance_0:-3bb<=-
3db<=EXCESS_LFG_VALUE<=3db
Response_Performance_1:does_not_apply

The second ingredient is the use of production rules (PRs) in formatting the available knowledge into modular procedures. The PRs are characterized by a format of the type:

Pattern - Action
IF - Then
Recognize - Act
Antecedent - Consequent
Situation - Procedure

5.2. Examples

1. IF

System and input matrices are controllable pair AND
System and output matrices are stabilizable pair AND
penalty weighting matrices are positive-definite

THEN

A positive-Definite Riccati Matrix for Optimal State.
Feedback Exists.

2. IF

Closed-Loop-Bandwidth in OK and
Gain-Margin is OK and
Low-Frequency-Gain is OK

THEN

Design is OK.

which is used to detect completion of the lead-lag compensator design sequences

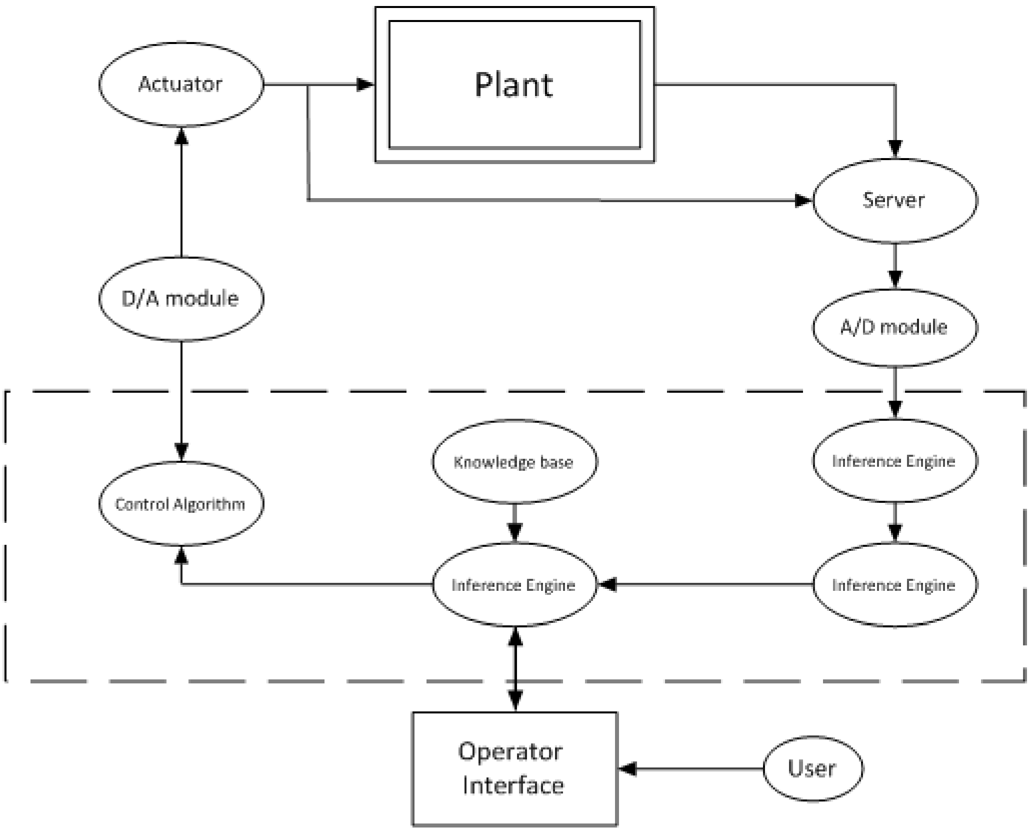


Fig. 8. Structure of Knowledge-Based Control Systems (KBCS)

Because of their modular representation of knowledge and their easy expansion and modifiability, PR are now probably the most popular AI knowledge representation, being chosen for most KB systems [45]. Among the desirable objectives of knowledge-based control systems are

1. the capability of making decisions,
2. the flexibility of producing corrective actions, and
3. the possibility of possessing a certain degree of intelligence.

A candidate structure of KBS is displayed in Fig. 8, from which we observe that

1. the overall system is operated in a closed loop form,
2. the loop is closed through the user (human operator),
3. the system dialogues with the human operator interactively for the necessary data input in order to make successive reasoning,
4. the system is not an autonomous one, however, it is only capable of performing rapid reasoning or decision makings based on the operator supplied data,
5. The core of KBCS contains that of KBS plus the numerical processing and control algorithm units. These units interface with the plant under control via ADA (Digital to Analogue D/A and Analogue to Digital A/D) converters,
6. The system should be capable of monitoring the operations of the plant and the controller,
7. Appropriate control algorithms best suitable for particular situations could be easily selected, and inserted, and
8. The possible system components failures or malfunctioning could be readily detected and, if necessary, some modification or substitution could be made to retain normal operation.

In this way the KBS is capable of providing an expert-like solution by a non-expert human being operator if the correct data are provided.

The functioning of KBCS should be smooth and reliable. More importantly, the closed-loop KBCS could be conceptualized in the form of a three-layer hierarchy shown in Fig. 9, for which we record the following:

1. at the top layer, all decision-makings, appropriate control algorithm selections and suitable diagnostic searches for the possible system component failures. This layer accommodates the fundamental human expertise and knowledge about overall system operation. From this perspective, this layer replicates the technical management experts which, strictly speaking, qualifies the posed KBCS as expert control system.
2. at the bottom layer, basic algorithms of control design are implemented. These include feedback loops, feed-forward paths, gain schedulers, auto- tuners and local controllers to meet design specifications. It is important to note that switching among different control algorithms is possible under the supervision of the middle and top layers, to face up abnormal situations.
3. at the middle layer, the main task is that of adaptation and on-line information acquisition, and hence it constitutes a crucial portion of the decision-making process. This layer draws heavily on the use of measured system variables from the sensor outputs directly and on the estimated system states and parameters. Aspects of information compression and/or preprocessing should be considered, to yield manageable knowledge-base. Here a compromise should be made between knowledge-base manageability, reduced decision periods and time needed for information processing in order to arrive at a meaningful system solution in minimum time; otherwise the issue of real-time KBCS does not hold any more. A final point regards the possible implementation of the KBCS, which concerns the selection of appropriate tools for each building block. Some guidelines to follows are:
   (a) For the inference engine, a good start would be to use either PROLOG or FRANZLISP language.
   (b) In the knowledge representation, it is advisable to rely on the production rules and/or frames.
   (c) Routines of numerical analysis and processing are recommended to come from standard packages like IMSL, MATLAB, KEDD C, MATRIXX.

5.3. Application 2: Artificially Intelligent Planning System (AIPS)

Artificially intelligent planning systems (AIPSs) are typically computer software programs that emulate the way experts plan. They have been used in path planning and high-level decisions about control tasks for robots. A generic planning system can be configured in the architecture of a standard control system, as shown below. Here, the problem domain, corresponds to the plant in control system terminology, defines the

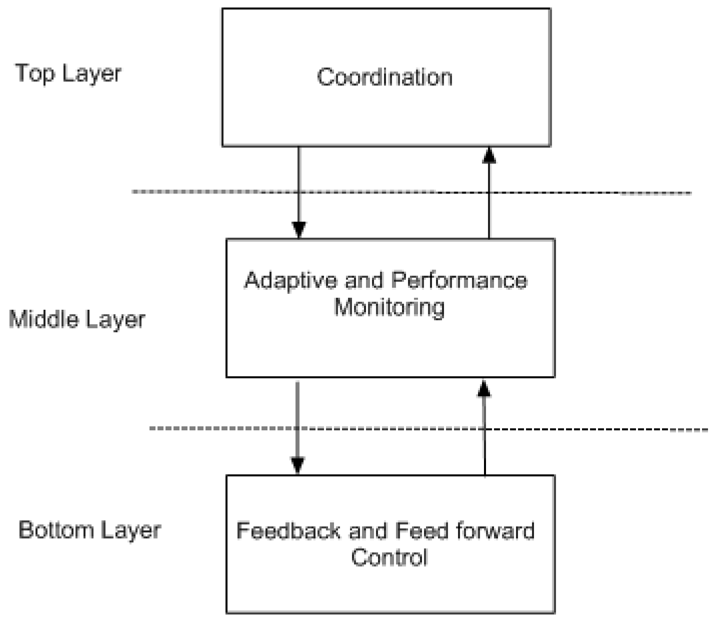
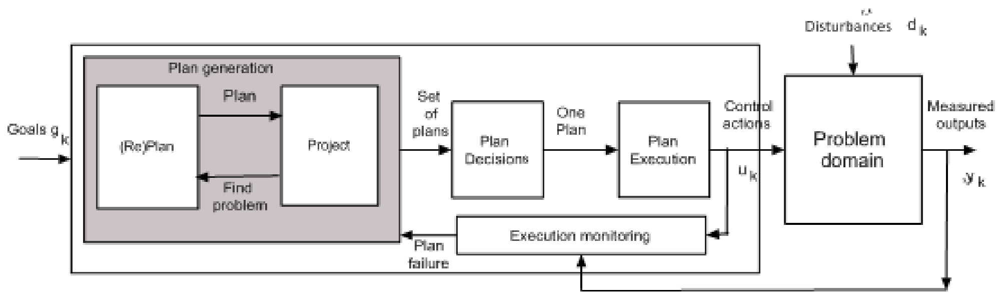

Top Layer
Coordination
Middle Layer
Adaptive and Performance Monitoring
Bottom Layer
Feedback and Feed forward Control

Fig. 9. A Three-Layer Hierarchy

environment that the planner operates in. There are measured output variable y at discrete interval k (variables of the problem. domain that can be sensed in real time), control actions uk (the ways in which we can affect the problem domain), disturbance dk (which represent random events that can affect the problem domain and hence the measured variable yk, and goals gk (what we would like to achieve in the problem domain). There are closed-loop specifications that quantify performance specifications and stability requirements.



It is the task of the planner to monitor the measured outputs and goals and generate control actions that will counteract the effects of the disturbances and result in the goals and the closed loop specifications being achieved. To do this, the planner performs plan generation where it projects into the future (usually a finite number of steps, and often used a model of the problem domain) and tries to determine a set of candidate plans. Next, this set of plans is pruned to one plan that is the best one to apply at the current time (where best can be determined based on, for instance, consumption of resources). The plan is then executed, and during execution the performance resulting from the plan monitored and evaluated. Often, due to disturbance, plan will fail, and hence the planner must generate a new set of candidate plans, select one, then execute that one.

Some planning systems use situation assessment to try to estimate the state of the problem domain (this can be useful in execution monitoring and plan generation); others perform world modeling where a model of the problem domain is developed in an on-line fashion (similarly to on-line system identification), and planner design uses information from the world modeler to tune the planner (so that it makes the right plans for the current domain problem).

5.4. Application 3: AI for VLSI Systems

5.4.1. VLSI Design and verification

A wide variety of algorithmic CAD tools at virtually every abstraction level of the design hierarchy has been developed. By definition, an algorithm involves a fixed sequence of instructions leading to a solution (if one exists). In contrast, a characteristic feature of the real-world chip design is that virtually an infinite number of solutions exist in the solution space. From these solutions, which satisfy given criteria within certain constraints, a number of trade-offs are made leading to an appropriate solution, which is acceptable in terms of operating speed, dissipation, and cost.

The human designer uses heuristics, which tell when, and how to use them in order to avoid expensive processing. A design often involves an interactive cycle of: design description, synthesis, evaluation of the outcome of the synthesis, modification of the design specification or circuit, synthesis. This sequence of decisions to be made in several phases of the design cycle lends itself very well to a knowledge-based approach of VLSI circuit design [53], [55], [55].

The purpose of high-level synthesis is to translate a behavioral input description into an output description in terms of an interconnected network of components. The input description, which may take the form of a high-level program, contains variables and operations on these variables. An operation, e.g., an addition, is assigned to a component, which could perform this operation (e.g., an adder or an ALU). Variables are used to transfer the data between operations and are assigned to data buses and registers. The order of execution is governed by the synthesis problem to be solved.

An intelligent choice from alternative solutions in intermediate steps can be made by an intelligent controller, called Design Critic [53]. The Design Critic can be considered as an expert system whose knowledge base contains rules which allows responsible decisions to be made. A scheme for an intelligent high-level synthesis system is given in Fig. 10. The whole synthesis procedure consists of three main steps:

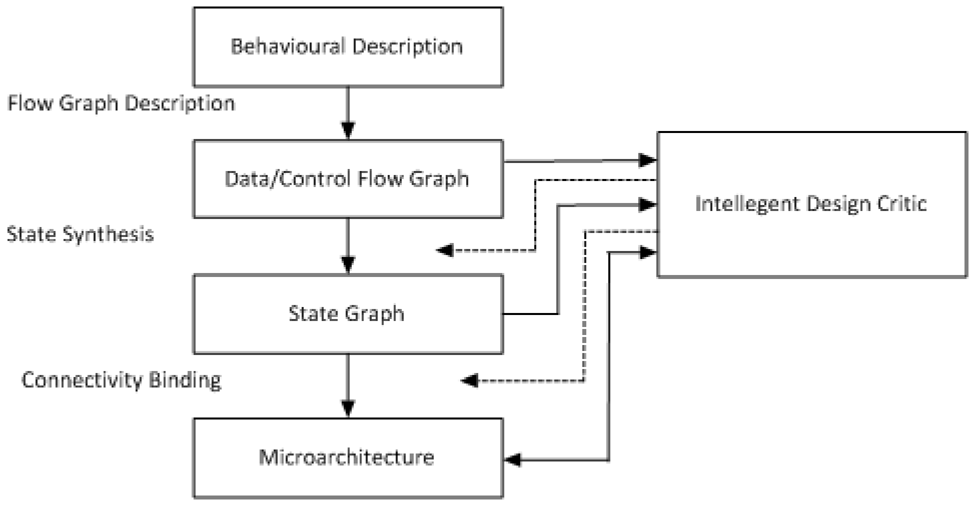


Fig. 10. Intelligent High-level Synthesis

Procedure consists of three main steps:

1. From input specification to flow-graph,
2. From flow graph to state graph, and
3. From state graph to microarchitecture.

A satisfactory degree of optimization can be achieved by using an intelligent decision approach, which involves intelligent choices from alternatives in subsequent steps of the synthesis procedure. In such a step, alternative design options are analyzed so that the best one can be selected. Brewer and Gajski have referred to this procedure as the ”knobs and gauges” method. An evaluator examines several measures of design quality (”gauges”) of various design alternatives. A planner uses these measures to select the appropriate design style anti strategies (”knobs”). The ”knobs and gauges” method is an iterative procedure embracing the steps planning (envisioning possible design styles and strategies), refining (the structure), optimization (improving the quality of the structure) and evaluation (of the design quality). Envisioning is a forward running tree process of pruning unacceptable branches. Measures of design quality include such design criteria as performance, cost, power consumption, operating speed, chip area, the required operators and their frequency of use.

5.4.2. Fault Diagnosis

Fault diagnosis of VLSI system deals with fault localization and fault identification based on the observed system behavior. Fault diagnosis in expert systems involves two sets of concepts hypotheses, which may be possible causes of the malfunction, and a set of deviant observations, which indicate the presence of the malfunction. Typically, a diagnostic problem starts with the observation of a malfunction, which is a deviation of the expected behavior. One strategy is to invoke one or more hypotheses, which may represent the cause of the observed malfunction. Knowledge pieces, or experiments relating malfunction observations to possible causes can be stored in a rule base. This cause-effect mapping can be used to find the causes when the malfunction observations are specified [56]. Going from behavioral observations to identification of the associated causes is a backward matching process. The opposite process is a forward matching. This strategy does not require a detailed knowledge of the internal structure of a system. Such a model is referred to as a shallow knowledge model. When the knowledge of the system implies detailed low-level information on the structure and behavior of the system. [57], [58], the model is referred to as a deep knowledge model. A deep knowledge model reasons from ”first principles” i.e. from understanding of the structure (interconnections of modules at several hierarchy levels) and behaviour (input/output relationships of the modules) of the circuit components. Deep knowledge penetrates to low levels of abstractions whereas a shallow model is concerned with a high-level black-box behaviour.

5.4.3. Test Generation

The objective of test generation is to find a test set with the largest possible fault coverage, i.e. the percentage of the specified fault detected by the test set. Most test generation approaches are based on path sensitizing. It is assumed that a pre-specified stuck-at-0 or stuck-at-1 fault is present at a certain node of a logic network. We are dealing with path sensitizing, when a given test produces opposite logic values for the faulty and fault-free network along a path, the sensitized path to a primary output of the network. In order to detect a fault s-a-d d = 0 or 1 the input vector (the test) must be chosen such that the signals this node in the fault-free network takes on the value $\bar{d}$. A test generation method with path sensitizing consists of three steps:

1. The postulation of a specific fault s-a-d and the assignment of the complementary value to the node at which the fault is present.
2. The construction of a sensitized path, so that the effect of that fault is propagated to the primary output.
3. Utilizing the results in 1 and 2, backtracking from the primary output to the primary inputs of the network is performed in order to assign values to nodes, which have no values as yet, without causing inconsistencies.

The algorithmic approaches are memoryless i.e. knowledge obtained during one iteration of test generation is discarded after this iteration finishes. Furthermore, although most algorithms use heuristics to guide the test process, these heuristics are all hard-wired into the program and hence the augmentation of new heuristic rules requires the modification and recompilation of the programs. Also such algorithms are known to be polynomial complex.

Human test programmers have the tendency to view the circuit at a high macroscopic level and thus attempt to incorporate more global aspects into testing systems. Therefore, many researchers have proposed using knowledge based technology to guide the testing process and several test generation systems using a knowledge-based technology have been implemented. An example to a knowledge-based testing system is depicted in Fig. 11. A typical knowledge-based test system permits the use of both human knowledge and algorithmic procedures [59], [60]. Knowledge is stored using frames and slots, allowing a hierarchical representation. Standard device and circuit models and previously compiled subcircuits are stored in a library. The waveform language is used for specifying both the stimuli applied to the circuit and the expected responses. A standard visual display unit is used for monitoring the logic analysis results.

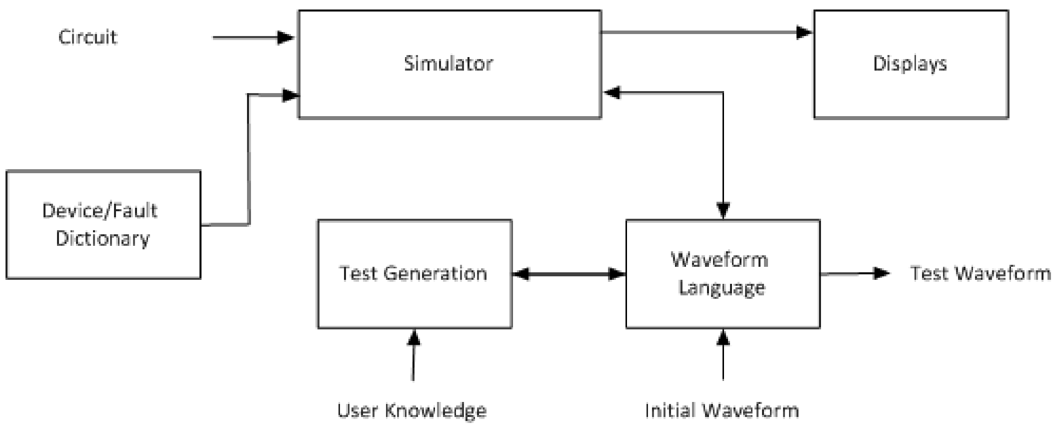


Fig. 11. Example knowledge-based test system

6. Artificial Neural Networks (ANNs)

6.1. Origin and Key Characteristics

Historically, it is known [12] that decision (control), information and neural science were once regarded as a common subject under the banner of Cybernetics. Since that time, the disciplines of decision (control), computing science (including AI) and neurobiology have tended to go their own separate ways. This eventually has led to an unfortunate breakdown in communication between the disciplines; in particular the differing jargon and notation now poses a barrier to effective interchange of ideas. In recent years there has been an increasing interest in studying the mechanism and structure of the brain. This has led to the development of new computational models, based on this biological background, for solving complex problems including pattern recognition, fast information processing and adaptation.

Thus, artificial neural networks (ANNs) are developed to mimic the flexibility and power of the human brain by artificial means. An ANN typically consists of a mesh of nodes and branches connected together, as shown in Fig. 12. The main processing element of every ANN is an artificial neuron, or simply a neuron, or a node as shown in Fig. 12. One early model of a simple ANN was called a perceptron by Rosenblatt [13]. In Fig. 13, a multilayer perceptron along with a single neuron is displayed. Associated with each perceptron are $(n + 1)$ input branches with n unknown weights (sometimes called synaptic weights), a threshold (or a bias weight) input wo, and an activation (nonlinear) function such as a hard limiter or a so-called sigmoid function. Thus, the output of the perceptron represents a nonlinear transformation of the input product of a weighting vector

$$w = [w_0 \;\; w_1 \;\; w_2 \;\; \ldots\ldots\ldots\ldots\ldots w_{\mathrm{n}} \;]$$

and the input vector

$$x = [x_0 \;\; x_1 \;\; x_2 \;\; \ldots\ldots\ldots\ldots\ldots x_{\mathrm{n}} \;]$$

such that

$$u = f\left(y\right), \quad y = w^t * x$$

Note in the foregoing model that the threshold input has been incorporated as a known input of value one [14].

Before describing the architectures and learning aspects of ANNs, we provide below some their key characteristics:

1. Due to their theoretical ability to approximate arbitrary nonlinear mappings, ANNs have great promise in the realm of nonlinear systems problems.
2. ANNs have a highly parallel structure which lends itself immediately to parallel implementation. Such an implementation can be expected to achieve a higher degree of fault tolerance than conventional

schemes. The basic processing element in a neural network has a very simple structure. This, in conjunction with parallel implementation, results in very fast overall processing.

3. ANNs can be readily constructed using VLSI hardware. This brings additional speed and increases the scale of networks which can be implemented in typical applications.
4. ANNs are trained using past data records from the system under study. A suitably trained network then has the ability to generalize when presented with inputs not appearing in the training data. ANNs can also be adapted on-line.
5. ANNs can operate simultaneously on both quantitative and qualitative data. In this respect ANNs stand somewhere in the middle ground between traditional engineering systems (quantitative data) and processing techniques from the artificial intelligence field (symbolic data).

It is clear that a modeling paradigm which has all of the foregoing features has great promise. A broad overview of the field of neural networks and their applications can be found in [15], [16], [17], [18].

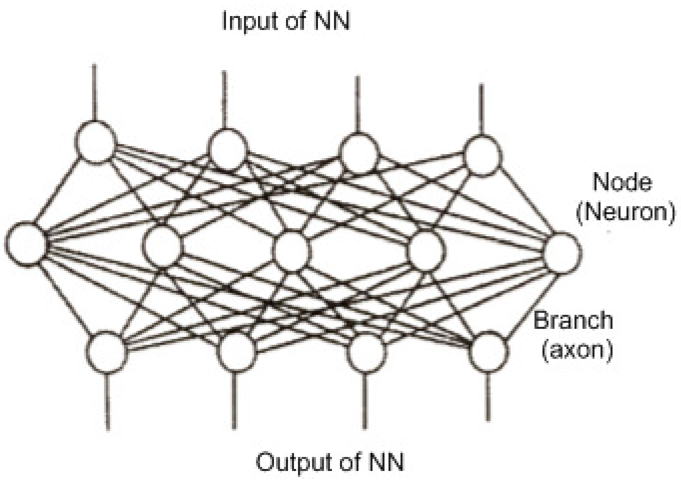


Fig. 12. A Typical Neural Network

6.2. Network Architectures

The network architecture is defined by the basic processing elements and the way they are interconnected. An elaboration of the perceptron model of Fig. 13 yields the ANN architecture depicted in Fig. 14 which provides a unifying standard model for several engineering applications. The basic processing elements are:

A weighted summer described by

$$v_j\,(t) = \sum_{k=1}^{N} a_{jk} y_k\,(t) + \sum_{r=1}^{N} b_{jr} u_r\,(t) + w_j \tag{1}$$

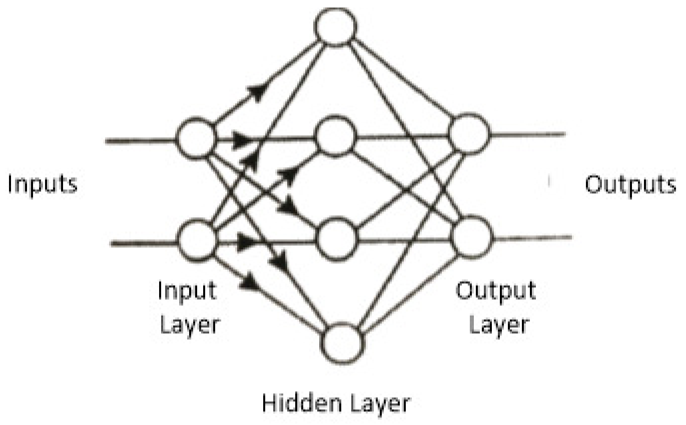
which gives a weighted sum v_j, in terms of the outputs of all elements y_k, external inputs u_r, and corresponding weights a_{jk}, and b_{jr}, together with constants w_j. By stacking N weighted sums v_j into a column vector v, the N outputs y_k into a vector y and M inputs u_r, into a vector u and the N constants w_j, into a vector v we obtain:

$$v\,(t) = \mathrm{A\ y\ (t)} + \mathrm{B\ u}\,(t)\ \ + \mathrm{w} \tag{2}$$

The linear dynamic system has input v_j, and output x_j which by

$$X_j\,(s) = H\,(s)\,V_j\,(s)\ \Leftrightarrow x_j\,(t) = \int_{-\infty}^{t} h\,(t-\tau)\,v_j\,(\tau)\,d\tau \tag{3}$$

where, $\{x_j(t),\ X_j(s)\ \}$, $\{v_j(t),\ V_j(s)\ \}$ and $(h(t),\ H(s))$ are Laplace transform pairs. The non-dynamic nonlinear function $g(.)$ gives the element output y_j in terms of the transfer function output x_j:



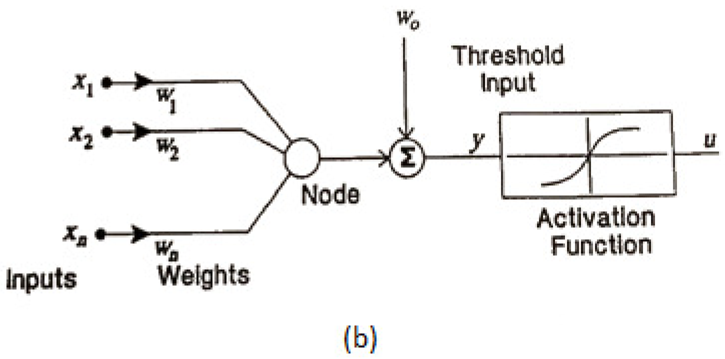


Fig. 13. a) A Three-Layer Perceptron, b) A Perceptron Model

$$y_i = g(x_j) \tag{4}$$

Some of the frequently used nonlinear functions $g(x)$ are shown in Table 2. Each representation of the basic unit has particular advantages which makes it suitable for typical applications [19], [20].

6.3. Connections

Strictly speaking, the neurons by themselves are not very powerful in terms of computation or representation but their interconnection allows us to encode relations between the variables giving different powerful processing capabilities. The three components of the neuron model of Fig. 14 can be combined in various ways. For example, if the neurons are all non-dynamic $(H(s) \equiv 1)$, then an assembly of neurons can be written as the set of algebraic equations in the form:

$$\begin{aligned} x(t) &== Ay(t) + Bu(t) + w \\ y(t) &= g(x(t)) \end{aligned} \tag{5}$$

If, on the other hand, each neuron has first order low-pass dynamics $H(s) = (1+Ts)^{-1}$, then an assembly of neurons can be cast into the form of ordinary differential equations:

$$\begin{aligned} \mathrm{T}\dot{\mathrm{x}}(\mathrm{t}) + \mathrm{x}(\mathrm{t}) &= \mathrm{Ay}(\mathrm{t}) + \mathrm{Bu}(\mathrm{t}) + \mathrm{w} \\ y(t) &= g(x(t)) \end{aligned} \tag{6}$$

6.4. Static Multi-layer Feedforward Networks

The connection of several layers gives the possibility of more complex nonlinear mapping between the inputs and outputs. This capability can be used to implement classifiers or to represent complex nonlinear

Table 2. Nonlinear Functions g(x)

Name	Formula	Characteristics
Threshold	+1 if x $>$ 0 else 0	Non-differentiable, step-like, positive
Threshold	+1 if x $>$ 0 else -1	Non-differentiable, step-like, zero-mean
Sigmoid	$(1+e^{-x})^{-1}$	Differentiable, step-like, positive
Hyperbolic tan-gent	tanh(x)	Differentiable, step-like, zero- mean
Gaussian	$e^{\frac{-x^2}{\sigma^2}}$	Differentiable, pulse-like

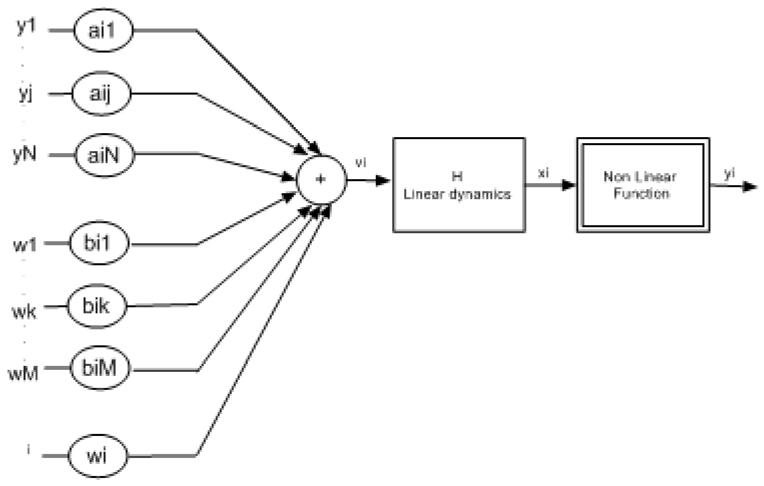


Fig. 14. Basic Model of a Neuron

relations among the variables. Such networks are typically non-dynamic $(H(s) \equiv 1)$. The connection matrix A is such that the outputs are partitioned into layers so that a neuron in one layer receives inputs only from neurons in the previous layer. There is no feedback in such networks. Typically, in a three-layer network with each layer containing N neurons, we get the model:

$$\begin{bmatrix} x^1(t) \\ x^2(t) \\ x^3(t) \end{bmatrix} = A \begin{bmatrix} y^1(t) \\ y^2(t) \\ y^3(t) \end{bmatrix} + B \begin{bmatrix} u^1(t) \\ u^2(t) \\ u^3(t) \end{bmatrix} + \begin{bmatrix} w^1(t) \\ w^2(t) \\ w^3(t) \end{bmatrix} \quad (7)$$

$$A = \begin{bmatrix} 0 & 0 & 0 \\ A^2 & 0 & 0 \\ 0 & A^3 & 0 \end{bmatrix}, B = \begin{bmatrix} B^1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (8)$$

Where, the indicated zeros are matrices of zeros with appropriate dimensions. Different characteristics are readily obtained using different nonlinearities $g(.)$ from Table 2.

6.5. Dynamic Networks

It is simple to observe that the introduction of feedback produces a dynamic ANN with several stable points. The general equation can be expressed as

$$\dot{x}\,(t) = F\,[x\,(t)\,,\; u\,(t)\,, \theta]$$

$$y\left(t\right)=G\left[x\left(t\right),\theta\right] \tag{9}$$

where *x*represents the network state, *u* is the external inputs, and stands for the parameter of the network. *F* and *G* are functions representing the network structure and the state-output relations, respectively. Originally, feedback (recurrent) networks were introduced in the context of associative or content-addressable memory (CAM) problems [21], [22] for pattern recognition. The best-known example of a CAM is the Hopfield net described by:

$$T_j\,\dot{x}\left(t\right)=-x_j\left(t\right)+\sum_{k=1}^{N}a_{jk}\,y_k\left(t\right)+u_j$$

$$y_j\left(t\right)=g_j[x_j\left(t\right)],\ \ \mathrm{j}\ =\ 1,\ldots,\ \mathrm{N} \tag{10}$$

where, g_j (.) are sigmoid functions.

Indeed, there exist numerous ANN architectures [23], [24], [25], [26], [27] and many of them are refinement/extensions to suit particular applications.

6.6. Learning Process

The application of ANNs normally proceeds in two phases. In the first (recall or training) phase, the network is trained using suitable data. A trained network with constant coefficients then provides a fixed behavior. The second (learning) phase concerns the automatic adjustment (adaptation) of the network coefficients. Learning algorithms can be broadly classified into three main groups:

1. Supervised learning which incorporates an external reference signal (teacher) and/or global information about the system,
2. Unsupervised learning which incorporates no external reference signal and relies only upon local information and internal signals, and
3. Structural learning which induces some change in the structure of the memory as represented by network weights.

For a single-layer non-dynamic architecture with w = 0 and described by
$x(t)=Bu(t),y(t)=g[x(t)]$
the standard ANN learning problem can be bossed as follows:

1. y and u contain signals available for measurement
2. The function $g[.]$ is specified
3. A desired output (or reference) value r corresponding to each u is known d) Find a parameter estimate B such that the square of the error $e(t)=r(t)-y(t)$ is minimized over all pairs y,u.

The main effort is centered about the dynamic characterization of B which eventually leads to different algorithms [29], [30]. Back propagation (BP), a learning algorithm common in the ANN literature, is a special case of this general problem and can be viewed as a gradient algorithm applied to a nonlinear optimization [29].

In dynamic or recurrent networks, the situation is qualitatively different due to the presence of feedback. One of the concepts employed in this regard is the fixed-point learning which aims at making the ANN reach the prescribed equilibria or perform steady-state matching by forcing the transients to die out. Another concept is the trajectory learning by training (adapting) the ANN to follow the desired trajectories in time and as time increases, it will eventually reach the prescribed steady-state.

The literature is abound with algorithms and learning methods, see [33], for an excellent survey. Machine Learning (ML) and has become relatively competitive to conventional regression and statistical models regarding usefulness [32]. A review of ML algorithms is given in [26]. Hinton et al. [31] introduced the concept of Deep Learning (DL) by utilizing ANNs for dimensionality reduction. ANNs became more powerful and complex, and literally deeper with many layers and neurons, the ability for deep learning to facilitate robust machine learning and produce AI increased. The main aim of using deep learning is the use of GPU (Graphics Processing Unit) hardware, data dependencies, and feature engineering. Data dependencies means which works with a large amount of data. DL provides automatic learning of features and their representation in a hierarchical manner at various levels. This powerful process of DL makes it robust in contradiction of traditional machine learning methods, in short, deep learning complete architecture is used for feature extraction and alteration process [27]. The initial layers perform simple processing of input data or learn the easy features and that output goes to the upper layers, which performs complex features learning. Therefore, deep learning is suitable for dealing with larger data and complexity [28].

7. Application 4: System Identification

It has been repeatedly affirmed that ANNs have great promise in the modeling of nonlinear systems. Without reference to any particular network structure we now discuss architectures for training networks to represent nonlinear dynamical systems and their inverses. An important question in system identification is that of identifiability [34], that is, given a particular model structure, can the system under study be adequately represented within system structure ?.

An answer to the posed question will be based on the assumption that all systems we are likely to study belong to the class of systems that the chosen network is able to represent. One procedure for training an ANN to represent the forward dynamics of a system (or forward modeling) is schematically illustrated by the structure of Fig. 15. In this structure, the neural network model is placed in parallel with the system and the error between the system and the network outputs (the prediction error) is used as the ANN training signal. We note that this learning structure resembles the classical supervised learning problem where the teacher (system) provides target values (its outputs) directly in the output coordinate system of the learner (the network model). In the particular case of a multilayer perceptron type network, straightforward back propagation of the prediction error through the ANN would eventually provide a possible training algorithm.

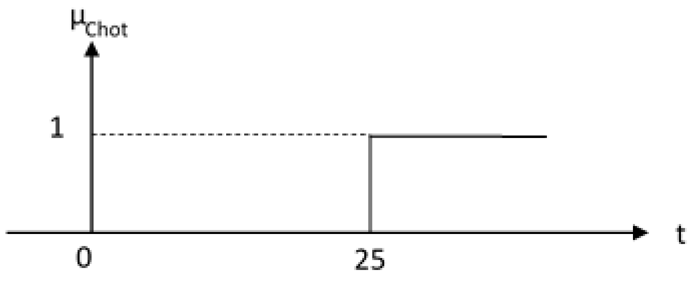
It should be observed that by either using recurrent networks or by introducing dynamic behavior into the neurons [35], one can directly deal with identification of dynamical systems. Other related results can be found in [36], [37].

8. Fuzzy Logic and Systems (FLSs)

Among the new technologies based on AI, fuzzy logic is now perhaps the most popular area, judging by billions of dollars worth of sales and close to 2,000 patents issued in Japan alone since the announcement of the first fuzzy chips in 1987. In many consumer products like washing machines and cameras, fuzzy logic controllers are used in order to obtain high machines IQ and user-friendly products. Other interesting applications include control of subway systems, image stabilization of video cameras, and autonomous control of helicopters. In this section, we briefly introduce fuzzy set theory and fuzzy logic. Then, we move to present fuzzy control systems.

8.1. Fuzzy Set Theory: A Brief Review

In 1965, Zadeh [9] wrote a seminal paper in which he introduced **fuzzy sets**, sets with *unsharp* boundaries. These sets are generally in better agreement with the human mind that works with shades of grey, rather than with just black or white. Fuzzy sets are typically able to represent linguistic terms, that is, *warm, hot, high, low*. Nearly 10 years later Mamdani [46] succeeded to apply fuzzy logic for control in practice.



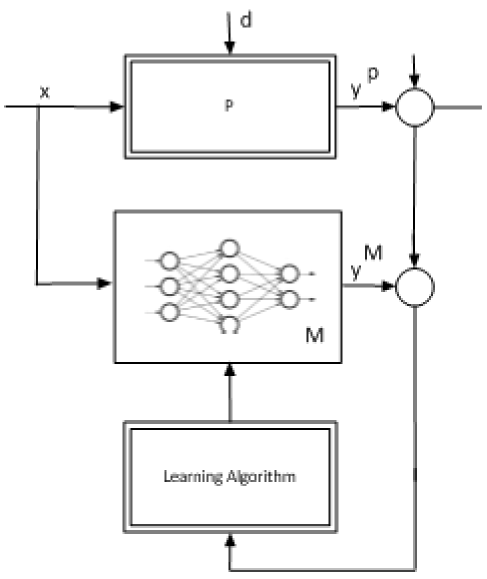
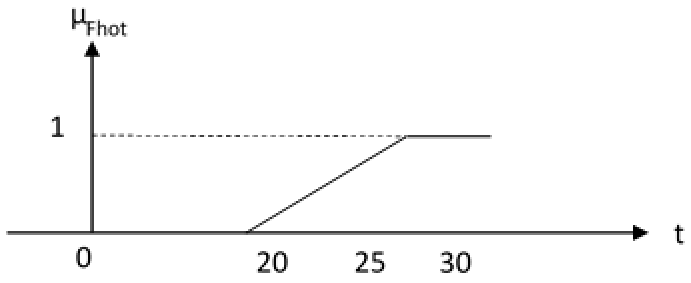


Fig. 15. Identification

In classical set theory, a set is denoted as a so-called crisp set and can be described by its characteristic function as follows

$$\mu_c : u \rightarrow \{0, 1\} \tag{11}$$

In (11) u is called the *universe of discourse*, that is, a collection of elements that can be continuous or discrete. In a crisp set each element of the universe of discourse either belongs to the crisp set (μ_c =1) or does not belong to the crisp set (μ_c = 0). Consider a characteristic function μ_{chot} of representing the crisp set *hot*, a set with all hot temperatures. As it is shown below, it considers temperatures higher than 25 degrees Celsius as hot. Note that for all the temperatures t we have $t \in u$.



According to [9], a *fuzzy* set is given by the characteristic function

$$\mu_f : u \rightarrow [0, 1] \tag{12}$$

In this case, the elements of the universe of discourse can belong to the fuzzy set with any value between 0 and 1. This value is called the *degree of membership*. If an element has a value close to 1, the degree of membership, or truth value is high. The characteristic function of a fuzzy set is called the *membership function*, for it gives the degree of membership for each element of the universe of discourse. If now the characteristic function of μ_{Fh} is considered, one can express the human opinion, for example, that 24 degrees is still fairly hot, and that 26 degrees is hot, but not as hot as 30 degrees and higher. This results in a gradual transition from membership (completely true) to non membership (not true at all). The membership function μ_{Fh} of the fuzzy set F_{hot} is shown below as a linear transition.

Note however that every individual can construct a different transition according to his own. Membership functions can have many possible shapes. In short, a fuzzy set constitutes a mathematical expression for the lack of precision in a quantitative fashion.

8.2. Fuzzy Logic

As discussed earlier, fuzzy logic stems from the notion of fuzzy sets. As in the traditional crisp sets, logical operations, that is, union, intersection and complement, can be applied to fuzzy sets [9]. Based on these operators, fuzzy relations and principles can be constructed [47] as universal mappings. According to [47], *predicate logic* and *approximate reasoning* form the basis for fuzzy calculus and mathematics. A predicate logic proposition is a linguistic statement contained with a universe of propositions which are either completely *true* or *false*. We can combine simple propositions using logical connectives to form new or compound propositions, and so on. By the same token, many ingredients of classical logic can be extended to fuzzy logic including contradictions, tautologies, deductive inferences [47], [48].

Now, any system that incorporates fuzzy logic is called a *fuzzy system*. The primary goal of fuzzy systems is to formulate a theoretical foundation for reasoning about imprecise propositions through fuzzy IF-THEN rules. A typical fuzzy rule has one or more *antecedents* (IF parts) and one or more consequents (THEN part). Consequents from different rules are numerically combined (typically unioned via a MAX operation) and are then collapsed (typically taking the weighted average or anteroid of the combined distribution) to yield a single real number (binary) output.

As an example, consider the following statement:

IF the temperature is Hot and the pressure is Low

THEN increase valve position to Large amount.

Treating Hot, Low, and Large as fuzzy variables (or sets), the forgoing statement might constitute a fuzzy rule or a part of fuzzy rule-based system. Note that fuzzy variables are usually the results of compressing or reducing a partition of numerical values of physical variables like temperature or pressure into linguistic value.

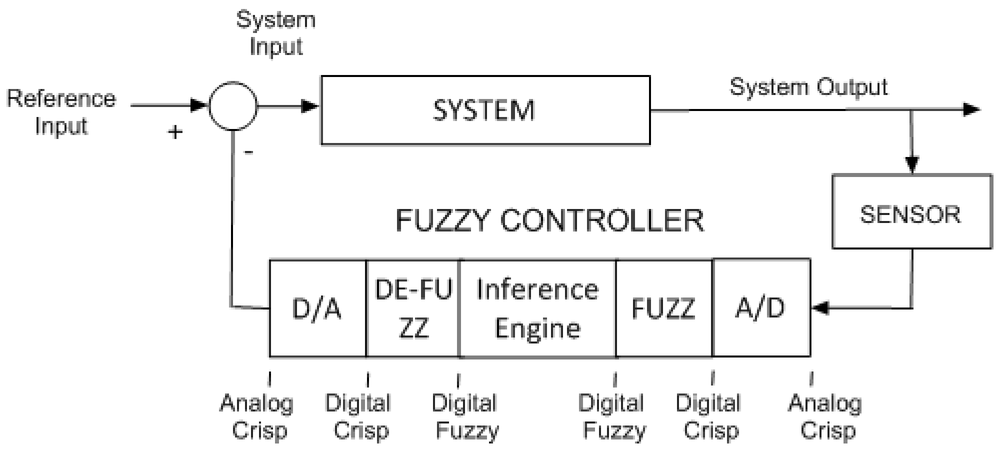


Fig. 16. The Architecture of a Fuzzy Controller

8.3. Fuzzy Control Systems (FCSs)

It becomes a prevailing trend in systems engineering that the process of design and analysis should be based on the best available knowledge instead of the simplest available model to treat the uncertainties in the system. This trend emphasizes the use of any available model as an integral part of the overall knowledge base of the system. The designer should pursue further information on the system and its operational practices to complete the knowledge.

In this section, we introduce a knowledge-based controller design using fuzzy system theory. By contrast to the KBCS discussed in Section 5, this forms another type of intelligent controller which are not solely model-based, but also knowledge-based. From a theoretical point of view, fuzzy logic rule base can be used to identify both a model, as a universal approximation, as well as a nonlinear controller. Bear in mind that the most relevant information about any system comes in one of three ways: a mathematical model, sensory input/output data, and human expert knowledge. In the light of the foregoing discussions, a typical fuzzy control architecture is shown in Fig. 16. We note that the sensory data goes through levels of interface, that is, the usual analog to digital (A/D) and crisp to fuzzy and at the other end in reverse order, that is, funny to crisp and digital to analog

(D/A).

The controller architecture contains essentially three operations:

1) Fuzzification It represents a mapping from a crisp point$x = [x_1\ x_2\ \ldots .x_n]^t \in u$ into a fuzzy set $A\in\ u$ where u is the universe of discourse. There are normally two categories of fuzzifiers is use. They are: **singleton** which has one point (value) x_p, as its fuzzy set support and **nonsingleton** in which the support is more than a point.

2) Inference Engine It is the corestone of the fuzzy controller as it consists of a set of expert rules which reflect the knowledge base and reasoning structure of the solution of any problem. In applications, standard multiterm controllers such as **PI**, **PD**, and **PID** can be easily represented in terms of fuzzy rules and their associated structure.

3) Defuzzification. There are several schemes of defuzzification [45], the most common of which is the center of *gravity defuzzifier*. In this scheme, the weighted average of the membership or center of gravity of the area bounded by the membership function curve is computed as the most typical crisp value of the union of all output furzy sets. Further information can be found in [44], [45], [46].

9. Application 5: Intelligent and Autonomous Systems (IASs)

Autonomous systems have the capability to independently perform complex tasks with a high degree of success. Consumer and governmental demands for such systems are frequently forcing designers to push many functions normally performed by humans into machines. For instance, in the emerging area of intelligent vehicle and highway systems (IVHS), engineers are designing vehicles and highways that can fully automate vehicle route selection, steering, braking and throttle control to reduce congestion and improve safety. In avionic systems a pilot's associate computer program has been performed to emulate the functions of mission and tactical planning that in the past may have been performed by the copilot.

From a historical perspective, such applications began at a low level of automation, and through the years have evolved into a more autonomous system. The general trend has been for engineers to incrementally add more intelligence in response to consumer, industrial and government demands and thereby create systems with increased level of autonomy. In recent years, the emphasis has been on emulating the functionality of an intelligent biological system to solve a technological problem and this is collectively called artificially intelligent systems and control techniques (AISCTs). As discussed earlier, such AISCTs include knowledge-based systems, artificial neural networks, fuzzy logic systems, genetic algorithms.

9.1. Architecture and Characteristics

A functional architecture of an intelligent autonomous controller shown in Fig. 15 where the interface to process involve sensing (via conventional sensing technology, vision, touch, smell and the like), actuation (via hydraulic, robotics, motors and the like) and there is an interface to humans and other systems via a driver, pilot, crew and the like.

The execution level has low-level numeric signal processing and control algorithms. These include PID, optimal, adaptive, or intelligent control; parameter estimators and failure detection and identification (FDI) algorithms. The coordination level provides for tuning, scheduling, supervision, and redesign for the execution-level algorithms, crisis management, planning and learning capabilities for the coordination of execution-level tasks, and higher-level symbolic decision making for FDI and control algorithm management. The management level provides for the supervision of lower-level functions and for managing the interface to the human(s) and other systems. In particular, the management level will interact with the users in generating goals for the controller and in assessing the capabilities of the system. The management level also monitors performance of the lower-level systems, plans activities at the higher level (and in cooperation with humans), and performs high-level learning about the user and the lower-level algorithms. Intelligent systems or intelligent controllers based on fuzzy, neural, genetic, expert or planning can be employed as appropriate in the implementation of various functions at the three levels of the intelligent autonomous controller.

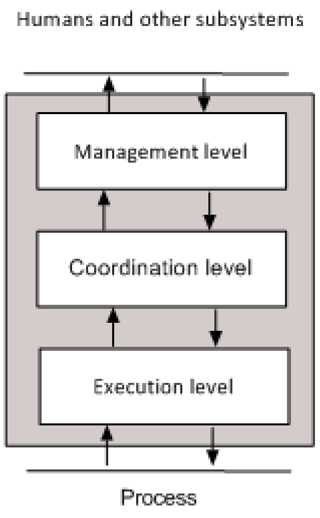


Fig. 17. Intelligent Autonomous Controller

There are several fundamental characteristics that have been identified for intelligent autonomous control systems:

1. There is generally a successive delegation of duties from the higher to lower levels, and the number of distinct tasks typically increases as we go down the hierarchy.
2. Higher levels are often concerned with slower aspects of the system's behavior and with its larger portions, or broader aspects.
3. There is a smaller contextual horizon at lower levels, that is, the control decisions are made by considering less information.
4. At higher levels, there is typically a decrease in time-scale density, a decrease in bandwidth or system rate, and a decrease in the decision (control action) rate.

In brief, there is increasing intelligence with decreasing precision as one moves from the lower to the higher levels.

9.2. Intelligent Vehicle and Highway Systems

In order to make the operation of autonomous systems and the notion of autonomy more concrete, we examine in the sequel an intelligent vehicle and highway systems (IVHS) problem of automating a highway system. One possible general functional architecture for automated high way systems is shown in Fig. 18 for the case where many vehicles operate on a large roadway system in the metropolitan area of a large city.

Execution Level Each vehicle is equipped with

1. vehicle control system that can control the brakes, throttle, and steering to automate the driving task (for normal operation or collision avoidance)
2. vehicle information system in each vehicle that provides information to the driver (like platoon lead vehicle information; information on traffic congestion, road construction, accidents, weathers, road conditions, lodging, and food) and information to the overall system about the vehicle.

   For the roadway there are

3. the traffic signal controllers (about intersections, and ramp metering) and
4. the roadway information systems that provide information to the driver and other subsystems (including automatic signing systems that provide rerouting information in case of congestion, road condition

warning systems, accident information). These four components form the execution level in the intelligent autonomous controller, and clearly, these components will be physically distributed across many vehicles, roadways, and areas of the metropolitan area.

Coordination Level In the coordination level, there is a manager for vehicle control that

1. may coordinate the control of vehicles that are in close proximity to form it platoons, maneuver platoons, and avoid collisions, and
2. provide information about such control activities to the rest of the system.

In addition, there is a manager for vehicle information that

1. makes sure that appropriate vehicles get the correct information about road, travel, and traffic conditions, and
2. manages and distributes the information that comes in from vehicles on accidents and vehicle failures so that the control manager can navigate platoons to avoid collisions. The manager for traffic signal control could

1. utilize information from the roadway information system (for example, on accidents or congestion) to adaptively change the traffic light sequences at several connected intersections to reduce congestion, and
2. provide information to the other subsystems about signal control changes (for example, to the vehicle information systems). The manager for roadway information

1. provides information on road conditions, accidents, and congestion to the other subsystems, and
2. provides information from the other subsystems to the roadway for changeable message signs (that is, rerouting information from the traffic signal control manager).

As indicated in Fig. 18, there are multiple copies of each of the managers and the entire coordination level as needed for different areas in the metropolitan region.

Management Level The management level is the traffic and vehicle management center, which provides for high level management of traffic flow. It provides an interface to their automated highway systems (perhaps in rural areas or other nearby metropolitan areas) and to other authorities (that is, to provides information on police and emergency services on accidents and to input information on construction, weather predictions, and other major events that affect traffic system behaviour. In case of an accident situation, the traffic and vehicle management center would react as quickly as possible to alert emergency vehicles.

The lower levels of the system have a smaller contextual horizon since they consider much less information in making decisions. Also, the decision rate tends to be higher at the lower rate levels in the sense that the rate at which control corrections are made as a vehicle automatically steers around a curve may be on the order of milliseconds, the decisions rate at the management level may be on the order of minutes or hours.

Clearly, the IVHSs call for a significant amount of interdisciplinary activity and wide range of technologies for its implementation. While conventional systems and control technologies will certainly find wide use in IVHS, it seems likely that AISs will prove to be useful for at least some functions, especially considering the focus on automating what has traditionally been largely a human control activity.

10. Relationship between Fuzzy Systems and Neural Networks

There are two ways in which there are relationships between fuzzy systems and neural networks. First, techniques from one area can be used in the other. Second, in some cases the functionality (that is, the nonlinear function that they implement) is identical. Some researchers label the intersection between fuzzy systems and neural networks with the term fuzzy-neural or neuro-fuzzy to highlight that techniques from both fields are being used. Barring this terminology, we highlight in the sequel the basic relationships between the two fields.

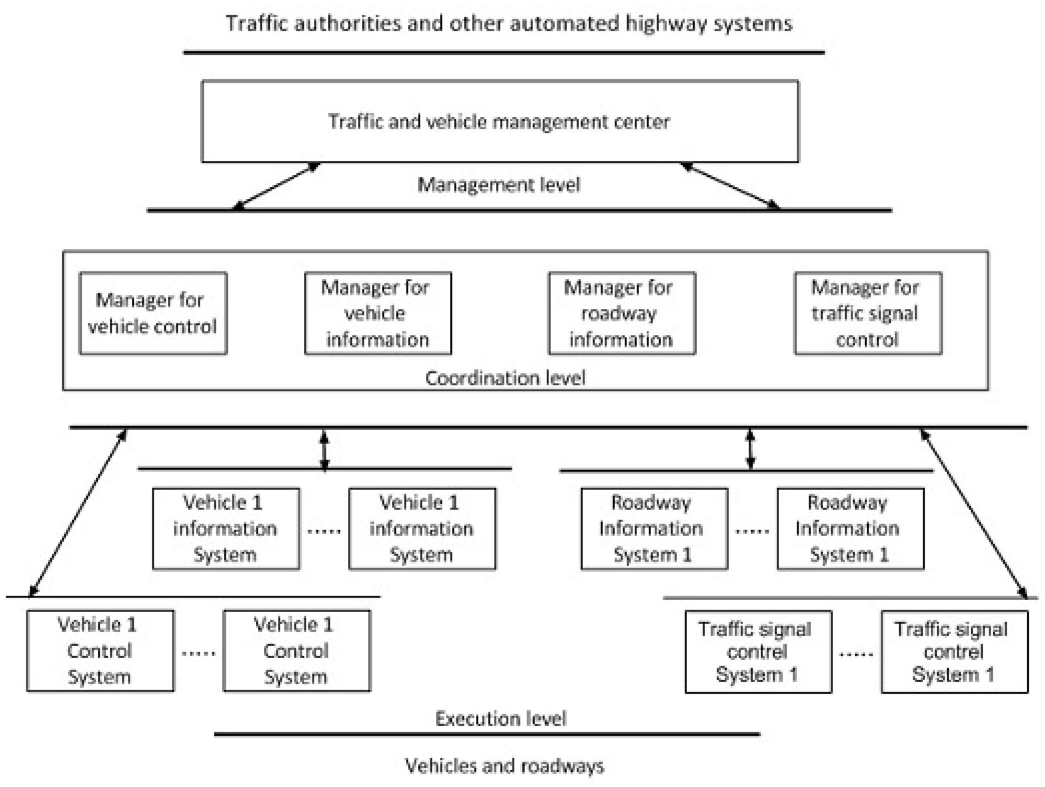

Traffic authorities and other automated highway systems
Traffic and vehicle management center
Management level
Manager for vehicle control
Manager for vehicle information
Manager for roadway information
Manager for traffic signal control
Coordination level
Vehicle 1 information System
Vehicle 1 information System
Roadway Information System 1
Roadway Information System 1
Vehicle 1 Control System
Vehicle 1 Control System
Traffic signal contrel System 1
Traffic signal contrel System 1
Execution level
Vehicles and roadways

Fig. 18. Intelligent Autonomous Controller for an Intelligent Vehicle and Highway System

10.1. Multilayer perceptrons

The multilayer perceptrons should be viewed as a nonlinear network whose nonlinearity can be tuned by changing the weights, biases, and parameters of the activation functions. The fuzzy system is also a tunable nonlinearity whose shape can be changed by tuning, for example, the membership functions. Since both are tunable nonlinearities, the following approaches are possible:

1. Gradient methods can be used for training neural networks to perform system identification or to act as estimators or predictors in the same way as fuzzy systems do.
2. By combining gradient and clustering methods, we obtain hybrid methods for training which can also be used for neural networks.
3. Indirect adaptive and gain scheduled control strategies can also be achieved with a multilayer perceptron by either treating them as the tunable nonlinearities or training them to map the associations between operating conditions and controller parameters.

10.2. Radial Basis Function Neural Networks

Some radial basis function neural networks are equivalent to some standard fuzzy systems that use center-average defuzzification. The meaning is they are functionally equivalent in the sense that given the same inputs they will produce the same output. It is interesting to note that, the functional fuzzy system (in particular, the more general version of the Takagi-Sugeno fuzzy system) is equivalent to a class of two-layer neural networks [47].

11. Conclusions

In this paper, we have provided an overview of the concepts and methodologies of artificially intelligent systems (AISs). To unify the discussion, we have presented some background information about decision making. A brief account of artificial intelligence has been included. Then we have presented three basic methodologies of AISs and outlined their main features. These include knowledge-based systems (KBSs), artificial neural networks (ANNs) and fuzzy logic and systems (FLSs). An engineering application has been given from each methodology to illustrate the concepts, merits and demerits. The relationship between ANNs and FLSs has been noted. Finally, the basic elements of the recently developed intelligent and autonomous systems (IASs) have been presented with particular emphasis on intelligent vehicle and highway systems.

Indeed, the subject of AISs is fascinating and productive. For further information, the reader is referred to [48], [49], [50], [51], [52].

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.
Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.

References

[1] R. Penrose, *The Emperor's New Mind*, Oxford University Press, Oxford, 1989, https://books.google.co.id/books?id=FZduoOiOtyMC.

[2] F. D. Peat, *Artificial Intelligence: How Machines Think*, Simon & Schuster, New York, 1985, https://books.google.co.id/books?id=mX5WAAAAYAAJ.

[3] S. Russell and P. Norvig, *Artificial Intelligence*, Prentice-Hall, New York, 1999.

[4] A. Caswsey, *The Essence of Artificial Intelligence*, Prentice-Hall, New York, 1998, https://books.google.co.id/books?id=EAsGAsSTCE8C.

[5] C. Manning and Hinrich Schutze, *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, MA, May 1999, https://books.google.co.id/books?id=3qnuDwAAQBAJ.

[6] A. Barr and E. A. Feigenbaum (Editors), *The Handbook of Artificial Intelligence: Volume 1, Volume 1*, William Kaufmann Inc., CA, 1981, https://books.google.co.id/books?id=3oviBQAAQBAJ.

[7] G. F. Luger and W. A. Stubblefield, *Artificial Intelligence and The Design of Expert Systems*, Benjamin/Cummings, CA, 1989, https://books.google.co.id/books?id=_IBQAAAAMAAJ.

[8] S. E. Umbaugh, *Computer Vision and Image Processing*, Prentice-Hall, New York, 1998, https://books.google.co.id/books?id=9GChGQAACAAJ.

[9] L. A. Zadeh, "Fuzzy Sets," Information and Control, vol. 8, 1965, pp. 338-353, https://doi.org/10.1016/S0019-9958(65)90241-X.

[10] J. P. Ignizio, *Introduction to Expert Systems: The Development and Implementation of Rule-based Expert Systems*, McGraw-Hill, New York, 1991, https://books.google.co.id/books?id=obJQAAAAMAAJ.

[11] C. L. Dym, and R. E. Levitt, *Knowledge-Based Systems in Engineering*, McGraw Hill, New York, 1991, https://books.google.co.id/books?id=joVRAAAAMAAJ.

[12] N. Wiener, *Cybernetics: or Control and Communication in the Animal and the Machine*, MIT Press, Cambridge, MA, 1948, https://books.google.co.id/books?id=mP1XxwEACAAJ.

[13] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," Psychological Review, vol. 65, pp. 386-408, 1958, https://doi.org/10.1037/h0042519.

[14] K. J. Hunt, D. Sbarbaro, R. Zbikowaski and P. J. Gawthrop, "Neural Networks for Control Systems-A Survey," Automatica, vol. 28, no. 6, pp. 1083-1112, 1992, https://doi.org/10.1016/0005-1098(92)90053-I.

[15] IEEE Control Systems Magazine, Special Issue on Neural Networks, volumes 8-10, 1988 -1990, https://books.google.co.id/books?id=hdA2mgEACAAJ.

[16] W. T. Miller, R. S. Sutton and P. J. Werbos, Neural Networks for Control, MIT Press, Cambridge, MA, 1995, https://books.google.co.id/books?id=prjMtIr_yT8C

[17] W. T. Miller, R. S. Sutton and P. J. Werbos, *Neural Networks for Control*, MIT Press, Cambridge, MA, 1990.

[18] G. Carpenter, M. Cohen and S. Grossberg, "Computing with Neural Networks," Science, vol. 235, pp. 1226-1227, 1987, https://www.science.org/doi/pdf/10.1126/science.3823881.

[19] M. Arbib and L. Hanson, *Vision Brain and Cooperative Computation*, MIT Press, Cambridge, MA, 1988, https://dl.acm.org/doi/abs/10.5555/26803.26804.

[20] W. J. Daunicht, “Defanet-A Deterministic Approach to Function Approximation by Neural Networks,” IEEE Int. Joint Conference on Neural Networks, IJCNN’90, 1990, pp. 161-164.

[21] T. Kohone, “Self-Organization and Associative Memory,” Springer-Verlag, Berlin, 1987, https://link.springer.com/book/10.1007/978-3-662-00784-6

[22] J. J. Hopfield, “Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons,” Proc. of the National Academy of Sciences, vol. 81, pp. 3088-3092, 1984, https://doi.org/10.1073/pnas.81.10.3088.

[23] H. Rahmanifard, T. Plaksina, “Application of artificial intelligence techniques in the petroleum industry: a review,” Artif. Intell. Rev., 2018, https://doi.org/10.1007/s10462-018-9612-8.

[24] O. Araque, I. Corcuera-Platas, J. F. Sanchez-Rada, C. A. Iglesias, “Enhancing deep learning sentiment analysis with ensemble techniques in social applications,” Expert Syst. Appl., vol. 77, 2017, https://doi.org/10.1016/j.eswa.2017.02.002.

[25] M. Jahnavi, “Introduction to Neural Networks, Advantages and Applications,” Towards Data Science, 2017, https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207.

[26] S. B. Kotsiantis, “Supervised machine learning: a review of classification techniques,” Informatica, vol. 31, no. 3, pp. 249–268, 2007, https://www.informatica.si/index.php/informatica/article/view/148.

[27] L. Zhang, S. Wan, B. and Liu, “Deep Learning for Sentiment Analysis: A Survey,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 8, no. 4, 2018, https://doi.org/10.1002/widm.1253.

[28] M. Gheisari, G. Wang, M. Z. A. Bhuiyan, “A survey on deep learning in big data,” 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2017, https://doi.org/10.1109/CSE-EUC.2017.215.

[29] D. E. Rumelhart, and J. L, McClelland (Editors), “Parallel Distributed Processing: Explorations in the Microstructures of Cognition vol. I: Foundations,” MIT Press, Cambridge, 1986, https://doi.org/10.7551/mitpress/5236.001.0001.

[30] R. S. Scalero and N. Teoedelenlioglu, “A Fast Algorithm for Neural Networks,” IEEE Int. Joint Conference on Neural Networks, IJCNN’90, 1990, pp. 70-74.

[31] Hinton, Geoffrey & Osindero, Simon & Teh, Yee-Whye, “A Fast Learning Algorithm for Deep Belief Nets. Neural computation,” Neural Computation, vol. 18, pp. 1527–1554, 2006, https://doi.org/10.1162/neco.2006.18.7.1527.

[32] V. S. Dave, K. Dutta, “Neural network-based models for software effort estimation: a review,” Artif. Intell. Rev., vol. 42, no. 2, pp. 295-307, 2014, https://doi.org/10.1007/s10462-012-9339-x.

[33] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, New York, 1999, https://books.google.co.id/books?id=bX4pAQAAMAAJ.

[34] T. Soderstrom and P. Stocia, *System Identification*, Prentice Hall, New York, 1989, https://books.google.co.id/books?id=X_xQAAAAMAAJ.

[35] M. J. Willis, C. Di Massimo, G. A. Montague, M. T. Tham and A. J. Morris, “On Artificial Neural Networks in Process Engineering,” Proc. IEE Part D., vol. 138, 1991, pp. 256 266, https://doi.org/10.1049/ip-d.1991.0036.

[36] K. S. Narendra and K. Parthasarathy, “Identification and Control of Dynamical Systems using Neural Networks,” IEEE Trans. Neural Networks, vol. 1, no. 1, pp. 4-27, 1990, https://doi.org/10.1109/72.80202.

[37] K. S. Narendra, “Neural networks for control theory and practice,” in Proceedings of the IEEE, vol. 84, no. 10, pp. 1385-1406, Oct. 1996, https://doi.org/10.1109/5.537106.

[38] B. Buchanan and E. Shortliffe, Rule-Based Expert Systems, Addison-Wesley, Reading, Mass., 1984, https://books.google.co.id/books?id=0uZQAAAAMAAJ.

[39] L. Brownston, R. Farrell, E. Kent and N. Martin, *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*, Addison-Wesley, Reading, Mass., 1985, https://books.google.co.id/books?id=9_2yAAAAIAAJ.

[40] R. Fikes and T. Kehler, “The Role of Frame-Based Representation in Reasoning,” Comm. ACM, vol. 28, 1985, pp. 904-920, https://doi.org/10.1145/4284.4285.

[41] P. Hirsch, M. Meier, S. Snyder and Stillman, "Interfaces for Knowledge Builder's Control Knowledge and Application-Specific Procedures," IBM J. Research and Development, vol. 30, 1986, pp. 29-38, https://doi.org/10.1147/rd.301.0029.

[42] M. Stefik, D. G. Borow, S. Mittal and L. Conway, "Knowledge Programming in LOOPS: Report on an Experimental Course," The Artificial Intelligent Magazine, vol. 4, no. 3, pp. 3-18, 1983, https://ojs.aaai.org//index.php/aimagazine/article/view/400.

[43] R. H. Michaelson, D. Michie and A. Boulanger, "The Technology of Expert Systems," Byte, vol. 10, 1985, pp. 303-312.

[44] E. H. Mamdani, "Applications of Fuzzy Algorithms for Simple Dynamic Plant," Proc. IEE, vol. 121, 1974, pp. 1585-1588, https://doi.org/10.1049/piee.1974.0328.

[45] M. Jamshidi, N. Vadiee and T. J. Ross (Editors), Fuzzy Logic and Control: Software and Hardware Applications, Prentice Hall, New York, 1993, https://books.google.co.id/books?id=fN9SAAAAMAAJ.

[46] T. J. Ross, *Fuzzy Logic with Engineering Applications*, John Wiley & Sons, 2016, https://books.google.co.id/books?id=ijsbDQAAQBAJ.

[47] J. T. Spooner and K. M. Passino, "Stable Adaptive Control using Fuzzy Systems and Neural Networks," IEEE Trans. Fuzzy Systems, vol. 4, pp. 339-359, 1996, https://doi.org/10.1109/91.531775.

[48] R. F. Stengel, "Toward Intelligent Flight Control," IEEE Trans. Systems Man and Cybernetics, vol. 23, pp. 1699-1717, 1993, https://doi.org/10.1109/21.257764

[49] B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice Hall, New York, 1992, https://books.google.co.id/books?id=fbJQAAAAMAAJ.

[50] K. M. Passino, *Intelligent Control for Autonomous Systems*, IEEE Spectrum, vol. 32, no. 6, pp. 55-62, 1995, https://doi.org/10.1109/6.387144.

[51] K. P. Valavanis and G. N. Saridis, *Intelligent Robotic Systems: Theory Design and Applications*, Springer Science & Business Media, 2012, https://doi.org/10.1007/978-1-4615-3568-3.

[52] D. White and D. Sofge (Editors), *Handbook of Intelligent Control: Neural Fuzzy and Adaptive Approaches*, Van Nostrand Reinhold, New York, 1992, https://books.google.co.id/books?id=_qEeAQAAIAAJ.

[53] M. C. McFarland and T. J. Kowalski, "Incorporating Bottom-up Design Into Hardware Design," IEEE Trans. CAD, vol. 9, no. 9, pp. 938-950, Sept. 1990, https://doi.org/10.1109/43.59070.

[54] F. Brewer, et al., "CHIPPE: A System for Constraint Driven Behavioral Synthesis," IEEE Trans. CAD, 9, pp. 681-695, July 1990, https://doi.org/10.1109/43.55208.

[55] M. S. Abadir, M. A. Breuer, "A Knowledge-based System for Designing Testable VLSI Circuits," IEEE Design and Test, 2, pp.56-68, Aug., 1985, https://doi.org/10.1109/MDT.1985.294746.

[56] A. J. Wilkinson, "MIND: An inside Look at an Expert System for Electronic Diagnosis," IEEE Design and Test, vol. 2, no. 4, pp. 69-77, 1985, https://doi.org/10.1109/MDT.1985.294748.

[57] R. Davis, "Diagnostic Reasoning Based on Structure and Behavior," Art. Intell., 24, pp. 347-410, 1984, https://doi.org/10.1016/B978-0-444-87670-6.50010-8.

[58] H. Tolba and A. K. Ali, "Fault Diagnosis of Digital Circuits Using CSP Techniques," 4th IEEE International Conf. On Electrical Circuits and Systems ICECS, Dec. 1997.

[59] S. J. Cosgrove and G. Musgrave, "Test Generation within an Expert System Environment," IEE Proc. -E (Computers and Digital Techniques), vol. 138, no. 1, pp. 36-40, Jan. 1991, https://doi.org/10.1049/ip-e.1991.0005.

[60] M. J. Bending, "HITEST: A Knowledge-based test Generation System," IEEE Design and Test, 1, pp.83-92, May, 1984, https://doi.org/10.1109/MDT.1984.5005617.