Software and Hardware for Intelligent Robots

G.N. Saridis
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

K.P. Valavanis
Northeastern University
Boston, MA 02115

1. ABSTRACT

Various architectures and their respective software for Hierarchically Intelligent Robots are discussed in this paper. They conform to the Principle of Increasing Precision with Decreasing Intelligence by following a three-level structure. The architecture of the organization and coordination levels is presented here and their algorithms are outlined.

2. INTRODUCTION

Intelligent Robots are a special type of Intelligent Machines [5]. They may be driven by controls with special characteristics described by the methods of Hierarchically Intelligent Control Systems [3].

Hierarchically Intelligent Control is based on the Principle of Increasing Intelligence with Decreasing Precision. Intelligent Robots (and thus Intelligent Machines) may be modeled based on the constraints imposed by the Theory of Intelligent Controls. They are considered to be composed of three interactive levels, organization, coordination and execution. They utilize feedback mechanisms from the hardware processes of the execution level to the organizer selectively, by aggregation of the information at every level (Figure 1).

A three interactive level probabilistic model has already been defined [4,5] for Intelligent Robots. The organization level is modeled after a Knowledge Based System; it performs general knowledge processing tasks with little or no precision. The coordination level is composed of a specific number of coordinators, each performing its own pre-specified functions; it performs specific knowledge processing tasks. The execution level is composed of specific execution devices (hardware) associated with each coordinator. The probabilistic model is obtained by defining the various operations associated with each level of Intelligent Robots in a mathematical way and then assigning a probabilistic structure to organize the appropriate tasks for execution.

A simple but complete architectural model that can accommodate fast and reliable operation for the levels individual functions has also been derived [5]. Each level has a functional task to perform. The functional task of each level is performed in the best possible way based on accumulated information and related feedback from the lower levels. Upon completion of this functional task, a command is issued to the immediate lower level, and the functional task of the lower level is accomplished. With this structure, each level evaluates and controls the performance of the immediate lower one. The architectural model associated with top-down hierarchical knowledge (information) processing is suitable for the decision phase of Intelligent Robots. The decision phase involves formulating complete and compatible plans, deciding which one is the best to execute the requested job and how to execute it. Therefore, it may accept special simple architectures specifically designed to implement the Hierarchically Intelligent Control Algorithms.

Specific hardware units must be built within each of the higher two levels for the upgrade phase of Intelligent Robots. This phase involves the upgrade of the individual and accrued costs and probabilities associated with a particular plan. Upgrade follows plan execution (completion of the decision phase) and is performed in a bottom-up way: costs and probabilities associated with the lower level are upgraded first followed by the ones with the higher level.

The architectural model (of the decision and upgrade phases) is appropriate for both modes of operation of an Intelligent Robot, the training mode and the well-trained mode. The training mode of operation is defined as the mode in which the Intelligent Robot "explores" and "learns" its capabilities and alternative actions given the user(s) requested job(s). While in this mode, the probabilities are significantly modifiable by the learning algorithms rewarding certain plans and penalizing others [3]. The well-trained mode of operation follows the training mode and is defined as the mode in which the Intelligent Robot is well trained and knows exactly the sequence of actions necessary to complete a user requested job [6]. The well-trained mode exists only when there are not situations which might include unpredictable events.

This paper describes both models for the higher two levels of Intelligent Robots. Section three of the paper presents the pertinent definitions necessary for the derivation of the models. Sections Four and Five

present the models for the decision-phase of the organization and coordination levels, while section Six explains the architectural model for the upgrade phase of Intelligent Robots. Section Seven presents the advantages of the models.

3. DEFINITIONS

For every Intelligent Robotic System define [4,5]:

1. The set of user commands $C=\{c_1,c_2,\ldots,c_M$; M fixed and finite$\}$ with associated probabilities $p(c_n)$, $n=1,2,..M$, sent to the Intelligent Robot via any remote or not channel.

2. The set of classified compiled input commands $U=\{u_1, u_2,\ldots,u_M$; M fixed and finite$\}$ with associated probabilities $p(u_j/c_n)$, $j=1,2,..M$, which are the inputs to the organization level of the system.

3. The task domain of the Intelligent Robot with the set of independent but not mutually exclusive disjoint sub-sets of non-repetitive and repetitive primitive events $E = \{E_{nr},E_r\} = \{e_1,e_2,\ldots e_{N-L},e_{N-L+1},\ldots e_N$; N fixed and finite$\}$.

4. The binary valued random variable x_i associated with each e_i indicating if e_i is active ($x_i=1$) or inactive ($x_i=0$) given a u_j, with corresponding probabilities $p(x_i=1/u_j)$ and $p(x_i=0/u_j)$ respectively.

5. The set of the (2^N-1) activities which are groups of primitive events concatenated together to define a complex task. They are represented by a string of binary random variables $X_{jm}=(x_1,x_2,\ldots x_n)_m$; $m=1,2,..,(2^N-1)$, which indicates which e_i's are active or inactive within an activity with a probability $P(X_{jm}/u_j)$.

6. The set of compatible ordered activities obtained by ordering the primitive events within each activity and represented by a string of compatible ordered binary random variables Y_{jmr}, where r denotes the rth ordered activity obtained from X_{jm}, with a probability $P(Y_{jmr}/u_j)$.

7. The set of compatible augmented ordered activities obtained by inserting repetitive primitive events within appropriate positions of each Y_{jmr} and represented by $Y_{jmr}(a_s)$, where a_s denotes the sth augmented activity obtained from Y_{jmr} with a probability $P(Y_{jmr}(a_s)/Y_{jmr})$.

8. The set of mask matrices M_{jmr} with associated probabilities $p(M_{jmr}/u_j)$ used to obtain the compatible ordered activities (Y_{jmr}) from the activities (X_{jm}).

9. The set of augmented mask matrices $M_{jmr}(a_s)$ with associated probabilities $p(M_{jmr}(a_s)/Y_{jmr})$ used to obtain the compatible augmented ordered activities from each Y_{jmr}.

When a user command C_n with a probability $p(C_n)$ is sent to the Intelligent Robot it is received and classified by a classifier to yield the classified command u_j; with a probability $p(u_j;/(n)$, which is the input to the organization level.

4. MODEL FOR THE ORGANIZATION LEVEL

The organization level performs five sequential functions as shown in Figure 2: machine reasoning, planning, decision making, feedback and long-term memory exchange. The last two are performed during the upgrade phase. The organizer formulates complete and compatible plans and decides about the best possible plan to execute the user requested job. This is done by associating u_j with a set of pertinent activities X_{jm} with corresponding probabilities $P(X_{jm}/u_j)$ (reasoning), and by organizing the activities in such a way (planning) to yield complete and compatible plans: The compatible ordered activities Y_{jmr} associated probabilities are: $P(Y_{jmr}/u_j = p(M_{jmr}/u_j)*P(X_{jm}/u_j)$. The compatible augmented ordered activities $Y_{jmr}(a_s)$ are obtained by inserting repetitive primitive events in appropriate positions within each Y_{jmr} and their corresponding probabilities are: $P(Y_{jmr}(a_s)/Y_{jmr})=p(M_{jmr}(a_s)/Y_{jmr})*P(Y_{jmr}/u_j)$. Every incompatible activity and incomplete plan is rejected [4]. The most probable complete and compatible plan Y^F is the final plan that is transferred to the coordination level.

After the completion of the requested job upgrade of the probabilities and stored information follows as it

will be explained in section Six. Figure 3 shows the analytical probabilities illustration of the complete organization level function.

The architectural model for the machine reasoning function is shown in Figure 4. The input to the model is the (classified) compiled input command u_j and the corresponding output, Z_j^R, the set of the (maximum) (2^N-1) pertinent activities with the corresponding addresses which store the activity probabilities $P(X_{jm}/u_j)$. The reasoning block RB contains the (2^N-1) strings of binary valued random variables stored at a particular order, which represent the activities associated with <u>any</u> compiled input command. The corresponding addresses which store the probabilities related to these activities are transferred from the memory D^R, a part of the long-term memory of the organizer. The memory D^R consists of M different memory blocks $D_1, D_2, \ldots, D_M$. One memory block, D_j, is associated with each compiled input command u_j. Once the compiled input command u_j has been recognized, realization of the switch s_1 activates (enables) the memory block D_j. Transfer of the data (addresses) contained in D_j is accomplished via the realization of the switch s_2. The switches are coupled with each other. The contents of the RB are transferred to the right most positions of the PRB (Probabilistic reasoning block) while the corresponding addresses which store the pertinent probabilities occupy the left most positions. The information stored in D^R is not modifiable by, or during the machine reasoning function. Therefore, D^R is considered as <u>permanent</u> memory whose values <u>do not change during an interaction cycle</u>, i.e., from the time the user has requested a job until its actual execution. The values of the probability distribution functions associated with the set of pertinent activities are upgraded <u>only after</u> the completion of the requested job through a specific hardware unit described in section Six.

The model for the machine planning function is more complicated. It is shown in Figure 5. The input to the planning model is Z_j^R. The output from the machine planning model is, Z_j^P, the set of all complete and compatible plans capable to the set of all compatible augmented ordered strings of primitive events $Y_{jmr}(a_s)$ formulated during the machine planning function. All compatible ordered activities are stored in the first planning box, PB1. Every compatible augmented ordered activity is stored in the second planning box, PB2. The two compatibility tests (one to obtain compatible ordered activities and one to obtain compatible augmented ordered activities) are performed within the boxes CPT1 and CPT2. The specific hardware unit for both compatibility tests is shown in Figure 5(b). The memory D_T, also a part of the long-term memory of the organizer, is divided into four sub-blocks, D_{T1}, D_{T2}, D_{T3} and D_{T4}: D_{T1} contains all incompatible pairs of the form (non-repetitive primitive, non-repetitive primitive), D_{T2} of the form (non-repetitive primitive, repetitive primitive), D_{T3} of the form (repetitive primitive, non-repetitive primitive) and D_{T4} of the form (repetitive primitive, repetitive primitive). Each ordered activity (augmented ordered activity) is transferred to the register R(R'). At the beginning of the compatibility test the pointer PT (PT') is at the left most position of R (R'). It transfers every pair of primitives to the two-position register R1 (R1'), which scans D_{T1} (D_{T1}, D_{T2}, D_{T3} and D_{T4}) to check if the stored pair is incompatible. If yes, the whole ordered activity (augmented ordered activity) is rejected. If not, the left most primitive event in R1 (R1') is discarded, the right most moves one position to the left and a new primitive event occupies the empty position. The test continues until the pointer has reached the last two primitive events of the activity. Therefore, the number of compatible activities is significantly reduced. The corresponding addresses of the compatible ordered activities with the mask probabilities $p(M_{jmr}/u_j)$ are transferred from the memory D_1^P via the realization of the coupled switches s_3 and s_4. The switch s_3 activates the corresponding memory block D_{jj} (once Z_j^R has been recognized), while the switch s_4 permits the transfer of data. The box PB1 now contains the compatible ordered activities with their corresponding addresses containing the probabilities $P(Y_{jmr}/u_j)$. The insertion of the repetitive primitive events is performed in the box INS while the compatible augmented ordered activities are stored in the second planning box, PB2. Their corresponding memory addresses with the augmented masks probabilities $p(M_{jmr}(a_s)/Y_{jmr})$ are transferred from D_2^P. Activation and transfer of data from the appropriate memory block D_{jjj} is accomplished via the realization of the two coupled switches s_5 and s_5 in a way similar to the ones described before. The information stored in D_1^P and D_2^P is not modifiable by, or during the machine planning function. This information is considered permanent within an iteration cycle, too. The output from the machine planning function Z_j^P is the set of all complete and compatible ordered activities that may execute the requested job. The completeness test is performed within LCMT. This test accepts every meaningful syntactically correct plan [5].

The model for the machine decision making function is shown in Figure 6. All complete and compatible plans Z_j^P are stored in MDMB (machine decision making box) and checked by pairs to find the most probable one. The most probable plan is stored in RR. If during the check, a complete plan with higher probability than the one

already stored in RR is found, it is transferred to RR while the already stored one is discarded. Once the check is over, the contents of RR indicate the most probable complete and compatible plan to execute the requested job.

Every complete and compatible plan is also stored in a particular part of the long-term memory of the organizer, D_{ss}. This memory contains every complete and compatible plan related to every compiled input command as shown in Figure 6. The idea of this particular memory is very important: It represents the situation of a well-trained Intelligent Robot (under the assumption that no unpredictable events may occur). An Intelligent Robot which has reached this mode of operation associates immediately after the recognition of the compiled input command the most probable of the plans (if more than one available) stored in D_{ss}, without going through every single individual function.

5. MODEL FOR THE COORDINATION LEVEL

The coordination level is composed of a specific number of coordinators as shown in Figure 7. Its purposes to coordinate the individual tasks, select the appropriate performance requirements for the execution level, identify space limitations and assign penalty functions, optimize the performance of the overall plan and use learning for performance improvement. It issues specific commands to the execution level which is composed of a number of execution devices associated with the coordinators at the coordination level. The interaction between the three levels is represented in terms of on-line (real-time) and off-line feedback information. The on-line feedback information is communicated from the execution to the coordination level during the execution of the requested job and is used to evaluate and upgrade the information stored in the long-term memory of the organization level (functions of feedback and long-term memory exchange).

A block diagram of the coordination level architectural model is shown in Figure 8. The complete and compatible plan Y^F stored in RR is transferred to a buffer B. The complete and compatible most probable (final) plan, contains actually a sequence of addresses. Each address corresponds to a primitive event (repetitive or non-repetitive). When the final plan is loaded into the buffer the BEVENT line is activated and the pointer PTl scans the buffer once from left to right. With the aid of the qualifier QLFR it is known how many times each coordinator will be accessed, as well as when it will be accessed. Thus, the association execution devices are activated when their coordinator is accessed via the qualifier QLFR. After the first scanning, the pointer PTl returns to the left most position. The BEVENT line is deactivated and the qualifier is used to activate another coordinator as the pointer moves from left to right. Upon completion of its operation a BENT signal is sent to the qualifier and the buffer and the pointer moves one position to the right. When a specific coordinator completes its specific functions and the on-line feedback information has (already) been communicated to it, this information is stored in the short-term memory of the coordination level. It may be used by other coordinators to complete their functions (during the execution of a requested job) and/or to calculate the overall accrued cost associated with the coordination level, that will be communicated to the organizer after the execution of the job. Thus, the different coordinators do not communicate directly with each other. But each coordinator has access to the short-term memory for storage and retrieval of data.

The coordination level does not have a particular memory equivalent to the D_{ss} memory of the organization level. This is because the workspace environment of the coordination level is dynamic: Given a final plan Y^F, the formulation of the actual control problem regarding the way of its execution depends not only on previous experience but also on the current configuration of the workspace environment. For example, previously chosen trajectories for the different motions of the manipulator(s) should be modified by the presence of more or less obstacles and/or of additional objects.

The execution level performs the commands issued by the coordination level. Each control problem is analyzed in the light of its special requirements. Therefore, there is not one general architectural model that can include every possible operation performed at the execution level.

But although this is the case, the execution level consists of a number of devices each associated with a specific coordinator and vice versa. Each device is accessed via a command issued by its coordinator. Hence, the hierarchical structure is preserved.

6. MODELS FOR THE UPGRADE PHASE

The architectural model for the upgrade phase of Intelligent Robots includes specific hardware units that must be built within each of the higher two levels to account for both types of feedback information, on-line and off-line feedback.

The off-line feedback mechanism (from the coordination to the organization level) is activated after the execution of the requested job and is used to upgrade the probabilities distribution functions of the sets of pertinent, ordered and augmented ordered activities associated with a compiled input command.

The probabilities upgrade algorithm is given by the equation:

$$p(t+1/u_j) = p(t/u_j)+\beta_{t+1}[\xi-p(t/u_j)] \qquad (1)$$

$$\xi = \begin{cases} 1; & j = j_{min} \\ 0 & \text{otherwise} \end{cases}$$

where $p(t+1/u_j)$ denotes the corresponding probability at iteration cycle (t+1), J the actual cost of executionof the job, J_{min} the minimum cost of execution and β a coefficient that obeys Dvoketsky's conditions [3]. Each upgrade requires one addition, one subtraction and one multiplication. During this process there are no data dependencies between probabilities. A special purpose hardware unit is the one shown in Figure 9. It consists of two adders, one multiplicator unit and three registers [7]. The probability value is fetched from the memory, passes through the three arithmetic operations and is returned to the memory, overwriting the previous value. This function is performed after the execution of the requested job and calculation of the overall accrued cost associated with the coordination level.

Five such hardware units must be built in the organization level to upgrade all pertinent pdfs as shown in Figure 10.

The on-line feedback mechanism (from the execution to the coordination level) is activated during the execution of the requested job. A block diagram of this mechanism is shown in Figure 11. Solid lines represent the on-line feedback information from the execution devices (at the execution level) to the different coordinators. Individual and accrued costs are calculated within each coordinator and are transferred to the short-term memory of the coordination level where the overall accrued cost associated with the coordination level is calculated and communicated back to the organizer after the execution of the job. The dotted lines illustrate how the information from the short-term memory is used by the different coordinators and their execution devices for the completion of the job.

7. REMARKS

The models presented here have two major advantages:

1. They are applicable to any Intelligent Machine operating under the constraints of Hierarchically Intelligent Control Systems, and,

2. The same architecture may be used if one uses a linguistic approach to design Intelligent Machines. Based on the models described, one may derive the context-free grammars that satisfy the same performance requirements and criteria.

The architectural model of the organization level may be also used if one wishes to modify the machine reasoning function in the algorithm of the organization level as follows: Given a compiled input command u_j, associated with it only a subset of the (2^N-1) pertinent activities which contains only those activities with corresponding probabilities greater than a prespecified $P_{min}(u_j)$. Therefore, the machine planning and decision making functions are limited to the formulation of complete and compatible plans that originate from this subset of the (2^N-1) possible pertinent activities. This approach requires decision making from the beginning: the machine reasoning function because it eliminates every activity with corresponcing probability less than $P_{min}(u_j)$.

The disadvantage of the architectural model is that it does not consider the possibility of unpredictable events during plan(s) execution.

REFERENCES

[1] Saridis, G. N., Valavanis, K. P., "Information Theoretic Approach for Knowledge Engineering and Intelligent Machines", Proceedings of the ACC Conference, Boston, MA, June 1985.

[2] Valavanis, K. P., Saridis, G. N., "Analytical Design of Intelligent Machines". Proceedings of the SYROCO Conference, Spain, November 6-8, 1985.

[3] Saridis, G. N., Self-Organizing Control of Stochastic Systems, NY, Marcel Dekker, 1977.

[4] Saridis, G. N., Valavanis, K. P., "A Mathematical Model for the Design of Intelligent Machines", submitted for publication.

[5] Valavanis, K. P., "A Mathematical Formulation for the Analytical Design of Intelligent Machines", Ph.D. Thesis, RAL #85, 1986.

[6] Boettcher, K. L., "An Information Theoretic Model of the Decision Maker", M. S. Thesis, LIDS-TH-1096, MIT, Cambridge, MA, June 1981.

[7] Graham, J. H., Saridis, G. N., "Linguistic Methods for Hierarchically Intelligent Control", School of Electrical Engineering, Purdue University, TR-EE 80-34, October 1980.

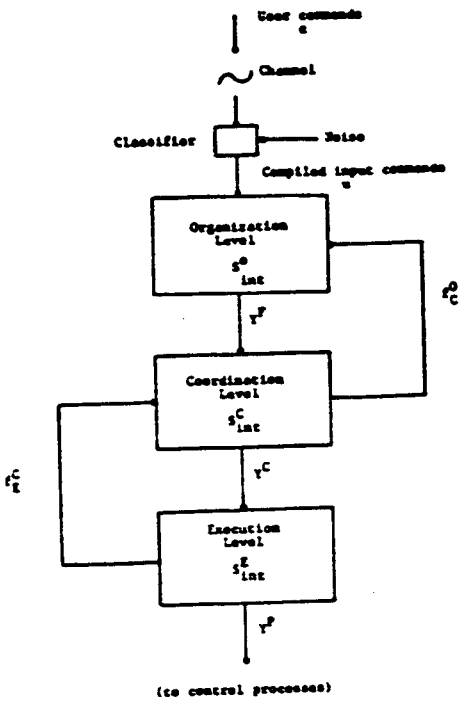


Figure 1. Block diagram of an Intelligent Machine

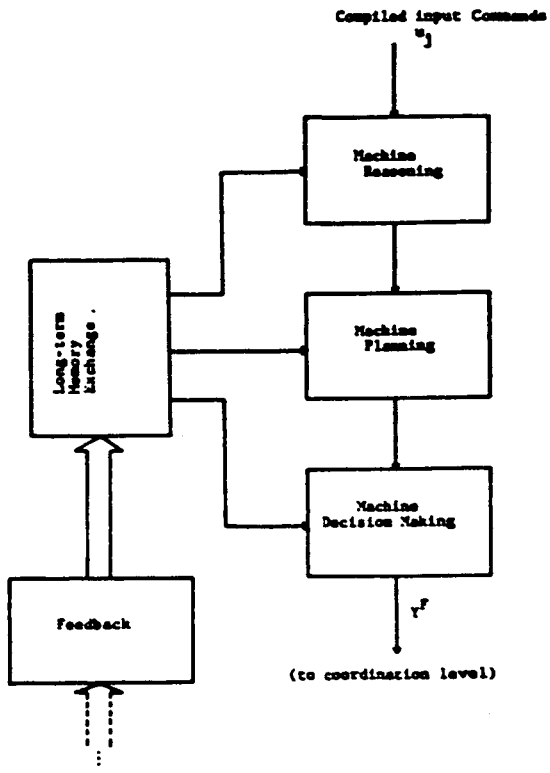


Figure 2. Block diagram of the organization level

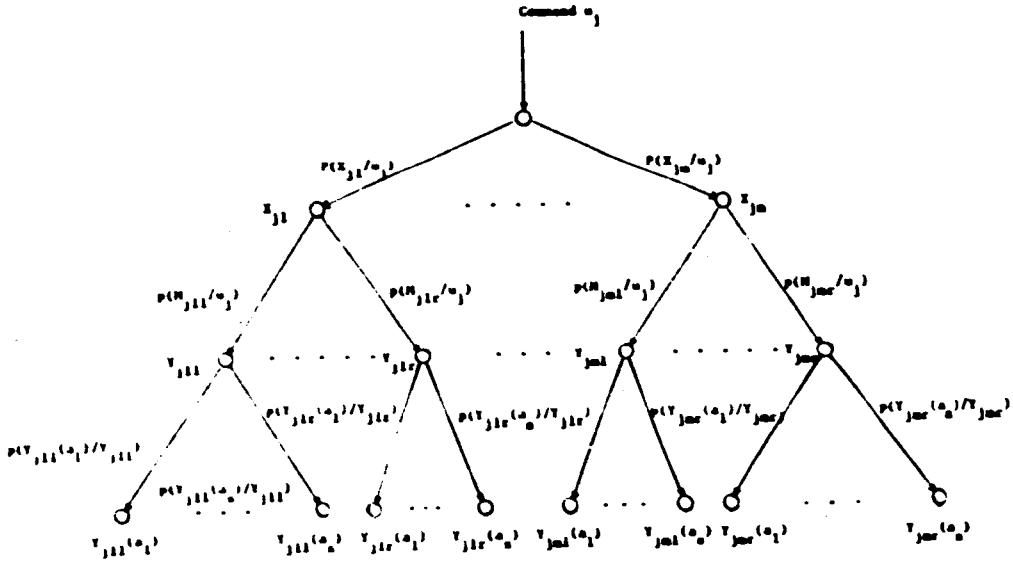


Figure 3a. Analytical illustration of the function of the organization level

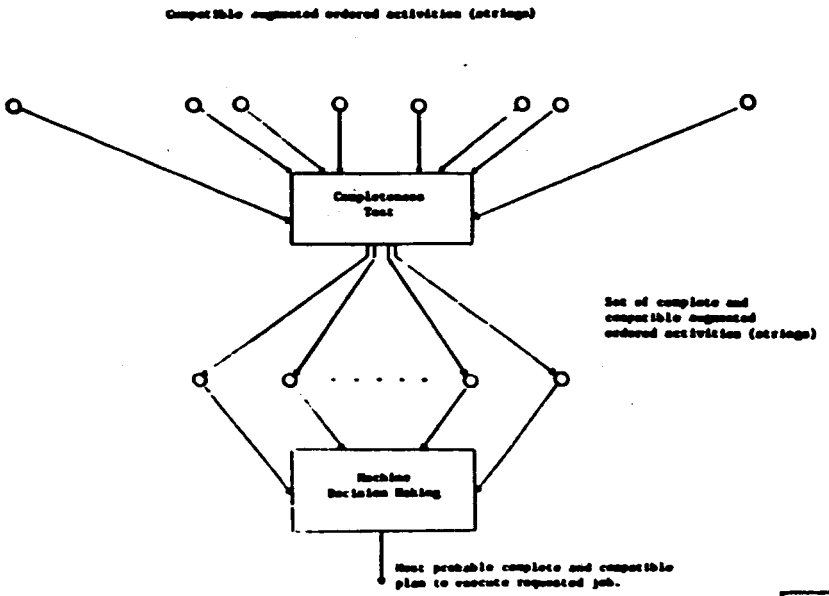


Figure 3b.

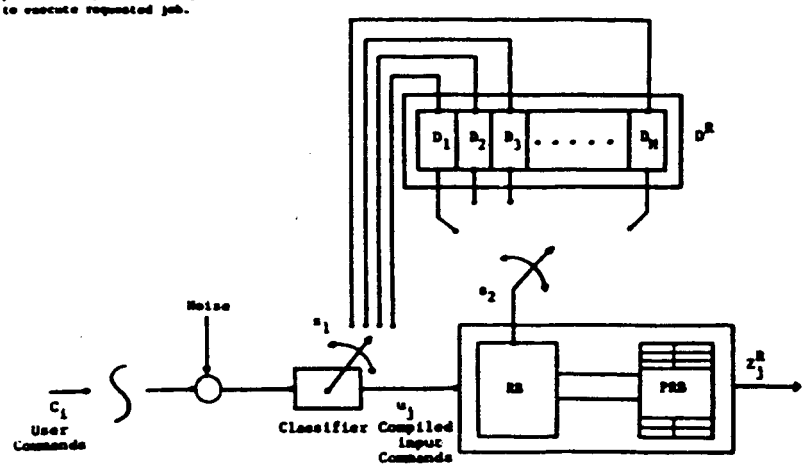


Figure 4. The Model for the Machine Reasoning Function

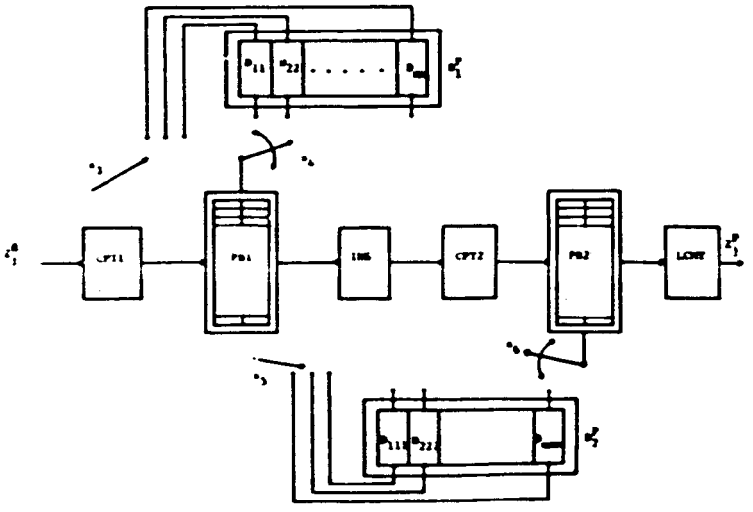


Figure 5a. The Model for the Machine Planning Function

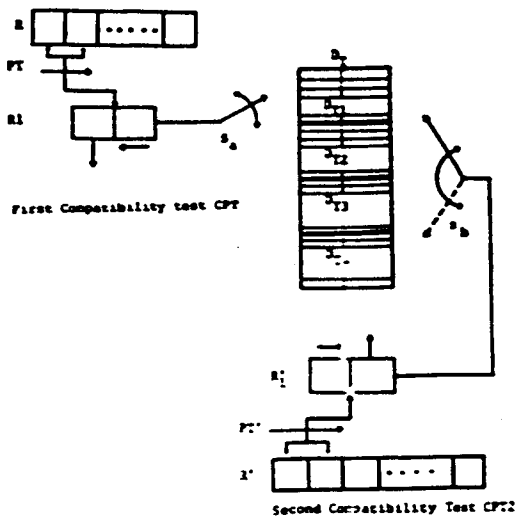


Figure 5b.

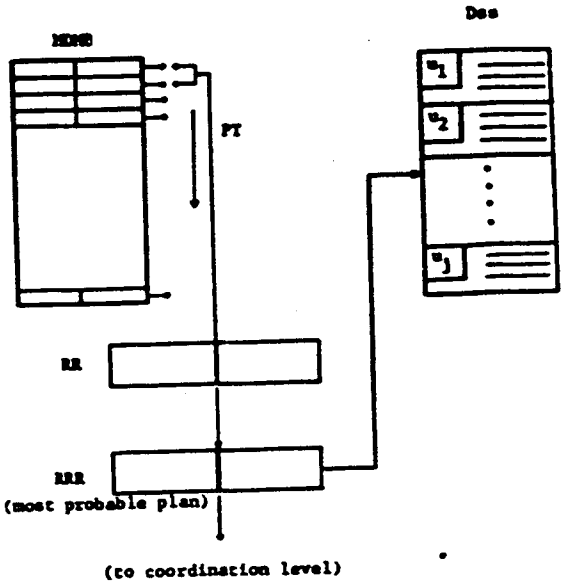


Figure 6. The Model for the Machine Decision Making Function

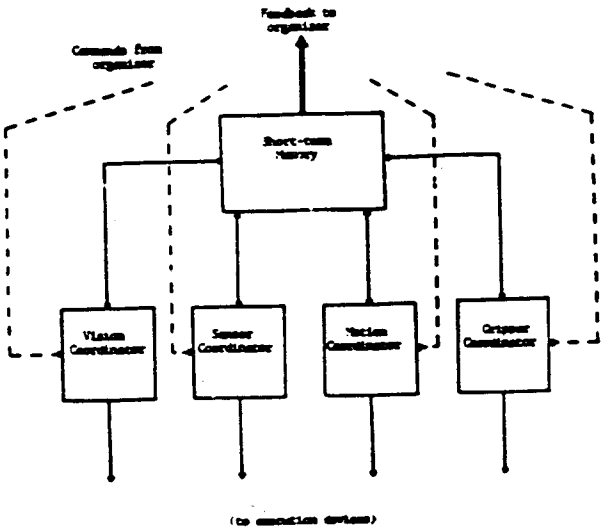


Figure 7 Block diagram of the Coordination Level

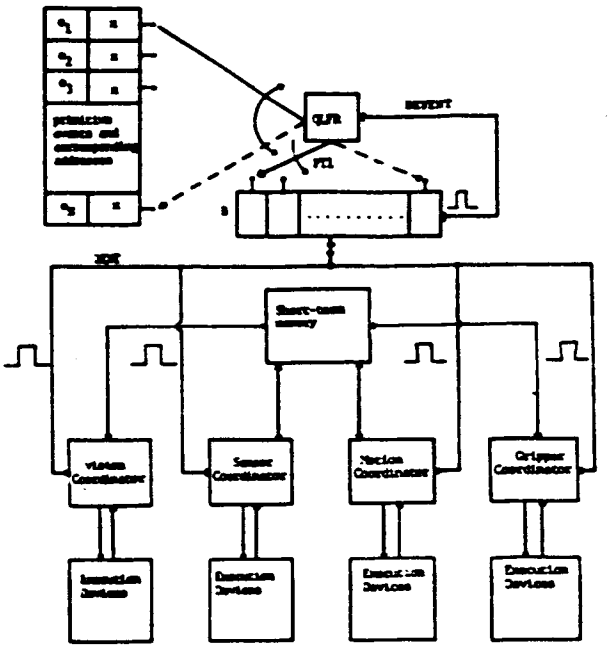
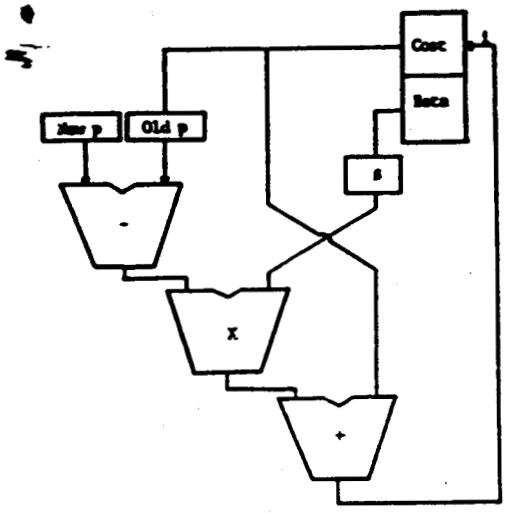


Figure 8 The Model of the Coordination Level



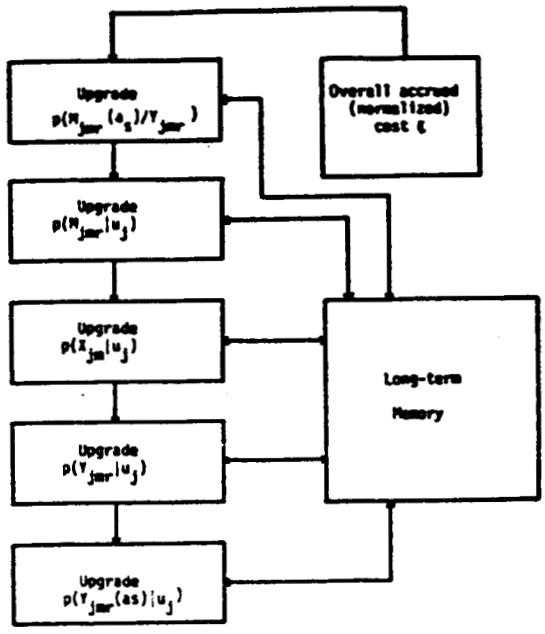


Figure 9 Probability Upgrade Hardware Unit

Figure 10. The Model of the Off-line Feedback Mechanism

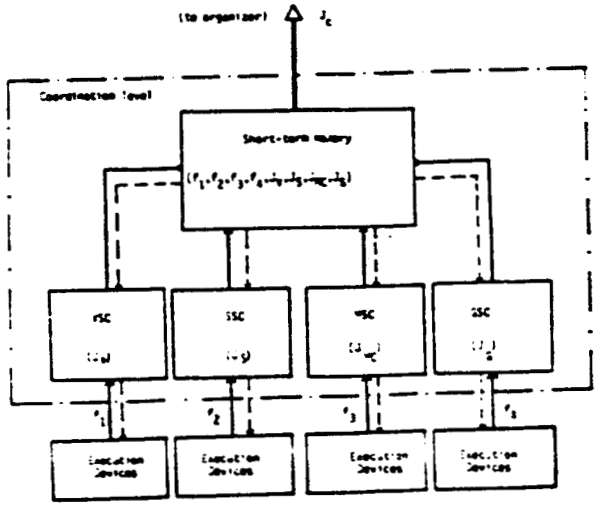


Figure 11. The Architectural Model for the On-line Feedback Mechanism