Understanding Control at the Knowledge Level

B. Chandrasekaran
Laboratory for AI Research
The Ohio State University
Columbus, OH 43210
Email: chandra@cis.ohio-state.edu

Abstract

What is it that unifies the control task in all its manifestations, from the thermostat to the operator of a nuclear power plant? At the same time, how do we explain the variety of the solutions that we see for the task? I propose a Knowledge Level analysis of the task which leads to a task-structure for control. Differences in availability of knowledge, the degree of compilation in the knowledge to map from observations to actions, and properties required of the solutions together determine the differences in the solution architectures. I end by discussing a number of heuristics that arise out of the Knowledge Level analysis that can help in the design of systems to control the physical world.

What is the Knowledge Level?

By now most of us in AI know about the Knowledge Level proposal of Newell [Newell, 81]. It is a way of explaining and predicting the behavior of a decision-making agent without committing oneself to a description of the mechanisms of implementation. The idea is to attribute to the agent a goal or set of goals and knowledge which together would explain its behavior, assuming that the agent is abstractly a rational agent, i.e., one that would apply an item of relevant knowledge to the achievement of a goal. Imagine the following conversation between a guest and a host at a house party:

G: Why does your cat keep going into the kitchen again and again?

H: Oh, it thinks that the food bowl is still in the kitchen. It doesn't know I just moved it to the porch.

The host attributes to the cat the goal of satisfying its hunger, and explains its behavior by positing a (mistaken) piece of knowledge that the food is in the kitchen. That the cat would naturally go to the kitchen under these conditions seems reasonable to the host and presumably to the guest. When people talk this way, they are not asserting that the neural stuff of the cat is some kind of a logical inference machine working on Predicate Calculus expressions. It is simply a useful way of setting up a model of agents and using the model for explaining their behavior. The attributions of the goal and knowledge can be changed on the basis of further empirical evidence, but the revised model would still be in the same language of goals and knowledge.

The Knowledge Level still needs a representation, but this is a representation that is not posited *in* the agent, but one in which outsiders talk *about* the agent. Newell thought that logic was an appropriate representation for this purpose, leaving open the possibility of other languages also being appropriate in some circumstances. Newell used the phrase "Symbol Level," to refer to the representational languages actually used to implement artificial decision agents (or explain the implementation of natural agents). Logic-based languages, Lisp and FORTRAN, neural net descriptions, and even Brook's subsumption architectures are all possible Symbol-Level implementations for a given Knowledge Level specification of an agent.

The Control Problem at the Knowledge Level

Consider the following devices and control agents: the thermostat, the speed regulator for an engine, an animal controlling its body during some motion, the operator of a nuclear power plant, the president and his economic advisors during the task of policy formulation to control inflation and unemployment, and a major corporation planning to control its rapid loss of market share. All these systems are engaged in a "control" task, but seem to use rather different control techniques and architectures. Is this similarity in high-level description just a consequence of an informal use of words in our natural language, or is it an indication of some important structural similarity that can have useful technical consequences? Formulating the control problem at the Knowledge Level can help us to see what makes these problems similar, and, at the same time, to explain the widely divergent implementations of the systems that I listed. Fig. 1 is a brief description of the control problem at the Knowledge Level.

> Control Agent *C*,
> System to be controlled *S*, state vector *s*, goal state *G*, defined as a wff of predicates over components of *s*,
> Observations *O*,
> Action repertoire *A*,
> *The Task*: Synthesize action sequence from *A* such that *S* reaches *G*, subject to various performance constraints (time, error, cost, robustness, stability, etc.)
>
> In fact, control in this sense includes a good part of the general problem of intelligence.

Fig. 1. The control task at the Knowledge Level

In order to see how different control tasks differ in their nature, thus permitting different types of solutions, we need to posit a *task structure* [Chandrasekaran, et al, 1992] for it. The task structure is a decomposition of the task into a set of subtasks, and is one plausible way to accomplish the goals specified in the task description. Fig. 2 describes a task structure for control.

Basically, the task involves two important subtasks: using *O*, model *S* and generate a control response based on the model. Both of these tasks could use as their subtask a *prediction* component. Typically, the modeling task would use prediction to generate consequences of the hypothesized model and check against reality to verify the hypothesis. The planning component would use prediction to check whether the plan would generate the intended behavior given the model.

- Common subtasks:
  Build a model of *S* using *O*
  (The general version of the problem is abductive explanation: from perception to scientific theory formation are examples of this.) The task might involve prediction as a subtask.
  » Create a proposed plan to move *S* to *G*
  (The general version of the problem is one of synthesis of plans.)
  » Predict behavior of *S* under plan using model
  (In general, simulation to analysis may be invoked.)
  » Modify plan

Fig. 2. The task-structure of control

Every control system need not do each of the tasks in Fig. 2 *explicitly*. It is hard to build effective control systems which do not use some sort of observation to sense the environment, or which do not make the generation of the control signal or policy depend on the observation. Many control systems can be interpreted as performing these tasks implicitly. The prediction subtask may actually be skipped altogether in certain domains.

The variety of solutions to the control problem arises from the different assumptions and requirements under which different types of solutions are adopted for the subtasks, resulting in different properties for the control solution as a whole.

As I mentioned, at one extreme, the subtasks may be done implicitly, or in a limited way that they only work under certain conditions. At the other extreme, they may also be done with explicit problem solving by multiple agents searching in complex problem spaces. And of course there are solutions of varying complexity in between.

The Thermostat, the Physician and the Neural Net Controller

Consider a thermostat (*C* in Fig. 1) For this system, S is the room, s consists of a single state variable, the room temperature, *G*, the goal state of *S*, is the desired temperature range, *O* is the sensing of the temperature by the bimetallic strip, and *A* consists of actions to turn on and off the furnace, the air-conditioner, the fan, etc.

The modeling subtask is solved by directly measuring the variable implicated in the goal predicate. The model of the environment is simply the value of the single state variable, the temperature, and that in turn is a direct function of the curvature of the bimetallic strip.

The curvature of the strip also directly determines when the furnace will be turned on and off. Thus the control generation subtask is solved in the thermostat by using a direct relation between the model value and the action. The two subtasks, and the task as a whole, are thus implemented by a direct mapping between the observation and the action.

Because of the extreme simplicity of the way the subtasks are solved, the prediction task, which is normally a subtask of the modeling and planning tasks, is skipped in the thermostat.

The control architecture of the thermostat is economical and analysis of its behavior is tractable. But there is also a price to pay for this simplicity. Suppose the measurement of temperature by the bimetallic strip is off by 5 deg. The control system will systematically malfunction. A similar problem can be imagined for the control generation component. A larger control system consisting of a human problem solver (or an automated diagnostic system) in the loop may be able to diagnose the problem and adjust the control behavior. This approach increases the robustness of the architecture, but at the cost of increased complexity of the modeling subtask.

Now consider the task facing a physician (*C*): controlling a patient's body (*S*). Various symptoms and diagnostic data constitute the set *O*. The therapeutic options available constitute the set *A*. The goal state is specified by a set of predicates over important body parameters, such as the temperature, liver function, heart rate, etc.

Consider the model-making subtask. This is the familiar diagnostic task. In some instances this problem can be quite complex, involving abductive explanation building, prediction, and so on. This process is modeled as problem space search. The task of generating therapies is usually not as complex, but could involve plan instantiation and prediction, again tasks that are best modeled as search in problem spaces.

Why can't the two subtasks, modeling and planning, be handled by the direct mapping techniques that seem to be so successful in the case of the thermostat? To start off, the number of state variables in the model is quite large, and the relation between observations and the model variables is not as direct in this domain. It is practically impossible to so instrument the body that every relevant variable in the model can be observed directly. With respect to planning, the complexity of a control system that maps directly from symptoms to therapies – some sort of a huge table look-up – would be quite large. It is much more advantageous to map the observations to equivalence classes – the diagnostic categories – and then index the planning actions to these equivalence classes. But doing all of this takes the physician far away from the strategies appropriate for the thermostat.

As a point intermediate in the spectrum between the thermostat, which is a kind of *reflex control system*, and the physician, who is a deliberative search-based problem solver, consider control systems based on PDP-like (or other types of) neural networks. *O* provides the inputs to the neural net, and the output of the net should be composed from the elements of the set *A*. Neural networks can be thought of as systems that select, by using parallel techniques, a path to one of the output nodes that is appropriate for the given input. The activity of even multiply layered NN's can still be modeled as a selection of such paths in parallel in a hierarchy of pre-enumerated and organized search spaces. This kind of connection finding in a limited space of possibilities is why these networks are also often called *associative*. During a cycle of its activity, the net finds a connection between actual observations and appropriate actions. This behavior needs to be contrasted with a model of deliberation such as Soar [Laird, et al, 1987] in which the essence of deliberation is that traversing a problem space and establishing connections between problem spaces are themselves subject to open-ended additional problem solving at run time.

The three models that we have considered so far – the thermostat, neural net controllers, and deliberative problem search controllers – can be compared along different dimensions as follows.

	Speed	Robustness	tractability*
Reflex	fast	low	easy

NN's	medium	medium	medium
Delib. engines	slow	potentially high**	low

*: tractability of analysis
**: depending on availability of knowledge

Table 1: Tradeoff between different kinds of control systems along different dimensions

By robustness in Table 1 I mean the range of conditions under which the control system would perform correctly. The thermostat is unable to handle the situation where the assumption about the relation between the curvature of the strip and the temperature was incorrect. Given a particular body of knowledge, deliberation can in principle use the deductive closure of the knowledge base in determining the control action, while the other two types of control systems in the Table use only knowledge within a fixed length of connection chaining. Of course, any specific implementation of problem space search may not use the full power of deductive closure, or the deductive closure of the knowledge available may be unable in principle to handle a given new situation.

The control systems in Table 1 are simply three samples in a large set of possibilities, but selected because of their prominence in biological control models. The reflex and the NN models more commonly used in the discussion of animal behavior and human motor control behavior, while deliberation is generally restricted to human control behavior where problem solving plays a major role. Engineering of control systems does not need to be restricted to these three families. Other choices can be made in the space of possibilities, reflecting different degrees of search and compilation.

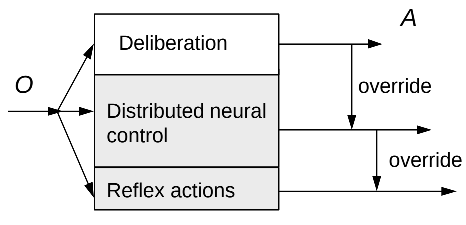
Sources of Power

I have been involved, over the last several years, in the construction of AI-based process control systems and also in research on causal understanding of devices. I have also followed the major trends in both control systems theory – much of which carried on in a mathematical tradition that different from the one that is prevalent in AI – and attempts to understand biological control. My own research has been motivated by trying to understand some of the pragmatics of human reasoning in prediction, causal understanding and real-time control. I have catalogued – and I will be discussing in the rest of the paper – a set of heuristics that I characterize as *sources of power* that biological control systems use. These ideas can also be used in the design of practical control systems, i.e., they are not intended just as explanations of biological control behavior. I do not intend them to be an exhaustive list, but as examples of heuristics that may be obtained by studying the phenomenon of control at an abstract level. These heuristics do not depend on what implementation approaches are used in the actual design – be they symbolic, connectionist networks or fuzzy sets.

Integrating Modules of Different Types

We have identified a spectrum of controllers: at one end are fast-acting controllers, but with very circumscribed ability to link up observations and actions; and at the other, slow deliberative controllers which search in problem spaces in an open-ended way. Biological control makes use of several controllers from different places in the spectrum. How the controllers are organized so as to make the best use of them is expressed as Heuristic 1.

Heuristic 1. Layer the modules such that the faster, less robust modules are at the lower levels, and slower, more robust models are on top of them, overriding or augmenting the control provided by the lower level ones. This is illustrated in Fig. 3.



The three modules above are biologically motivated. Engineering systems do not need to be restricted to these three layers precisely.

Fig. 3. Layering of modules

Many control systems in engineering already follow this heuristic. For example, process engineering systems have a primary control layer that directly activates certain important controls based on the value of some carefully chosen sensors. In nuclear power plants, the primary cooling is activated instantly as soon as certain temperatures exceed preset thresholds. In addition, there are often additional controllers that perform more complex control actions, some of them to augment the control actions of the lower level modules. When humans are in the loop, they may intervene to override the lower level control actions as well. In general, emergency conditions (say in controlling a pressure cooker or driving a car) will be handled by the lower level modules. In the case of driving a car, making hypotheses about the intentions of other drivers or predicting the consequences of a route change would require the involvement of higher-level modules performing more complex problem solving.

In addition to overriding their controls as appropriate, the higher level models can influence the lower level modules in another way. They can decompose the control task and pass on to the lower modules control goals at a lower level of abstraction that the lower-level modules can achieve more readily. For example, the deliberative controller for robot control may take the control goal of “boiling water” (say, the robot is executing an order to make coffee) and decompose it into control goals of “reaching the stove” and “turn the dials on (or off)”. These control goals can be achieved by motor control programs by using the more compiled techniques similar to those in neural and reflex controls.

Real-time control

The next set of heuristics are important for the design of real-time control systems and are based on ideas discussed in [Chandrasekaran, et al, 1991].

Control with a guarantee of real-time performance is impossible. Physical systems have, for all practical purposes, an unbounded descriptive complexity. Any set of measurements can only convey limited information about the system to be controlled. This means that the best model that any intelligence can build at any time may be incomplete for the purpose of action generation. No action generation scheme can be guaranteed to achieve the goal within a given time limit, whatever the time limit. On the other hand, there exist control schemes for which the more time there is to achieve the actions, the higher the likelihood that actions can be synthesized to achieve the control goals. All of this leads to the conclusion that in the control of physical systems, the time required to assure that a control action will achieve the goal is unbounded.

The discussion in the previous paragraph leads to two desiderata for any action generation scheme for real-time control.

Desiderata:

- 1. For as large a range of goals and situations as possible, actions need to be generated rapidly and with a high likelihood of success. That is, we would like as much of the control as possible to be *reactive*.
- 2. Some provision needs to be made for what to do when the actions fail to meet the goals in the time available, as will inevitably happen sooner or later.

Desideratum 1 leads to the kind of modules at the lower levels of the layering in Fig. 3. The following Heuristics 3 and 4 say more about how the modules should be designed.

Heuristic 2. Design sensor systems such that the system to be controlled can be modeled as rapidly as possible.

As direct a mapping as possible should be made from sensor values to internal states that are related to important goals (especially threats to important goals). Techniques in the spirit of reflex or associative controls could be useful here. In fact, any technique whose knowledge can be characterized as “compiled,” in the sense described in [Chandrasekaran, 1991], would be appropriate. However, there is a natural limit to how many of the situations can be covered in this way without an undue proliferation of sensors. So only the most common and important situations can be covered this way.

Heuristic 3. Design action primitives such that mapping from models to actions can be made as rapidly as possible.

Action primitives need to be designed such that they have as direct a relation as possible to achieving or maintaining the more important goals. A corresponding limit here is the proliferation of primitive actions.

Let us discuss Heuristics 2 and 3 in the context of some examples. In driving a car, the design of sensor and action systems has evolved over time to help infer the most dangerous states of the model or the most commonly occurring states as quickly as possible, and to help take immediate action. If the car has to be pulled over immediately because the engine is getting too hot – and if this is a vital control action – install a sensor that recognizes this state directly. On the other hand, we cannot have a sensor for every internal state of interest. For example, there is no direct sensor for a worn piston ring. That condition has to be inferred through a diagnostic reasoning chain, using symptoms and other observations. Similarly, as soon as some dangerous state is detected, the control action is to stop the car. Applying the brake is the relevant action here, and cars are designed such that this is available as an action primitive. Again, there are limits on the number of action primitives that can be provided. For example, the control action of increasing traction does not have a direct control action associated with it. A plan has to be set in motion involving a number of other control actions.

Desideratum 2 leads to the following heuristic.

Heuristic 4. Real-control control requires a framework for goal-abandonment and substitution. This requires as much pre-compilation of goals and their priority relations as possible.

As we drive a car and note that the weather is getting bad, we often decide that the original goal of getting to the destination by a certain time is unlikely to be achieved. Or, in the control of a nuclear power plant, the operator's attempts to achieve the goal of producing maximum power in the presence of some hardware failure might not be bearing fruit. In these cases, the original goal is abandoned and substituted by a less attractive but more achievable goal. The driver of the car substitutes the goal of getting to the destination an hour later. The power plant operator abandons the goal of power production, and instead pursues the goal of radiation containment.

How does the controller pick the new goal? It could spend its time reasoning about what goals to substitute at the least cost, or it could spend the time trying to achieve the new goal, whatever it might be. In many important real-time control problems replacement goals and their priorities can be pre-compiled. In the nuclear industry, for example, a prioritized goal structure called the *safety function hierarchy* is made available in advance to the operators. If the operator cannot maintain safe power production and decides to abandon the production goal, the hierarchy gives him the appropriate new goal. We acquire over time, as we interact with the world, a number of such goal priority relations. In our everyday behavior, these relations help us to navigate the physical world in close to real time almost always. We occasionally have to stop and think about which goals to substitute, but not often.

Qualitative reasoning in prediction

The last set of heuristics that I will discuss pertain to the problem of prediction. Prediction, as I discussed earlier, is a common subtask in control. Even if a controller is well-equipped with a detailed quantitative model of the environment, the input to the prediction task may be only qualitative[1]. Of course, the model itself may be partly or wholly qualitative as well. de Kleer, Forbus and Kuipers have all proposed elements of a representational vocabulary for qualitative reasoning and associated *semantics* for the terms in it (see [Forbus, 1988] for a review of the ideas). The heuristics that I discuss below can be viewed as elements of the *pragmatics* of qualitative reasoning for prediction.

Whatever framework for qualitative reasoning one adopts, there will necessarily be ambiguities in the prediction due to lack of complete information. The ambiguities can proliferate exponentially.

How do humans fare in their control of the physical world, in spite of the fact that qualitative reasoning is a veritable fountain of ambiguities? I have outlined some of the ways in which we do this in [Chandrasekaran, 1992]. The following simple example can be used to illustrate the ways in which we manage.

Suppose we want to predict the consequences of throwing a ball on a wall. By using qualitative physical equations (or just commonsense physical knowledge), we can derive a behavior

[1] I am using the word "qualitative" in the sense of a symbol that stands for a range of actual values, such as "increasing," "decreasing," or "Large." It is a form of approximate reasoning. The literature on qualitative physics uses the word in this sense. This sense of "qualitative" should be distinguished from its use to stand for "symbolic" reasoning as opposed to numerical calculation. The latter sense has no connotation of approximation.

tree with ever-increasing ambiguities. On the other hand, consider how human reasoning might proceed.

1. If nothing much depends on it, we just predict the first couple of bounces and then simply say, "it will go on until it stops."

2. If there is something valuable on the floor that the bouncing ball might hit, we don't agonize over whether the ball will hit it or not. We simply pick out this possibility as one that impacts a "Protect valuables" goal, and remove the valuable object (or decide against bouncing the ball).

3. We may bounce the ball a couple of times slowly to get a sense of the its elasticity, and use this information to prune some of the ambiguities away. The key idea here is that we use physical interaction as a way of making choices in the tree of future states.

4. We might have bounced the ball before in the same room, and might know from experience that a significant possibility is that it will roll under the bed. The next time the ball is bounced, this possibility can be predicted without going through the complex behavior tree. Further, using another such experience-based compilation, we can identify the ball getting crushed between the bed and the wall. This possibility is generated in two steps of predictive reasoning.

5. Suppose that there is a switch on the wall that controls some device, and that we understand how the device works. Using the idea in 2 above, we note that the ball might hit the switch and turn it on and off. Then because we have a functional understanding of the device that the switch controls, we will be able to make rapid predictions about what would happen to the device. In some cases, we might even be able to make precise predictions by using available quantitative models of the device. The qualitative reasoning identifies the impact on the switch of the device as a possibility, which then makes it possible for us to deploy additional analytic resources on the prediction problem in a highly focused way.

The above list is representative of what I mean by the pragmatics of qualitative reasoning, which are the ways in which we manage to control the physical world well enough, in spite of the qualitativeness inherent in our reasoning. In fact, we exploit qualitativeness to reduce the complexity of prediction (as in point 5 above). The list above leads to the following heuristics.

Heuristic 5. Qualitative reasoning is rarely carried out for more than a very small number of steps.

Heuristic 6. Ambiguities can often be resolved in favor of nodes that correspond to "interesting" possibilities. Typically, interestingness is defined by threats to or supports for various goals of the agent.

Additional reasoning or other forms of verification may be used to check the occurrence of these states. Or actions might simply be taken to avoid or exploit these states.

Heuristic 7. Direct interaction with the physical world can be used to reduce the ambiguities so that further steps in prediction can be made.

Heuristic 7 is consistent with the proposals of the *situated action* paradigm in AI and cognitive science.

Heuristic 8. The possibility that an action may lead to an important state of interest can be compiled from a previous reasoning experience or stored from a previous interaction with the world. This enables the states to be hypothesized without the agent having to navigate the behavior tree generated from the more detailed physical model.

Heuristic 9. Prediction often has to jump levels of abstraction in behavior and state representation, since goals of interest occur at many different levels of abstraction.

Compiled causal packages that relate states and behaviors at different levels of abstraction are the focus of the work on *Functional Representations*, which is a theory of how devices achieve their functionality as a result of the functions of the components and the structure of the device. Research on how to use this kind of representation for focused simulation and prediction is reviewed in [Chandrasekaran, 1994].

Concluding Remarks

The reader will note that, as promised by the use of the term "Knowledge Level" in the title, I have avoided all discussion on specific representational formalisms. I have not gotten involved in the debates on fuzzy versus probabilistic representations, linear control versus non-linear control, discrete versus continuous control and so on. All of these issues and technologies, important as they are, still pertain to the Symbol Level of control systems. The Knowledge Level discussion enabled us to get some idea both about what unifies the task of control, and about the reasons for the vast differences in actual control system design strategies. These differences are due to the different constraints on the various substasks and different types of knowledge that are available. We can see an evolution in intelligence from reflex controls which fix the connection between observations and actions through progressively decreasing rigidity of connection between observations and actions, culminating in individual or social deliberative behavior which provides the most open-ended way of relating observations and actions. I also discussed a number of biologically motivated heuristics for the design of systems for controlling the physical world and illustrated the relevance of these heuristics by looking at some examples.

Acknowledgments

This research was supported partly by a grant from The Ohio State University Office of Research and College of Engineering for interdisciplinary research on intelligent control, and partly by ARPA, contract F30602-93-C-0243, monitored by USAF Rome Laboratories. I thank the participants in the interdisciplinary research group meetings for useful discussions.

References

[Chandrasekaran, 1991] B. Chandrasekaran, "Models vs rules, deep versus compiled, content versus form: Some distinctions in knowledge systems research," *IEEE Expert*, 6, 2, April 1991, 75-79.

[Chandrasekaran, et al, 1991]
B. Chandrasekaran, R. Bhatnagar and D. D. Sharma," Real-time disturbance control," *Communications of the ACM*, August 1991, Vol. 34, # 8, 33-47.

[Chandrasekaran, 1992] “QP is more than SPQR and Dynamical systems theory: Response to Sacks and Doyle," *Computational Intelligence*, 8(2), 1992, 216-222.

[Chandrasekaran, et al, 1992]
B. Chandrasekaran, Todd Johnson, Jack W. Smith, "Task Structure Analysis for Knowledge Modeling," *Communications of the ACM*, 33-9, Sep, 1992, 124-136.

[Forbus, 1988] Forbus, K. D., 1988, Qualitative Physics: Past, Present and Future, in *Exploring Artificial Intelligence*, H. Shrobe, ed., San Mateo, CA, Morgan Kauffman, 239-96.

[Laird, et al, 1987] Laird, J.E., Newell, A. & Rosenbloom, P.S. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33, (1987), 1-64.

[Newell, 1981] Newell, A. The Knowledge Level. *AI Magazine*, Summer (1981), 1-19.