# A CONCEPTUAL FRAMEWORK FOR INTELLIGENT REAL TIME INFORMATION PROCESSING

Robert Schudy

BBN Laboratories Incorporated
10 Moulton Street
Cambridge, Massachusetts 02238

## ABSTRACT

By combining artificial intelligent concepts with the human information processing model of Rasmussen, we have developed a conceptual framework for real-time AI systems which provides a foundation for system organization, control and validation. The approach is based on the description of system processing in terms of an abstraction hierarchy of states of knowledge and processing functions which connect those states of knowledge. The states of knowledge are organized along one dimension which corresponds to the extent to which the concepts are expressed in terms of the system inouts or in terms of the system response. Thus organized, the useful states form a generally triangular shape with the sensors and effectors forming the lower two vertices and the full evaluated set of courses of action the apex. If the representations and processing steps in the slopes of this pyramid are correct and complete, them the processing sequence from inputs to outputs following the slopes of this pyramid results in correct behavior. Unfortunately, this path is generally too computationally expensive to be performed in real time, either by natural or artificially intelligent systems. Within the boundaries of the triangle are numerous processing paths which shortcut the detailed processing, by connecting incomplete levels of analysis to partially defined responses. Shortcuts at different levels of abstraction include reflexes, sensory-motor control, rule-based behavior, and satisficing. The correctness of shortcuts depends on whether the response inferred on the processing shortcut is consistent with the responses which would have been inferred by the computations which are being shortcut. By clarifying assumptions, relationships, and the knowledge and processing which is being approximated, this approach provides a foundation for knowledge acquisition, system design, and system validation. We have used this approach in the design of a real-time tactical decision aiding system, in defining the requirements for an intelligent aiding system for transport pilots, and in the design of an autonomous system.

Figure 1 schematically describes the overall processing steps for a real-time intelligent agent such as a pilot, an autonomous system, or a real-time aiding system. The vertical, abstraction, axis in the figure corresponds to the extent to which the processing is removed from the concrete sensors and effectors. Movement along the horizontal axis represents progression of the processing from the sensors to the effectors. The ovals in the figure represent states of knowledge, and the labelled arcs between those states represent processing which implements the indicated transitions between the states of knowledge. Different knowledge representation requirements are associated with the different states of knowledge, and different computational requirements are associated with the different classes of transitions. The particular states of knowledge and processing steps should not be interpreted as required functional partitions for particular systems. The actual flow of processing in nontrivial systems involves many more stages than are shown, and results will generally be used by several processing levels to the right of where they are produced. Many more pathways may exist between the indicated states of knowledge, and other states of knowledge may be important for some applications; these other pathways are summarized by the three classes of shortcuts listed within the three large arrows.

## 1. PROCESSING WITHOUT SHORTCUTS

The discussion in the next 21 paragraphs follows the information flow in Figure 1 from sensors to effectors, taking no shortcuts, and alternating between states of knowledge and the processing steps which accomplish the transformations between those states.

*Sensors.* The sensor state of knowledge is just the sensor data and other inputs to the intelligent agent. This input data includes "self" information from sensors such as kinesthetic sensors (e.g. actuator positions, velocities, stresses, and strains), system status data (e.g. self-test and built-in-test results), and internal environmental sensor data (e.g. temperatures, pressures, accelerations, flow rates, fuel quantity, vibration, and power supply voltages). This input data also includes environmental information communicated from other agents.

*Perception.* Perception is the mapping of the sensor data into symbolic descriptions of the entities and states of relevance to the intelligent agent. This signal-to-symbol processing includes of all vision processing up to and including the level of scene description, and all diagnostic processing up to the level of system status.

*Entity and State Descriptions.* The output of perceptual processing is a symbolic description of the state of the agent and its environment. These descriptions could be represented in terms of instances of archetypical entities and states, with the attributes refined from the sensor data. The state descriptions include relationships between entities, when those relationships can be derived from the input data.

*Assessment.* Assessment includes the processing stages in which the significance of the current situation is derived, in terms of the impact of the situation on the agent, its goals, plans, and actions. Assessment processing proceeds

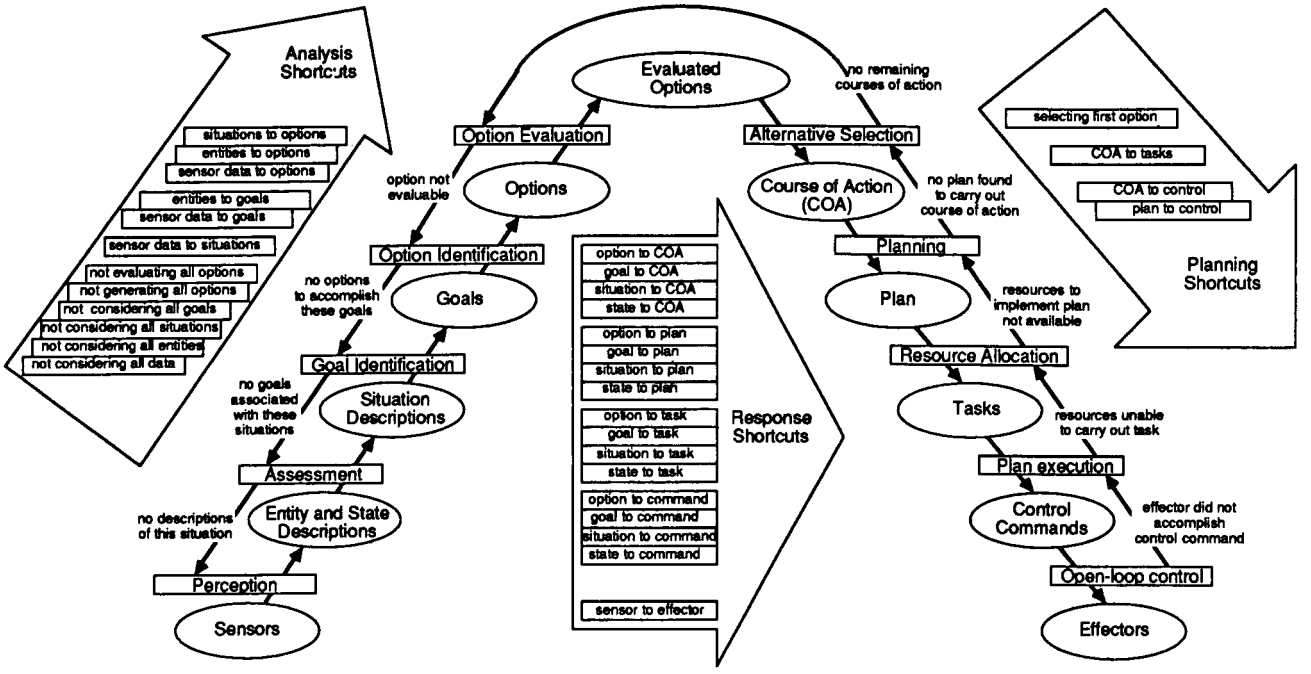


Figure 1. General Information Processing Model with Shortcuts

through the generation of successively more abstract descriptions of the situation attributes and of the situation as a whole.

*Situation Descriptions.* Situation descriptions are a proper superset of entity and state descriptions. Because of the scope of situation descriptions, it may no longer be feasible to describe situations in terms of single situation types. Thus situation descriptions may take the form of interrelated instances of different situation types, each of which describes some aspect of the total situation.

*Goal Identification.* Goals Direct and focus behavior. Thus while goals may enter at any level before this stage, they must be defined by the end of this stage, so that the goals can be used to drive the option identification step. While very general static goals such as survival may be useful in strategic planning, more specific and more near-term situation-dependent goals are more effective in driving behavior. Goal identification proceeds from assessments of the situation and from more global goals contained in the plan structure. Since different aspects of the situation may dictate different goals, resolution of goal conflicts may be required.

*Goals.* Goals combine descriptions of future states and situations and statements of the desirability of those states and situations. Thus the goal representation language is a proper superset of the situation representation language. Since goals involve not only states, but also situations, actions, and the temporal and other relationships between states and situations, the representation for goals shares many of the characteristics of plans.

*Option Identification.* With descriptions of the state, situation, and goals, feasible behavioral options can now be generated. For the overall processing to be optimal, in the sense that the overall behavior is the best for attaining the goals, this processing step must identify all options which may be optional; however not all options identified need be optimal.

*Options.* Options are descriptions of potential courses of action for the intelligent agent. Option descriptions may include state and entity descriptions, situation descriptions, and goals. Thus the option representation language is a proper superset of the goal representation language.

*Option Evaluation.* The option evaluation step is the detailed application of the goal criteria against the identified options. If the options are parametric, then evaluation can involve generating the members of the sets represented by the parametric options. This can in principle produce any number of options to be evaluated. Unless the optimality criteria impose a total ordering on the courses of action, this set may not be finite. The technologies of heuristic search differential game theory, and operations research can each address option evaluation problems in different domains, but no efficient general methods for option evaluation are currently known.

*Evaluated Options.* Unless powerful abstractions, heuristics, and other means are available for representing and pruning the space of options, usually only a small set of the whole family of options can be evaluated. The evaluated option set may include all of the representations in the option representation, plus information, such as total or partial orderings, and evaluative criteria, resulting from the evaluation of those options. In order that the behavior be optimal, all attributes of evaluative interest must be captured in the representation of the options. Thus subsequent representations may be restrictions of this representation of options, and can only enlarge on the evaluated options representation by expansion of abstractions contained in the evaluated options, and only then when the evaluation of those options does not depend on the expanded detail.

*Alternative Selection.* The alternative selection process may be trivial, if the goals impose a total ordering on the options, or it may involve the invocation of additional criteria to select amongst an equivalence class of best options.

*Course of Action.* The course of action is a subset of the evaluated options. If the agent is operating in an uncertain

environment involving unpredictable agents, or other sources of uncertainty, then the course of action may include branches to accommodate the generically different responses during plan execution. Thus the course of action, like the options, evaluated options, plans, and tasks, may be represented as a semilattice, with a root representing the current state, and nodes representing branches and joins of tasks and situations during possible plan executions. The course of action may still involve free parameters, or set-valued parameters, if the values of those parameters was not required to evaluate the option.

*Planning.* The planning process is the expansion of the course of action to the level of the operations to be performed. Note that many of the computational steps which would be considered as steps in planning, such as option identification and evaluation, may have already been performed to support option evaluation and the selection of the course of action.

*Plan.* The plan representation may include entity and state representations, situation representations, decision criteria, and alternative options. From a representational standpoint it is thus equivalent to the evaluated options representation. The plan differs from that representation in two ways. First, it is uniformly expanded to the level necessary to support resource allocation processing; it thus must include all parallel operations, and detailed models of timing. Secondly, the plan includes only those options which passed the alternative selection process. The plan is thus more detailed than the course of action, but includes fewer options than the evaluated options.

*Resource Allocation.* Resource allocation binds specific resources to specific actions in the plan. Some resource conflicts may have been resolved earlier in the processing to evaluate the course of action and to develop the plan; the resource allocation process completes those allocations. Resource allocation processing completes the development of task descriptions to support plan execution.

*Tasks.* The task representation is the plan representation expanded to the level of specific tasks for specific resources. Task descriptions include any parameters, other than inputs, required by the execution procedures. These task descriptions include sensing tasks, and decision tasks, information processing tasks. It may also include enabling or initiating processing shortcuts such as servo control processes.

*Task Execution.* Task execution follows the procedure for tasks of that type. This involves task initiation, performance monitoring, and parameter adjustment. Task execution can be influenced by the results from decision tasks, and by feedback from other running tasks.

*Control Commands.* The output of the task execution processing are control commands to the effectors and other resources. These control commands include all information other than input data necessary to determine the operation of those resources. The specific data depends on the resource controlled.

*Control.* Control processing may involve servo-loops, or can operate with limited or no feedback.

*Effectors.* Effectors include all resources under control of the agent, including information processing, actuators, and sensors.

If all of the processes and representations described above are complete and correct, and they are to run completion, then no feedback is required for the system to produce correct behavior. However, efficiency can be improved by implementing feedback from later processing stages to control processing at earlier stages. This cascading can take the form of the integration of later functions into earlier functions, where it can help narrow the number of options considered, or it can take the form of specific information fed back from later stages when those stages have examined partial results those earlier stages. An example of the processing of the first sort is the integration of resource allocation constraints into the option generation and evaluation stages; an example of the second sort is feedback from the alternative selection function about partially evaluated options.

# 2. SHORTCUT PROCESSING

Figure 1 shows three broad classes of shortcuts - those of analysis (intent formation), those of intent execution, and a class of "response" shortcuts which move at various levels of abstraction from left to right, bridging from partial analyses to skeletal responses. We currently believe that response shortcuts are a predominant pathway in human information processing. The response shortcuts range in abstraction from simple servo-control laws linking sensors and effectors, up through satisficing [Simon, 1969], which shortcuts exploring all of the courses of action. Shortcuts at an intermediate level of abstraction are particularly important in real-time decision making. Situation-response shortcuts, which move rightward from the vicinity of the *situation* description level of analysis to the vicinity of the *tasks* level of execution, are described in the companion paper *A Situation-Response Model for Intelligent Pilot Aiding.*

Situation-response processing has a role in the less abstract skill-based behavior. The pathways in Figure 1 for skill-based shortcuts proceed nearly horizontally from the vicinity of sensing and perception to the vicinity of the effectors. The establishment of such skill-based shortcuts reduces workload and reduces processing delay by uncoupling the situation assessment process from the process of adapting to changing situational parameters. In our model, the activation and management of skill-based behavior is one of the normal functions of rule-based behavior. The situation assessment function then assumes the role of enabling the execution of the skill-based behavior.

Both the risks and speed of such shortcut processing are well known. Interviews with airline pilots revealed the following example: During aircraft takeoff roll, the pilot-flying noticed a fluctuation in right engine readings. Just before takeoff velocity the pilot-flying heard a loud boom and felt the plane vibrate. Reflexively he reached to shut down the right engine (having mentally established its potential for failure). The pilot-not-flying stopped this move because he had determined that it was the left engine that had failed. Thus, the processing shortcut (enabled by the assessment the situation as a possible-right-engine-problem) resulted in a rapid, but inappropriate response.

Situation-response processing also has an important role in the more abstract knowledge-based behavior. The normal pattern is that when the limits of situation-response processing is reached, knowledge-based reasoning is initiated. Then either the situation changes while the knowledge-based processing takes place, or the knowledge-based processing produces a useful result. In either case the processing reverts to the situation-response model. Note that in this model situation-response processing serves as an input filter for knowledge-based processing, guaranteeing that the scarce knowledge-based processing resources are only invested in unusual and hence presumably fruitful problems.

## 3. CONCLUSIONS

Real-time performance in natural or artificially intelligent systems depends on using the lowest feasible levels of abstraction. Processes operating at different levels of abstraction can be organized so that correct behavior can be implemented using information processing shortcuts. The shortcut and other levels of processing can be organized so that less abstract processes develop abstractions for more abstract processing, and more abstract processing supervises the less abstract. The development of shortcuts in the information processing abstraction hierarchy is a basic mechanism for learning to perform more quickly.

## 4. References


Simon, H.S., The Sciences of the Artificial, M.I.T. Press, 1969.

Rasmussen, J., Skills, Rules, Knowledge; Signals, Signs and Symbols; and other Distinctions in Human Performance Models. IEEE Transactions of Systems, Man, and Cybernetics SMC-13 (3), 1983.

Rasmussen, J., Strategies for State Identification and Diagnosis in Supervisory Control Tasks, and Design for Computer-based Support Systems. In (Rouse) Advances in Man-Machine Systems Research, Vol. 1 pp. 139-193, 1984.