An Expert System for Automated Highway Driving

Axel Niehaus and Robert F. Stengel

The principal objective of this work is to study the applicability of expert systems to the task of guiding an automobile on a limited-access highway. The vehicle is assumed to be equipped with sensors detecting the surrounding traffic, road signs, and road geometry, as well as control logic and actuators governing the throttle, steering angle, and brakes. The goal of the expert system is to issue commands to the controllers, given the traffic situation, traffic signals, road signs, and the strategy chosen by the driver. The system presented here consists of a *rule base* providing the required driving knowledge, a *backward-chaining inference engine* that performs the reasoning, a *knowledge-base compiler* that optimizes the reasoning process, and a *highway-traffic simulator* that simulates vehicles on a highway, either controlled by a preset strategy or by an instance of the expert system.

Introduction

Population growth is increasing the number of vehicles and passengers on the nation's highways, causing increased traffic congestion and diminished safety margins, particularly in metropolitan areas [1]-[3]. Given current trends in suburban and urban development, it is unlikely that these problems can be solved entirely by building new highways. Furthermore, it is undesirable to restrict commuting and commercial traffic by requiring car pooling or by limiting access of certain vehicle types to special roads or lanes.

Intelligent vehicle/highway systems (IVHS) have been suggested as a solution for safe and efficient travel on increasingly congested arteries [3],[4], and have been the object of much research abroad as well in the U.S. [5]-[10]. The term "IVHS" describes a wide range of devices, including traffic management systems that control the overall flow, vehicle systems that aid the motorist, and cooperative systems that link the two. One function of the second type is "automatic chauffeuring," which requires sensing, actuation, and computational "intelligence" on board the vehicle. The intelligence that must go into such a system is the subject of this paper.

Specifically, we are interested in an intelligent guidance system for an automobile on a limited access highway, capable either of driving the vehicle in a fully automatic way or of giving the human driver useful advice. Such a system hopefully would enable the operation of vehicles at higher speeds while, at the same time, reducing the likelihood of highway accidents caused by human errors [4],[11]. Whether or not a fully automatic vehicle would be a desirable option remains to be determined. Knowing how well a computer-controlled vehicle would perform can help answer this question.

Fig. 1 shows a block diagram of an intelligent guidance system for headway and lane control. Three major issues have to be addressed in the development of such a system: providing an interface between the system and the real world, building an intelligent guidance system that is capable of planning and producing control commands, and developing controllers that implement these commands for accelerating, braking, and steering the vehicle. A dynamic expert system can be developed to perform the intelligent guidance function. With few exceptions, prior work on expert systems has been directed at static systems, i.e., expert systems that work with stationary or quasi-stationary knowledge bases and that do not have to produce goal states in "real time" [12],[13]. (In the present context, "real time" implies that problems would be solved at the time scale of vehicle motions.) Dynamic expert systems differ from their static counterparts in that they are included in the control loop; consequently, they have strong execution time constraints [14]-[17].

The first section of this paper introduces the concept of intelligent automotive guidance and defines the goal as well as the various functions of the expert system implementing it. The second section is dedicated to the development of the different parts of an expert system that has been implemented, and it addresses issues related to strategy estimation of other vehicles. The third section describes the implementation and the testing environment, including the highway traffic simulator and simulation results. The final section summarizes the benefits associated with the use of the described system.

The Intelligent Automotive Guidance Concept

While driving a vehicle on a limited access highway, a driver has to make many guidance and control decisions. The relatively mechanical task of controlling the vehicle can be distinguished from the guidance problem: making guidance decisions for a vehicle requires careful planning, analysis of the traffic situation, and logical reasoning, all of which appear to be "intelligent" tasks, demanding precise control and qualitative decision making. In this context, a rule-based expert system, called intelligent guidance for headway and lane control (IGHLC), has been developed to provide the necessary "intelligence" and symbolic computation power for the guidance task, the actual control of the vehicle being left to other electromechanical systems. Using rule-based expert systems for the guidance task has two main advantages. The guidance task is structured and can easily be divided into functions that can be implemented with a limited number of rules. This yields a system that is clear and easy to develop, test, and maintain. Unlike most expert systems, the developer can use his or her own driving knowledge as a starting point for developing rules.

The block diagram in Fig. 1 depicts the

Presented at the 1990 American Control Conference, San Diego, CA, May 23-25, 1990. *The authors are with the Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ 08540.* *This work was supported by a grant from the General Motors Research Laboratories, Warren, MI.*

0272-1708/91/0400-0053$01.00© 1991 IEEE

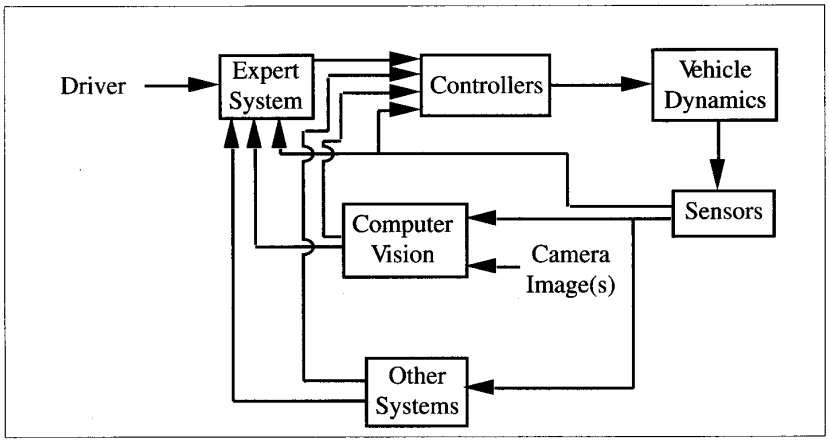


Fig. 1. Block diagram of intelligent guidance and control system.

integration of the IGHLC expert system in the control loop of the vehicle. The expert system must analyze the information it receives from the human driver and the various sensors and issue commands to the controllers. The controllers perform the basic control of the vehicle, such as steering to stay in the center of the lane, to change lane, and to command the throttle and brake to the desired speed or acceleration. Typical commands issued by the expert system are *Stay in lane and accelerate to 55 MPH* or *Perform an immediate left lane change while maintaining constant speed.* At each iteration, the IGHLC system has to generate two commands to issue to the controllers — one for the lateral controller (lane control), and one for the longitudinal controller (headway control).

The IGHLC expert system performs several functions. The top-level search function is in some sense the "executive" of the expert system, and it guides the other functions to find the two parameters that command the two controllers (Fig. 2). These controller parameters are determined either by a normal expert or by an emergency expert subsystem. Many of the rules needed during "normal" driving are not valid in an emergency situation, when the primary task is to avoid danger. Thus, the expert system must decide whether or not an emergency exists, using logic in the box marked situation assessment. Given information about the road geometry, the positions and velocities of the IGHLC-controlled car and other vehicles, as well as an estimate of what to expect from the other vehicles, the situation is analyzed and control is passed on to either the emergency expert or the normal driving expert (indicated in Fig. 2 by the dotted lines between situation assessment and the switch).

In each expert subsystem, an optimal trajectory generator produces candidate plans for the vehicle, using the same information as used in situation assessment (Fig. 3). Once the options have been identified, an evaluator picks the one that satisfies selection criteria. In an emergency, the only criterion used is safety, and the emergency trajectory evaluator selects the trajectory offering the highest security. In the normal driving situation, safety is not the only criterion; the normal trajectory evaluator selects a trajectory that is safe but that also satisfies the goal of the vehicle. This goal may be preset by the driver, for example setting it to *Cruise at 60 MPH.* The system does not force the driver to impose a strategy; where no goal has been entered, it chooses its own default strategies.

Expert System Development

The IGHLC expert system is composed of a *data base*, a *rule base*, and an *inference engine*. The *data base* contains information about the traffic situation. The *rule base* contains knowledge about highway driving. The *inference engine* uses the data base and the rule base to obtain a guidance decision.

Data Base

The data base is a set of external and internal parameters that have symbolic or numerical values. *External parameters* represent the traffic situation and the driver inputs. Traffic situation parameters are updated at every iteration of the control loop. *Internal parameters* are used by the expert system to store information that has been inferred by the system. Consequently, all internal parameters are initially set to *unknown.*

Rule Base

The knowledge about highway driving is given to the system in the form of rules contained in a rule base. A rule is an "if-then" statement containing a *premise* and an *action.* A major distinction can be made between the rule base of the IGHLC expert system and the rule base of a static expert system. In most cases, the expert system is permitted to query the user for more information, if needed during the reasoning process [12],[13]. Because the IGHLC system has to determine the controller

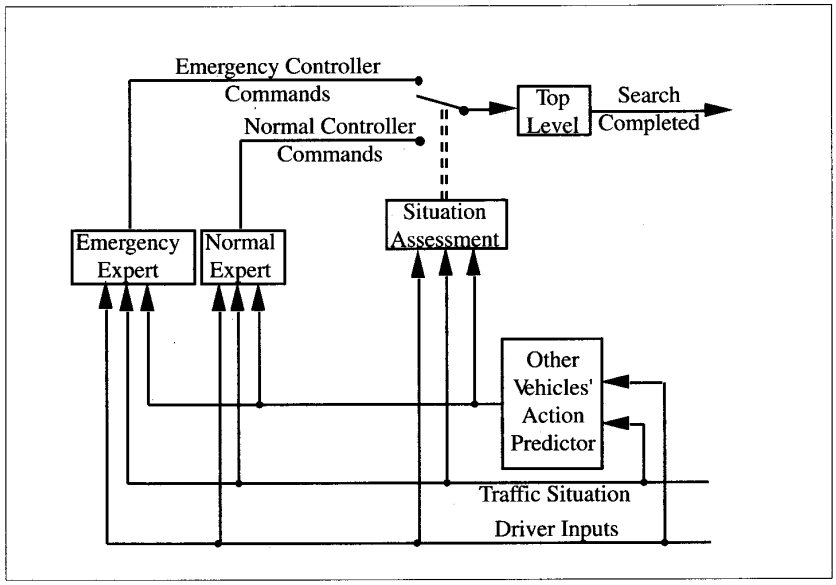


Fig. 2. Block diagram of expert system.

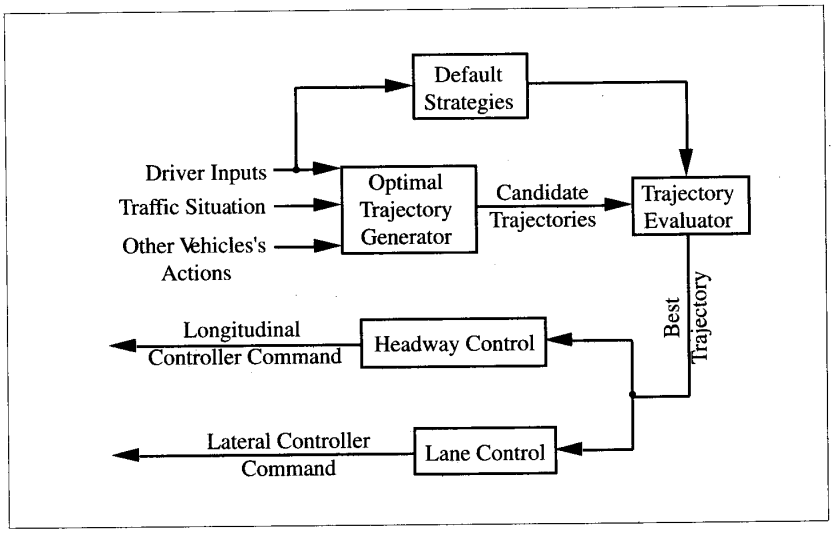


Fig. 3. Block diagram of normal and emergency experts.

parameters at a relatively high rate (about five times per second), there is no time for such questions, so the IGHLC rule base must be complete, enabling the determination of the controller parameters from the given data in *all* cases. As a result, all the functions of Fig. 2 and 3 have been implemented in the IGHLC rule base. Table I shows the various sections of the rule base, along with the numbers of rules, external parameters, and internal parameters in each.

Top-Level Search. The unique rule in the top-level search function has the task of guiding the other functions to determine the value of the two controller parameters. In Symbolics Common Lisp [19], it is stated as:

```
(SETQ RB-TOP-RULE1 (MAKE-RULE
:NAME          'RB-TOP-RULE1
:PREMISE       '(COND
    (( OR(EQ $LONGITUDINAL-
     CONTROLLER-COMMAND
    'UNKNOWN)
     (EQ $LATERAL-
    CONTROLLER-COMMAND
     'UNKNOWN))
    'UNKNOWN)
    ( T T ))
:ACTION    '(SETQ $SEARCH
        -COMPLETED T)
:DOC"The longitudinal and lateral controller
        commands have been found, so the
        search is completed." ))
```

The first line sets the value of the parameter RB-TOP-RULE1 to the rule structure created by the MAKE-RULE instruction. The following eight lines specify the name of the rule, the premise of the rule, its action, and a documentation string to be printed when the rule fires, which is used to analyze the reasoning process of the inference engine. The premise of this rule tests the two internal parameters $LONGITUDINAL-CONTROLLER-COMMAND and $LATERAL-CONTROLLER-COMMAND. (A dollar sign as the first character of a parameter name indicates an internal parameter. As explained later, this is used during compilation). If one of these two parameters has a value of *unknown*, the premise evaluates to *unknown*; as soon as both are known, its value becomes *true*. When this happens, the action of the rule sets internal parameter $SEARCH-COMPLETED to *true*. At each iteration in the control loop, the inference engine is called to determine the value of $SEARCH-COMPLETED. Because this top-level search rule is the only rule affecting this parameter, the inference engine deduces that it must find the value of its premise. This guides the search towards the two controller parameters.

Situation Assessment. This group of rules determines whether or not the traffic situation is an emergency. Since "emergency" has a rather wide meaning, it must be given a precise definition. The situation assessment rules define a traffic situation to be an emergency when a deceleration of more than some threshold percentage of the maximum current obtainable deceleration is needed to avoid a collision. As a result, this section contains rules that compute mean decelerations and that set the internal parameter $EMERGENCY accordingly. Nevertheless, what looks as simple as just deciding whether or not a situation is an emergency can be quite complex. For example, if another vehicle is ahead, what is the system supposed to "think" that vehicle will do in the future? Will it accelerate? Will it slam on its brakes? It is clear that whatever is assumed on the part of the system will largely affect its response, and this section uses other lower-level sections to make assumptions on the actions of other vehicles.

Emergency Expert. In an emergency, the IGHLC system tries to maximize safety. Given a vehicle A and the vehicle A+1 ahead

Table I
Number of Rules, Internal, and External Parameters in Various Sections of Rule Base

Function	Number of Rules	Number of External Parameters	Number of Internal Parameters
Top-Level Search	1	0	3
Situation Assessment	13	5	9
Emergency Expert	31	14	74
Normal Expert	37	15	64
Acceleration Expert	19	10	12
Optimal Trajectory Generator	24	50	74
Projected Action Determination	14	3	1
Other Vehicles's Action Determination	31	23	20
Default Strategies	6	7	3

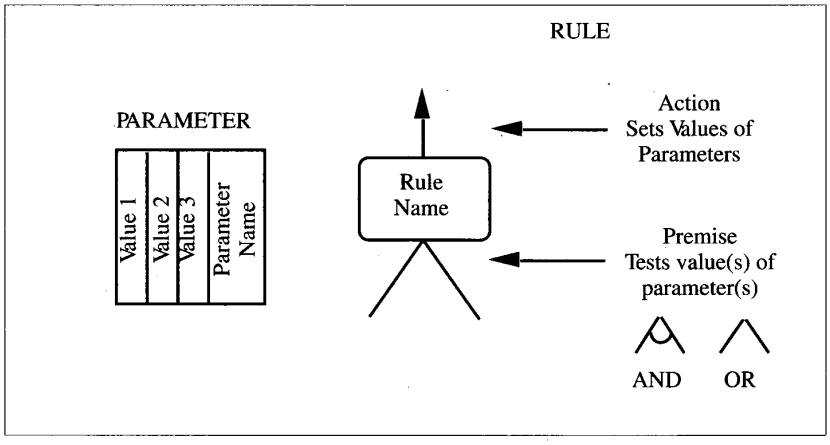


Fig. 4. Graphical representation of parameters and rules.

of it, the *security ratio* SR_A of vehicle A is defined as

$$SR_A = \frac{\text{distance (vehicle } A, \text{ vehicle} A+1 \text{)}}{\text{Security Distance (vehicle } A \text{)}}.$$

Security distances are consistent with the guidance provided by a typical drivers' manual, based on vehicle speed and road condition [20]. The *minimum security ratio* is the minimum, over the intended trajectory, of the security ratios of the IGHLC car and the vehicle behind it. The rules implementing the emergency trajectory evaluator choose the option with the highest minimum security ratio. Since the vehicles ahead and behind the IGHLC car are not, in general, the same over the whole intended trajectory, this ratio depends on the vehicles in adjacent lanes. In taking the minimum over the intended trajectory, the system not only takes into account which position on the road is currently desirable but the implications of that decision in the future. A decision that looks promising but actually leads to a critical situation after a certain time is discarded. To avoid oscillations in the decision, the system accounts for the *projected action*, i.e., the decision of the preceding iteration of the guidance loop, adapted to the present situation. For example in an emergency, the system always tries to follow the projected action, unless there exists another option with a significantly better minimum security ratio. This insures persistency in the output of the system.

Normal Driving Expert. When not in an emergency, the goal of the expert system is to satisfy the driver, who specifies cruising speed and two external parameters: SECURITY-FACTOR and AGGRESSIVENESS-FACTOR. All other guidance decisions, such as choice of lane, are made by the expert system. Given a goal to achieve, drivers may define optimum response differently. Some prefer to be very cautious while others like to maintain a minimum security level. Some are comfortable with high levels of acceleration and deceleration, and others barely touch the accelerator or brake. The purpose of the security and aggressiveness parameters is to enable the driver to choose a comfortable driving style.

When the driver omits certain inputs, like telling the computer what maximum speed he or she authorizes for passing maneuvers (this speed is usually taken higher than the speed limit), the computer must be able to choose

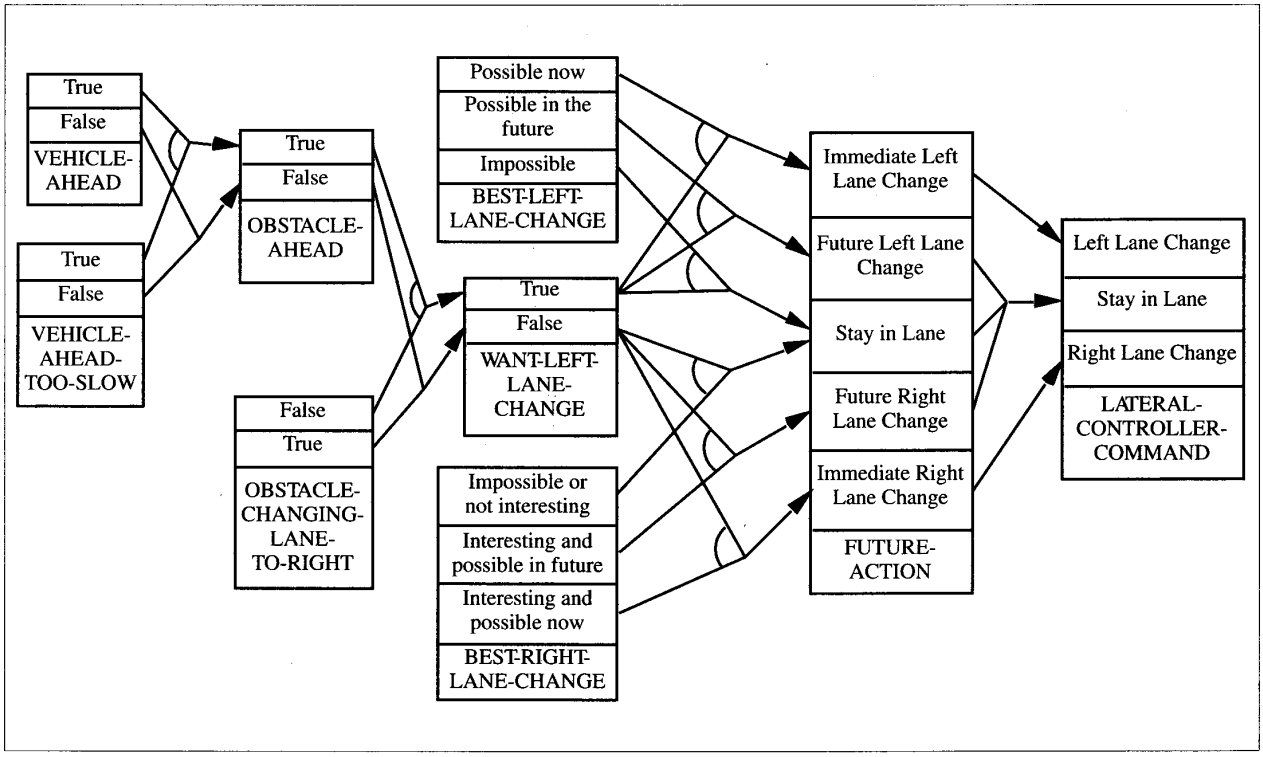


Fig. 5. Simplified logic for normal driving.

default strategies. This task is performed by the default strategies function, the defaults being chosen in terms of the aggressiveness and security factors given by the driver. Maintaining a given speed on a highway is not always possible, and in order to come close to it, the computer will have to perform lane changes and passing maneuvers. However, unlike the emergency expert, the normal driving expert does not have the right to command illegal actions. Given most state laws, for example, a car must drive in the rightmost lane unless a passing maneuver is desired. Passing on the right is prohibited, even in the case of a slow vehicle ahead in the left lane with no vehicle in the right lane. A left lane change has to be performed to try to make that vehicle change lane to the right, in order to pass it.

Logical relationships within the expert system can be represented graphically (Fig. 4). Each rectangle contains a parameter, along with the list of values the parameter can acquire. A rule is represented by arrows, lines, and a rounded box, the latter containing the name of the rule. The arrow leaving a given box points to the value of the parameter set by the action of the corresponding rule. The premise of a rule is represented by lines between the parameters it tests and the box of the rule. Multiple conditions represented by multiple lines are treated as a disjunction (OR), unless there is an arc joining them, in which case they are treated as a conjunction (AND).

Consider a normal driving example (Fig. 5). First, the system determines if there is a slow vehicle ahead, in the same lane as the IGHLC car. If *yes*, and if the vehicle does not show any signs of a right lane change, left lane changes are examined. If the system determines that a left lane change is or will be available, it guides the vehicle accordingly. Where no left lane change is available in the near future, the system computes an approach trajectory to the vehicle ahead and waits. When there is no vehicle ahead, or if the vehicle ahead is changing lane to the right, right lane changes are examined. Unlike left lane changes, a right lane change will be made only if there is one available *and* the corresponding position in the right lane will not immediately lead to a lane change back to the current lane, i.e., it is "*interesting*." If there are slow vehicles in the right lane, the normal driving expert will issue commands to stay in the passing lane.

Optimal Trajectory Generator. Both the emergency and the normal driving experts use an optimal trajectory generator to compute candidate trajectories. The current rule base uses position, velocity, and state of brake lights and turn signals of other cars as inputs. One important aspect of the expert system is to produce an accurate estimate of what other vehicles will do in the future (Fig. 2). The easiest estimate is to assume that all vehicles move at constant speeds and do not change lanes. For small time scales, this works reasonably well, and it has the advantage of being very fast; however, when the prediction time interval is pushed further ahead, uncertainty about other cars' actions increases. An intelligent system would analyze how the other vehicles are driven, deduce the strategy underlying those actions, and thus deduce the most probable future actions of these vehicles. Such a system may be too complex to be considered for real-time application, so some compromise between accuracy and execution time must be found. The current system is limited to rules that predict the longitudinal motions of the other vehicles. One of the goals of the testing environment described in the next section is to find out just how good the estimates of the other vehicles' actions must be.

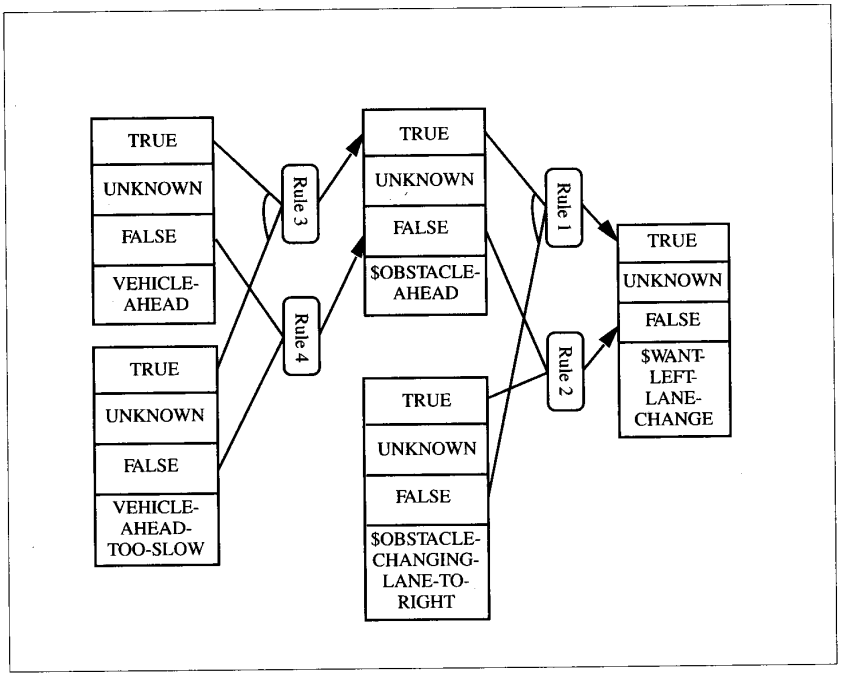


Fig. 6. Simple example of knowledge base used to explain backward chaining.

Inference Engine

Given the knowledge about highway driving contained in the rule base, the goal of the expert system is to infer the two controller commands (internal parameters since they are generated by the system), using information about the traffic situation and driver inputs. The inference engine is a program that applies rules from the rule base to the knowledge in the data base to infer new knowledge. Given the name of an internal parameter, the inference engine searches the rule base for a list of rules. These rules are selected so that when applied in sequence, they infer new knowledge until the last rule provides a value for the desired parameter. There is not one single rule that immediately yields the two controller commands; the system has to use many rules to determine other parameters first. A typical example of this is the internal parameter $EMERGENCY. Initially, the system does not know whether or not the traffic situation is an emergency, and the value of $EMERGENCY is *unknown*, forcing the inference engine to first determine its value by using other rules.

The search method implemented by the inference engine is called *backward-chaining*, which examines only those rules that effectively have a chance of giving a result. *Forward-chaining* repetitively tests all the rules, in some arbitrary order, until either no more rules fire (no more information can be inferred) or the value of the desired parameter has been found. Backward-chaining starts with the desired parameter, and searches backwards to determine if part of the initial knowledge will produce a value [18]. The fact that the IGHLC rule base is well structured in groups of rules implementing the various functions of Fig. 2 and 3 insures that only a minimal number of rules will be examined.

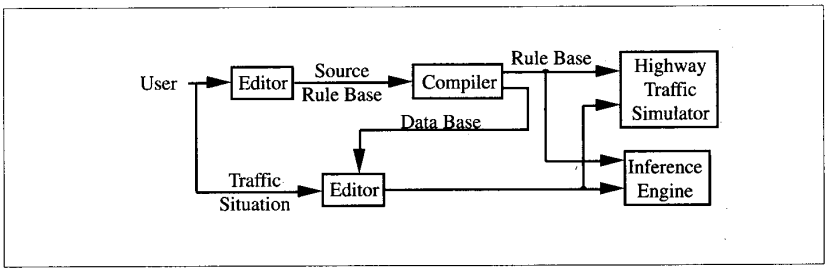


Fig. 7. IGHLC rule base development and testing environment.

Backward-chaining can be explained using a simple example: determining if a left lane change is wanted in the sample knowledge base shown in Fig. 6. The inference engine is initially called to determine the value of internal parameter $WANT-LEFT-LANE-CHANGE. First, it checks to see if this parameter already has a value. If it does, the inference engine returns the corresponding result. In the more probable case where the parameter is unknown, the inference engine searches the rule base for rules setting $WANT-LEFT-LANE-CHANGE, finding Rule 1 and Rule 2. Rule 1 is then tested, checking for an obstacle ahead.

Let's assume a traffic situation where there is a slow car ahead that is not changing lane to the right. $OBSTACLE-AHEAD is an internal parameter, and its value is initially *unknown*. Rule 1 is unable to provide an immediate value for $WANT-LEFT-LANE-CHANGE, so Rule 2 is tested. Since the value of OBSTACLE-CHANGING-LANE-TO-RIGHT is *false* (by assumption), Rule 2 also needs the value of parameter $OBSTACLE-AHEAD to find the truth value of its premise. Since these are the only rules capable of providing a value for $WANT-LEFT-LANE-CHANGE, the inference engine concludes that it must find a value for parameter $OBSTACLE-AHEAD. It calls itself recursively to find the new desired parameter. Upon entry, the inference engine starts a search for the rules setting $OBSTACLE-AHEAD, thus finding Rules 3 and 4. Rule 3 is tested first; the inference engine finds that its premise equals *true* (the values of external parameters VEHICLE-AHEAD and VEHICLE-AHEAD-TOO-SLOW are *true* by assumption). The inference engine executes its action part, setting $OBSTACLE-AHEAD to *true*. Having found a value for parameter $OBSTACLE-AHEAD, the inference engine is exited, and returns to the first call. The value of the premise of Rule 1 now being *true*, Rule 1 fires, setting $WANT-LEFT-LANE-CHANGE to *true*, and finally, the inference engine is exited with the desired value, *true*.

Implementation of the IGHLC Expert System

The IGHLC system originally was implemented on a Symbolics 3670 LISP Machine, using the Common LISP language for the inference engine and the logic part of the rule base, and FORTRAN for the numerical procedures implementing the optimal trajectory generator. The system has been transferred to a NeXT computer, using Common LISP and Objective C for implementation.

The current rule base consists of 172 rules, 86 external parameters, and 168 internal parameters. Given this size, and in general the complexity of the whole project, the need for a convenient and efficient testing and development environment is easily seen. The current development environment is shown in Fig. 7. The developer starts by writing rules to a *source rule base*, which is translated into an executable rule base by a *knowledge base compiler*. The executable rule base can be tested for single traffic situations, or, using a *highway traffic simulator*, it can be tested in a traffic situation that evolves in time. Given the results of testing, the developer can go back to the source rule base to change rules producing undesirable actions or add rules to improve it.

Knowledge Base Compiler

The knowledge base compiler has two functions. First, it parses the source rule base to produce a list of all internal and external parameters (it is during this parsing that the dollar sign is used to distinguish internal parameters from external ones), and it creates functions that initialize the rule base and data base. Second, it performs an optimization of the search process.

As seen in an earlier section, the inference engine must perform various searches to deduce the value of a parameter. These searches can be divided into four categories. First, given some internal parameter, the list of rules affecting that parameter must be determined. This list is used when deciding which rules to examine in order to obtain a value for the parameter. Second, given some internal parameter, the inference engine must search for all the rules that use the parameter in their premises. This determines which rules are affected when a value has been found for the parameter. Third, the inference engine needs to know which internal parameters figure in the premise of a given rule. This list of internal parameters is used when the premise of the rule evaluates to *unknown*, and the inference engine has to determine which parameters help to find a value for the premise. The last type of search is the determination of all the internal parameters set by the action of a given rule. This search is used when the rule fires, and the inference engine has to determine which rules might be affected by the new knowledge. It is clear that all four of these searches can be performed ahead of time, the results being stored away, together with the parameters and rules. The compiler performs this task, which would be tedious for the developer.

Using the compiled IGHLC rule base, current execution times are 1 to 4 seconds, depending on the traffic situation. These numbers are obtained while running in the LISP environment, which is good for rule base development but is too slow for real-time simulation (because of "garbage collection," among other things). The system could be translated into a more efficient language such as C or Pascal for true real-time operation, as demonstrated by the Princeton Rule-Based Cotrol System [21].

Highway Traffic Simulator

The system was first developed and tested using single traffic situations, where the developer gives the inference engine a data base containing the situation and compares the output of the reasoning process to the expected result. Currently under development is a traffic simulator that gives the developer a graphical view of the evolving situation when the expert system is driving a vehicle, and reveals the long term implications of a given decision.

The simulator uses five windows to obtain and display information. The Simulator Window is the graphical output, representing the road and the vehicles. This view is always centered vertically on the IGHLC car, the road background being scrolled down as the IGHLC car advances. The length of the portion of the road displayed varies between 144 ft and 864 ft, depending on the scale factor chosen by the tester. The control window enables the developer to click buttons to obtain more information, change the traffic situation, add or take out vehicles, choose

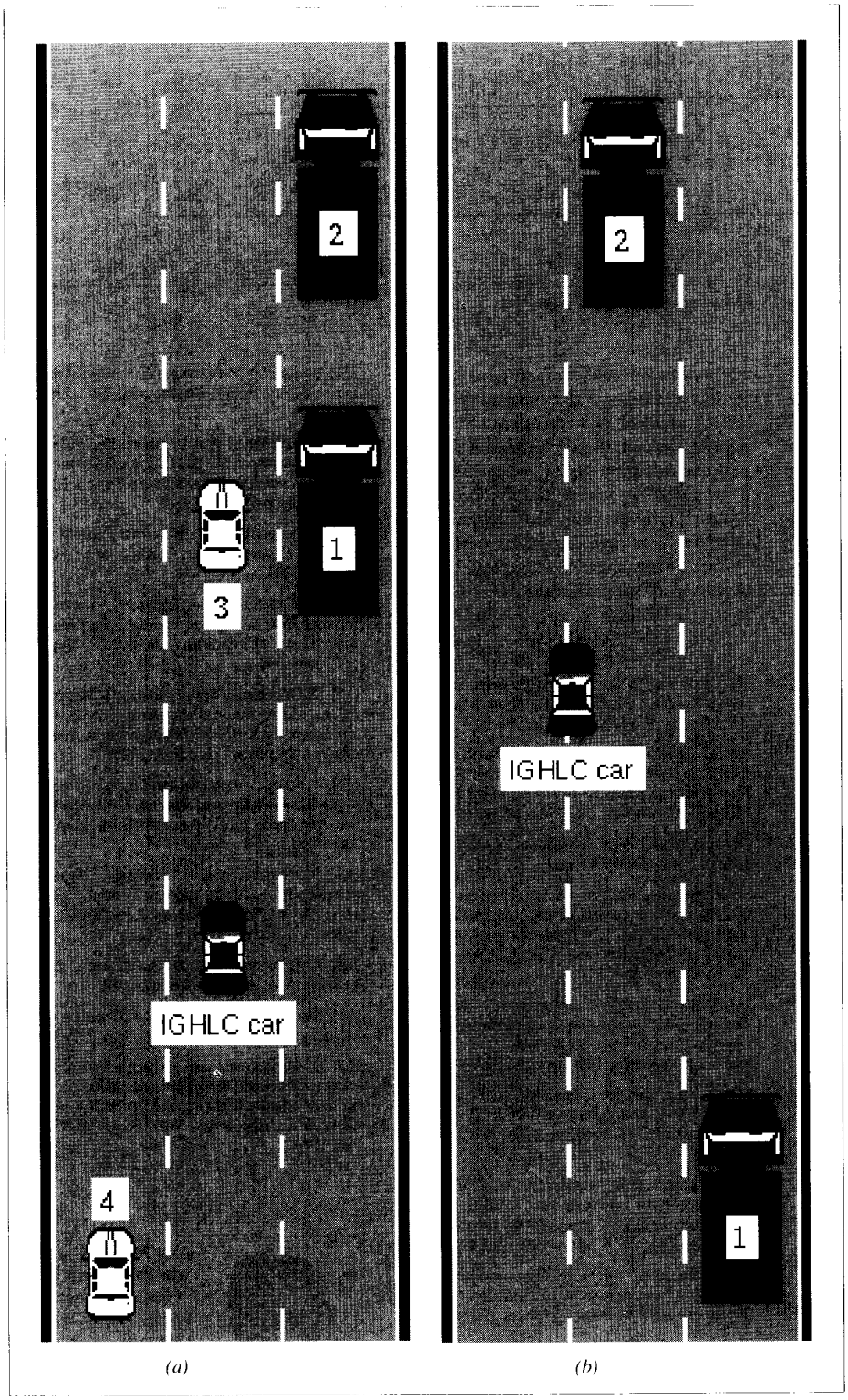


Fig. 8. (a) Initial traffic situation for simulation. (b) Emergency lane change performed by the IGHLC car during simulation.

strategies for vehicles, and load or save traffic situations or simulation results. Keyboard inputs are reflected in the Interaction Window. The other two windows display the rules that are being fired by the expert system and the parameters in the data base.

Using this simulator, the developer can follow the evolution of an initial traffic situation, while observing the rules and parameters that are responsible for it. One important feature of the simulator is that the developer can choose the strategies the various vehicles follow. In this way, a given vehicle can be controlled by an expert system, or it can simply have a predefined strategy. Examples of the use of this feature include the case where all the vehicles are controlled by similar expert systems (to analyze the effect of a given set of rules on traffic congestion and throughput), as well as the analysis of the expert system when confronted with "abnormal" cases.

IGHLC System Test

Fig. 8(a) shows an example traffic situation used to test the simulator. The IGHLC car is in dark gray, and it is surrounded by four other vehicles. Vehicles 1 and 2, ahead of the IGHLC car at 240 ft and 350 ft, are both trucks traveling at 40 ft/s. Ahead of the IGHLC car at 240 ft, Vehicle 3 is traveling at 80 ft/s. Finally, behind at 170 ft, Vehicle 4 is traveling at 85 ft/s. Except the IGHLC car, which is controlled by the IGHLC expert system, all vehicles stay in lane and travel at constant speeds. The IGHLC car is currently travelling at 80 ft/s, but it has a desired speed (set by the human operator) of 100 ft/s.

Initially, IGHLC ordered the car to be in the middle lane to pass the two trucks. Since Vehicle 3 was slower than the desired speed, the expert system determined that the IGHLC car should make a left lane change to the leftmost lane to pass it as well. However, since Vehicle 4 was coming up, a left lane change was judged unsafe, and the final decision was to stay in lane, following Vehicle 3 until Vehicle 4 had passed.

In order to test the Emergency Expert as well as the Normal Expert, the simulator forced Vehicle 2 (the front

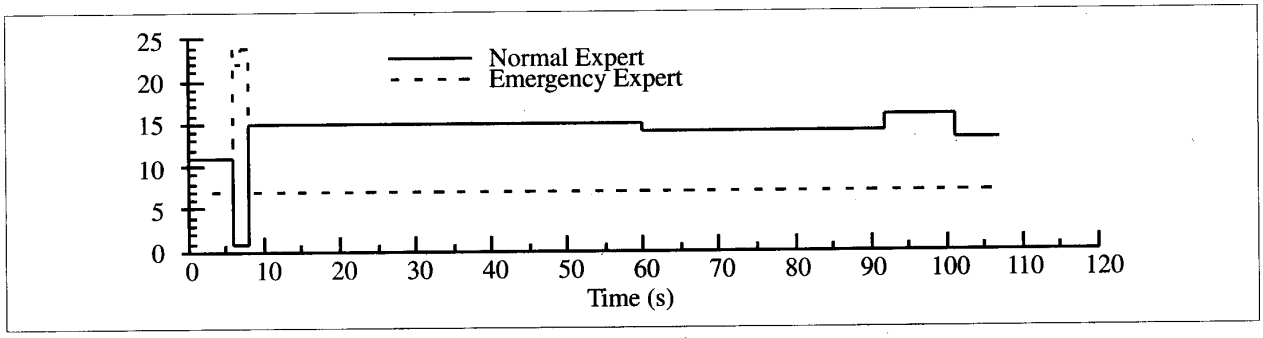


Fig. 9. Plot of normal and emergency expert activity during simulation of Fig. 8(a).

black truck) to change lane to the middle lane, as soon as the IGHLC car was next to Vehicle 1, creating an emergency given the closeness of Vehicle 2 (80 ft, taking into account the lengths of the vehicles) and the difference in speeds (40 ft/s). This effectively triggered the Emergency Expert, which determined that the safest option was to change lane to the left to avoid Vehicle 2, although Vehicle 4 was coming up behind. This lane change maneuver is shown in Fig. 8(b).

After passing Vehicle 2, the IGHLC car stayed in the left lane (since it already was there and Vehicle 4 had slowed down) to pass Vehicle 3. Finally after passing Vehicle 3, it returned to the rightmost lane, since no more obstacles were detected ahead.

To further analyze the expert system, a plot was produced showing the number of rules in the normal expert and emergency expert tested at every iteration (Fig. 9). When Vehicle 2 produces the emergency at $t = 6$ s, a peak shows up on the curve of the emergency expert activity (which goes from 7 rules tested per loop to 24) and the normal expert activity drops from 11 rules/loop to 1 rule/loop. After successfully changing lane to the left, the situation is reversed at $t = 8$ s, when the emergency is ended. The drop in the Normal Expert's activity at $t = 60$ s corresponds to the IGHLC car passing Vehicle 3. The increase at $t = 92$ s indicates that security distance in front of Vehicle 3 has been reached and the initialization of the right lane changes. Finally the drop at 101 s signals that the IGHLC car has reached the rightmost lane.

Conclusions

An expert system for intelligent guidance of a vehicle on a highway has been developed and tested using a highway simulator. There are two principal conclusions to be drawn. First, the expert system approach, combined with a structured rule base that implements the driver's many tasks appears to have many advantages, including ease of programming, testing, debugging, and breakdown of the driving problem in a consistent and natural manner. Second, the simulation results suggest that an expert system is capable of implementing the driving task. Although the system performs many driving functions, more details of highway driving must be included. Future research will extend the rule base to handle new situations.

References

[1] *Urban Traffic Systems and Parking*, Transportation Research Board, Transportation Research Record 1181, Washington, DC, 1988.

[2] A. Reno, "Personal mobility in the United States", *A Look Ahead Year 2020*, Special Rep. 220, Transportation Research Board, National Research Council, Washington, DC, 1988, pp. 369-393.

[3] G. Witcher, *Smart Cars, Smart Highways, Wall Street J.*, May 22, 1989, p. B1.

[4] P. Koltnow, "Advanced technologies: Vehicle and automobile guidance", *A Look Ahead Year 2020*, Special Rep. 220, Transportation Research Board, National Research Council, Washington, DC, 1988, pp. 425-429.

[5] D. Mc Mahon, J. Hedrick, and S. Shladover, "Vehicle modelling and control for automated highway systems", in *Proc. 1990 Amer. Control Conf.*, San Diego, CA, May 1990, pp. 297-303.

[6] H. Peng, and M. Tomizuka, "Vehicle lateral control for highway automation," in *Proc. 1990 Amer. Control Conf.*, San Diego, CA, May 1990, pp. 788-794.

[7] W. Zhang, and R. Parsons, "An intelligent roadway reference system for vehicle lateral guidance/control," in *Proc. 1990 Amer. Control Conf.*, San Diego, CA, May 1990, pp. 281-286.

[8] P. Bumann, "Vehicle communications in Europe," in *Proc. Convergence 1988 - Int. Congress Transportation Electronics*, 1988, pp. 229-235.

[9] Siemens AG, "LISB: Leit- und Informationssystem Berlin", brochure.

[10] P. Walzer and W. Zimdahl, "European Concepts for Vehicle Safety, Communication and Guidance", in *Proc. Convergence 1988 - Int. Congress Transportation Electronics*, 1988, pp. 91-95.

[11] *Annual Report to Congress*, National Transportation Safety Board, Washington, DC, 1983, pp. 21-28.

[12] B. Buchanan and E. Shortliffe, *Rule-Based Expert Systems : The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley, 1984.

[13] F. Hayes-Roth, D. Waterman, and D. Lenat, *Building Expert Systems*. Reading, MA: Addison-Wesley, 1983.

[14] D. Handelman and R. Stengel, "An architecture for real time rule-based control," in *Proc. 1987 Amer. Control Conf.*, Minneapolis, MN, June 1987, pp. 1636-1642.

[15] D. Handelman and R. Stengel, "Perspectives on the use of rule-based control", in *Proc. IFAC Workshop on Artificial Intelligence in Real-Time Control*, Swansea, U.K., Sept. 1988.

[16] R. Stengel, *Artificial Intelligence Theory and Reconfigurable Control Systems*, Dept. Mechanical and Aerospace Eng., Princeton Univ., Rep. 1664, June 1984.

[17] R. Stengel and A. Niehaus, "Intelligent guidance for headway and lane control," *Eng. Applications Artificial Intell.*, vol. 2, pp. 307-314, Dec. 1989.

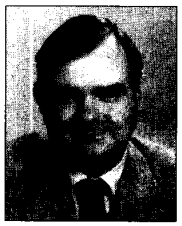
[18] P. Winston, *Artificial Intelligence*, 2nd ed. Reading, MA: Addison-Wesley, 1984.

[19] Symbolics, Inc, *Symbolics Common Lisp - Language Concepts*, Cambridge, MA, Feb. 1988.

[21] D. Handelman and R. Stengel, "Combining expert system and analytical redundancy concepts for fault-tolerant flight control," *AIAA J. Guidance, Control, and Dynamics*, Vol. 12, pp. 39-45, Jan.-Feb. 1989.

Axel Niehaus is a fourth year PhD student in Mechanical and Aerospace Engineering at Princeton University. He graduated from Ecole Centrale de Paris in 1987, and received his M.A. from Princeton University in 1989. His current research interests include intelligent estima-

tion, intelligent stochastic optimal control, and applications of neural networks to control.



Robert Stengel is a Professor of Mechanical and Aerospace Engineering at Princeton University, where he directs the Topical Program on Robotics and Intelligent Systems and the Laboratory for Control and Automation. Prior to his 1977 Princeton appointment, he was with The Analytic Sciences Corporation, The Charles Stark Draper Laboratory, U.S. Air Force, and National Aeronautics and Space Administration. A principal designer of the Project Apollo Lunar Module manual attitude control logic, he also contributed to the design of the Space Shuttle guidance and control system. Dr. Stengel received degrees from M.I.T. (Aeronautics & Astronautics, S.B., 1960) and Princeton University (Aerospace and Mechanical Sciences, M.S.E., M.A., Ph.D., 1965, 1966, 1968). He is an Associate Fellow of the AIAA, a Senior Member of the IEEE, and a Member of the SAE Aerospace Control and Guidance Systems Committee. Professional positions include Associate Editor at Large of the *IEEE Transactions on Automatic Control*, Member of the *Journal of Micromechanics and Microengineering* Editorial Board, and Chairman of the AACC Awards Committee. He was Vice Chairman of the Congressional Aeronautical Advisory Committee and has served on numerous governmental advisory committees. His current research focuses on system dynamics, control, and machine intelligence. He teaches courses on control and estimation, aircraft dynamics, space flight engineering, and aerospace guidance. Dr. Stengel wrote the book, *Stochastic Optimal Control: Theory and Application* (Wiley, 1986) and is writing a book on aircraft dynamics and control. He has authored or co-authored over 150 technical papers and reports.

1990 ACC Tutorial Workshops

Joe Chow, of Rensselaer Polytechnic Institute, Workshop Chairman for the 1991 American Control Conference to be held in Boston, MA, June 26-28, 1991, has announced workshop plans for the two days before the Conference. Six one-day workshops have been scheduled.

Recursive Algorithms for Tracking in Clutter

Yaakov Bar-Shalom, University of Connecticut, Monday, June 24, 1991. This workshop reviews some recent developments in real-time implementable recursive algorithms for maneuvering target tracking in clutter. The emphasis is on algorithms with fixed computational and memory requirements, rather than the much more complex approach of examining each possible sequence of measurements and dynamic behaviors. The workshop material - a 800 page set of notes "Multitarget-Multisensor Tracking: Principles and Techniques," and a PC demo diskette are available to the participants.

Theory and Applications of Intelligent Control Systems

Kimon Valavanis, University of S. W. Louisiana; Levent Acar, University of Missouri; K. M. Passino, Ohio State University, Monday, June 24, 1991. This workshop covers several different Intelligent Control Methods and approaches. Emphasis is given to the design of structure-based hierarchies and knowledge-rich distributed controllers, the analysis of planning systems in autonomous control, and modeling and analysis of general hierarchical systems with diagnostic intelligence capabilities. Applications to flexible manufacturing systems, robotic assemblies and robotic systems with diagnostic capabilities will be discussed.

Fault Tolerant Computer Control Systems

N. Viswanadham, Indian Institute of Science; K. Dean Minto, General Electric Company; K. Trivedi, Duke University Monday, June 24, 1991. This workshop will provide an integrated overview of the various aspects involved in designing fault tolerant computer control systems. Particular attention will be devoted to fault detection, isolation and reconfiguration (FDIR), fault masking, and fault coverage. Practical details of the application of EDIR and fault masking techniques will be presented. Recent results on robustness in FDIR schemes and multirate sampling will also be covered.

Discrete Event Systems: The State-of-the-Art, Theory and Applications

Xi-ren Cao, Digital Equipment Corporation; Christos G. Cassandras University of Massachusetts; Y. C. Ho, Harvard University, Tuesday, June 25, 1991. This tutorial workshop emphasizes the fundamental concepts, techniques, and practical methodologies in the modeling, analysis and control of discrete-event systems. The fundamental resource contention problems, simulation methods, software packages, and the advantages/disadvantages of simulation compared with analytical techniques will be discussed. Applications involving real-time control and optimization of computer networks and distributed processing systems will be presented

Neural Networks in Control Systems

K. S. Narendra, K. Perthasarathy, Yale University; P. Werbos, National Science Foundation; I. Unger, University of Pennsylvania, Tuesday, June 25, 1991. The workshop provides a guided tour through the rapidly expanding and often intricate field of neural networks. The application of well-established techniques in the areas of identification and control to the analysis and synthesis of dynamical systems containing artificial neural networks as subsystems will be discussed. The theoretical perspectives will be supplemented by applications in process control, flight control and robotics. The workshop will follow a forthcoming monograph "Neural Networks in Dynamical Systems" by K. S. Narendra.

Digital Implementation of Controllers

Herbert Hanselmann, dSpace Gmbh, Germany. Tuesday, June 25, 1991. This workshop shows what can be done in controller implementation using today's VLSI processor hardware and software technology. The implementation of high-order multivariable state-variable controllers for fast systems on standard microprocessors, signal processors and microcontrollers will be discussed. Case studies including mechatronics applications illustrate problems and solutions.

For further information, contact Professor Joe Chow, Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, New York 12180, telephone (518)276-6374 (email: chowj@ecse.rpi.edu).