JANUS: An Architecture for Integrating Automatic and Controlled Problem Solving

David S. Day
Experimental Knowledge Systems Laboratory
Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

Abstract: This paper attempts to unify two problems in cognitive science: the relationship between "controlled" and "automatic" processing and the competing computational models of intelligence proposed by symbolic Artificial Intelligence and the connectionist school. An architecture is proposed in which symbolic and connectionist problem solving systems interact and take advantage of their different strengths. It is argued that the resulting system can account for much of the problem solving behavior associated with automatic and controlled processing as well as their complex interplay. Thus, the architecture can account for how expertise can be transformed from "explicit" to "compiled" forms via automatization, and how the opacity of the resulting automatic behavior can be counterbalanced in a cognitively plausible manner by explanations generated *ex post facto*.

1. Introduction.

This paper proposes a model for integrating *controlled* and *automatic* processing. This model assumes that these two problem solving styles are distinct not only in behavior but in implementation as well, and so an architecture has been designed in which the two competing mechanisms can also communicate and cooperate with each other. It is believed that this will result in system-wide behavior that accords well with psychological accounts of controlled and automatic problem solving and related behaviors.

The assumption of the distinct implementation of controlled and automatic problem solving grows out of the current progress in connectionist models of cognition and the resulting tension between this view and that of the symbolic Artificial Intelligence (AI) approach to these same problems. A debate has formed that pits these two views against each other as competing models. The present work joins a small but growing corpus of research devoted to establishing that a synthesis of these two models is both possible and desirable. (See, for example, Touretsky & Hinton [17], Touretsky [18], Derthick [3], Rumelhart, Smolensky, McClelland & Hinton [11], Anderson [1], Derthick & Plaut [4], among others.) Whereas other work has concentrated on establishing the theoretical possibility of incorporating one model within another—usually by simulating symbolic AI techniques in connectionist systems—we advocate the pragmatic incorporation of both models into a single system to study the complex interplay between these types of computation. Specifically, this paper claims that the proper computational model for controlled problem solving is derivable from the symbolic AI view of rule-based, categorical reasoning, while the proper computational model for automatic problem solving can be found in the connectionist view of parallel distributed processing.

This claim requires demonstrating how such distinct computational models can interact so as to present a plausible account of the rich interplay characteristic of the corresponding cognitive phenomena. The design of an architecture to support this interplay is the first step in this direction; an implementation of the system described here has not yet been completed.

The *ultimate* goal of this research is to eliminate the knowledge- or rule-based techniques used to implement this system. That is to say, we believe that the categorical, multi-step reasoning that is currently best exhibited by symbolic AI programs can eventually be incorporated into a wholly connectionist framework. However, we feel that attempting to accomplish this directly delays addressing important questions whose answers can help direct the eventual development of the totally connectionist systems. In addition, some of the issues brought out by this attempt to integrate what can be called the "propositional" and the "experiential" are interesting in their own right. For example, we believe that it is necessary for connectionist models to adopt some form of propositional representation to successfully model the important cognitive behaviors described later in this paper.

2. Competition and Cooperation Between the Controlled and the Automatic.

The concepts of controlled and automatic processing have been a topic of research in psychology for some time. (See, for example, Schneider & Shiffrin [16], Treisman [19], and Schneider, Dumais & Shiffrin, among others.) Shiffrin & Dumais [13] characterize automatic processes as highly parallel, exhibiting a marked ability to improve with practice, a limited propensity to "transfer" this expertise to dissimilar problem solving situations, making minimal demands on processing resources (other than those on which the processing is being directly carried out), and being outside of the explicit control of the problem solver. Controlled processing, on the other hand, is characterized as serial, exhibiting little improvement with practice[1], being under the direct and explicit control of the problem solver, and being much more amenable to its application in "unfamiliar" situations.

While these behaviors seem quite distinct, there is considerable interplay between the two types of processes. First, virtually all automatic skills (we will concentrate on cognitive skills in this paper) are originally played out under the direct control of the problem solver. Thus, a complex chess opening requires considerable analysis by a player when it is first encountered, but given that this opening appears a large number of times in subsequent play, it is likely that recognizing and reacting to the defense will tend to become automatic and stylized. How is this transformation, or "automatization," effected? Some problem solving is of a "mixed" character, where some steps are automated, and others require "strategic" intervention. Even when some behavior has been automated it may be possible to override its "suggestions" and solve the problem again, "from first principles." This might be done in situations that call for extreme care for one reason or another. In addition, fully automated behaviors tend to be opaque with respect to introspection and explanation, and yet sometimes explanations (or "justifications") are nonetheless provided for what clearly seem automatic cognitive skills.[2] All of this suggests that an explanation of this behavior requires a model not just of the independent mechanisms but of their interaction as well.

While the psychological literature on these issues has grown, Schneider [14] points out that the treatment of many of the particular phenomena associated with this distinction has tended to remain at the level of only vague verbal descriptions of the underlying mechanisms. Schneider's paper proposes a four phase model of the development of automatic processing. While Schneider's work concentrates on a particular problem and provides a very detailed account of this example, the present paper advocates a general computational architecture. Schneider describes a particular algorithm for variably-mapped category search with two categories. The imple-

[1]This excludes the process of automatization, of course, in which the skill becomes automatic over time.

[2]This has been illustrated in many places. For example, in Expert System construction it has appeared when interviewing experts about why they performed certain actions in some task. Often their explanations seem either to disagree with the rapidity of the subject's choice of action, or else the explanation is insufficient to account for actions chosen in other similar situations.

mentation involves the comparison, modification and combination of activation values of threshold units. As a controlled process, these steps are carried out by an algorithm that is "hard-coded" into the system, which manipulates the values and gates in the system of units.

The JANUS architecture described in this paper might be seen as a generalisation of this process, since this and any other algorithm can be encoded within a system of productions.[3] This allows the researcher to further study effects of attention and problem solving strategy within the production system paradigm—at least so far as *controlled* cognitive skills are concerned. The relationship between these productions and the values of units that are involved in the subsequent automatization is much more complex than that in Schneider's model due to the translation of these rules into a distributed representation. Despite these differences, we believe a set of phases similar to Schneider's four phase model will fall out of the system described here as well (see Section 3).

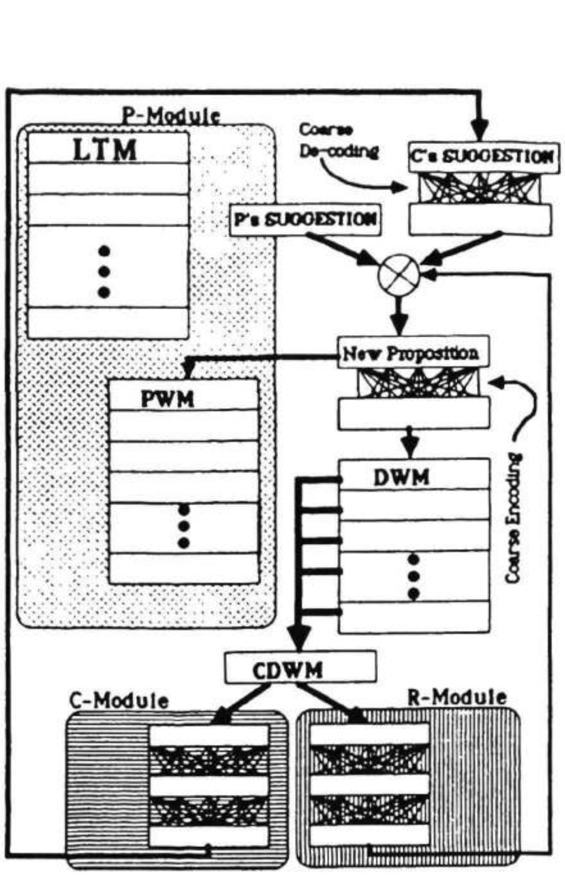

Figure 2-1.

2. A Description of the JANUS Architecture.

The proposed system contains three modules (see Figure 2-1). One contains a simple rule-based problem solving system. This is called the P-module (for "propositional"). As with any rule-based system, the P-module consists of a long term memory in which are stored the rules, and a working memory (PWM in Figure 2-1) in which are stored what is currently asserted (or "believed") by the P-module.[4] The PWM of the P-module contains the description of the current problem state. As the steps in solving a problem are followed, the intermediate results, hypotheses and final solution proposed are also placed in this working memory.

The other two modules are both three layer feedforward networks (where two layers have modifiable weights) utilizing the backpropagation learning rule (Rumelhart, Hinton & Williams [10]). These two modules are the C-module (for "connectionist") and the R-module (for "resolution"—though it, too, is a connectionist module). The input layer of both connectionist modules is obtained in the following way. As each new proposition is asserted by the P-module and thereby placed in its local propositional working memory (PWM), it gets coarse-coded (in a manner to be described shortly), and this distributed representation of the proposition is placed at the front of a relatively small queue of such encoded propositions—it can hold only ten such

[3]Schneider makes note of the logical connection between the controlled-processing operations in his model and rules in a production system.

[4]One atypical restriction on this rule-based system is that the size of the working-memory is limited to a fairly small number of propositions—approximately twenty or so. This is to keep the P-module's available working memory close to the same size as that of the C- and R-modules, discussed below.

propositions. This queue is called the DWM (Distributed Working Memory) and as each new distributed proposition is added to the top-most cell, all the other proposition are pushed down, removing the proposition in the bottom-most cell altogether. All ten of these distributed representations of propositions are then coalesced into one distributed representation of all of them by simply adding the vectors together. It is this coalesced vector (the CDWM in the figure) which is the input layer for both the C- and R-modules.

It might at first appear that information is being hopelessly confused when the separate propositions in the DWM are "coalesced" into the single CDWM. However, the loss of information can be significantly reduced by using an adequate coarse-coding scheme. This is because each feature in the coarse-coded representation has the ability to distinguish the role being played by some domain object (or concept) in the proposition from which it is derived. This allows the C-module and the R-module to include a more complete "description" of the current problem state by viewing up to ten propositions at one time. (McClelland's CID mechanism [8] utilizes a similar technique.)

The C- and R-modules differ, however, in their output layers and the effects of different values on these output layers. The output of the C-module is a distributed representation of a proposition. The output of the R-module is a single binary unit. The C-module's output layer is placed in a vector ("C's Suggestion") which is then translated into a local (propositional) representation of the module's output. That is, there is a single layer network which takes a distributed representation of a proposition and generates on its output layer a "local" representation of that proposition. This process is simply computing the inverse of the coarse-coding process mentioned earlier. If the value of the R-module's output unit is *greater* than zero, it has the effect of placing this local representation of the C-module's output layer in the "New Proposition" buffer (see Figure 2-1). If the R-module has instead produced an output *less* than zero, then the C-module's output layer would be ignored, and the contents of the local working memory buffer within the P-module ("P's Suggestion") would be the source of the next addition to the propositional working memory ("New Proposition") within the P-module.

Thus, the basic structure of the system is that the P- and C-modules are both generating propositions based on the contents of the P-module's working memory (though the C-module's version is somewhat removed from the original because of the coarse-coding and coalescing processes), and both are "competing" to have their proposed propositions "asserted" by the system as a whole by having them placed in the "New Proposition" buffer, from whence it is forwarded to the working memories of all three modules. The arbiter of this competition is the R-module. The actions that the C- and P-modules would carry out on the P-module's propositional working memory (PWM) are 'gated,' and this gate is controlled by the R-module.

Training the Connectionist Modules.

The R-Module. The R-module is a back-propagation network, and the basis of its training schedule is to prefer the suggestions from that module which lead to a solution. However, the R-module also has a built-in bias towards letting the C-module set the PWM (since we want to encourage this more efficient form of problem solving). The R-module is thus computing a confidence rating of the C-module's abilities. This means that in the early stages of performance, since the C-module has had too little time to learn anything, the R-module will quickly learn to favor the P-module's suggestions for how to change the state of PWM in order to solve a given problem. Over time, as the C-module begins to learn to mimic the P-module's actions, the R-module's bias towards automatic processing will lead to the C-module's taking over some or all of the problem solving. The R-module's input layer is provided with the whole CDWM in order to allow it to distinguish those problem states for which the suggestions from either the C- or P-module should be selected.[5]

The C-Module. The C-module is only trained (it's error backpropagated and weights modified) if the *P-module's* "suggestion" is placed in the PWM (via the "New Proposition" buffer). In this event, the C-module is trained to match the P-module's suggestion. If the C-module's *own* suggestion has been chosen by the R-module to be placed in the PWM, then no training of the C-module is done. If the resulting solution leads to an error, then only the R-module is penalized for relying on the C-module's suggestion. (This is discussed further in the next section.)

Distributed Representation of Propositions. The propositional working memory (PWM) of the P-module requires that propositions be used to represent the problems and the intermediate states leading to solution in order to allow rules to be invoked using pattern matching. But symbolic propositions cannot be presented directly to the connectionist modules C- and R-; they require feature vectors. Space precludes a detailed treatment of this important representational issue, but a transformation of the P-module's local propositional WM can be performed by a coarse-coding technique adapted from the work of Hinton [6], McClelland & Kawamoto [7], and Saund [12]. The basic idea here is that symbols in the propositional patterns of WM should be identifiable independent of the different roles they might play in different propositions. At the same time, the role they play is also important to express the meaning of the proposition. The method adopted by Hinton [6] is particularly useful in this regard.

The distributed representation is generated by a training schedule which must be performed before the various modules are put together. This network would be trained to map the local feature vector representation of the proposition to another local feature vector representation of this same proposition through the intermediate layer of units, which would consist of a smaller number of units than either of the other two layers (see Saund [12] for a similar setup). It is this process which derives the coarse coding representation used in the C- and R-modules of the JANUS architecture. Once this set of weights has been developed, the weights between the first and intermediate layer of this network (coarse *en*coding) are used to translate between the most recent propositional addition to the P-module's PWM (via the "New Proposition") and the distributed representation of that proposition in the top-most queue entry of the DWM (see Figure 2-1). The weights between the intermediate and output layers (coarse *de*-coding) are used to map the C-module's propositional "suggestion" before being placed in the "New Proposition" buffer.

3. Using the Architecture to Model Cognitive Behavior.

The architecture just outlined was developed as an attempt to model the following types of behavior.

Categorical Reasoning. The basis of connectionist problem solving is, so far, via problem *recognition*; a solution to even a novel situation is based on having generated a mapping from problem states to solutions which, through generalization, will capture situations sufficiently similar to those in the training set. However, there are classes of problems—addition is a good example, as is trying to determine the voltage on some wire in a fairly complex circuit diagram—where it seems that a multiple step solution is called for based on what has been called a 'model' of the domain. Similarity to previous problems tends to be a weak indicator. So far, such models have been able to be most easily expressed and used in 'categorical,' symbolic AI systems. As the P-module is a relatively standard rule-based

[5]It is probable that the R-module could actually be incorporated into the C-module by adding an extra output unit to the C-module with the above interpretation. However, this aspect is separated out for ease in testing alternative formulations of the R-module's behavior.

system, such knowledge will be able to be encoded and used in this powerful way by the P-module. As mentioned in the previous section, the P-module will very soon tend to be favored over the C-module as the module to be "trusted" in novel situations—that is, in novel states of the PWM.)

Automaticity. The C-module is not idle, however, when the P-module is being deferred to. While the P-module is holding sway,[6] the C-module is being trained to produce the solutions generated by the P-module, which is the source of training examples for the C-module. As a growing number of problems are presented to the system as a whole, it is presumed that the C-module will be able to generate a mapping such that the solutions can be suggested on the basis of the similarity of the problem state to previously seen problems.[7] This is simply the standard way in which connectionist networks learn to perform mappings.

Automatization. This refers to the transfer of knowledge from an explicit, categorical (or propositional) form (in the P-module) to the distributed and more efficient recognition-based form in the C-module. It is the main goal of this research to demonstrate that the knowledge that is initially encoded explicitly and that can be used only by chaining through some number of reasoning steps will be subject to incorporation into the C-module after sufficient instances of such problem solving are presented. This process would proceed in stages not dissimilar to those described in Schneider [14] and the "compilation" process modelled in Anderson, et al. [2]. As inferences become automated by the C-module, this allows the corresponding propositions in the P-module to be skipped. This, in turn, allows more propositions of the PWM to "fit into" the distributed working memory (DWM), which would allow further automatization to occur. In this way, more and more multi-step inferences are slowly encompassed by single step inferences carried out by the C-module. The advantage of this scheme over the rule-based scheme adopted by Anderson, et al. [2] is that the C-module can generalize and thereby generate plausible suggestions for changing the PWM for problems never seen before by the system as a whole, as long as there is sufficient similiarity to previously seen problems.

Explanation. A major difficulty of connectionist systems has been their opacity. When a network presents some 'solution' on its output units, it is not at all clear the 'reasons' that support this computation. Of course, the only truly *valid* explanation would involve a complete listing of all the net's previous training examples which led it to have the weights it now has; and then these weights, and their ramifications, might also be explicated in some fashion. Except for trivial problems, this approach seems ludicrous.

On the other hand, following a plausible model of how we generate some explanations ourselves, we can imagine that the explanations of fully (or partially) automatic behaviors are generated *post facto* by utilizing any relevant categorical knowledge within the P-module. For example, in solving for the voltage on a wire in some complex circuit diagram a system as described here might generate an opinion based on the similarity of the network layout to a large number of previously seen layouts. Asked for an explanation ('justification' might be a more accurate term), the system might use the proposed solution and the known initial problem state to generate a solution using the *P-module's* knowledge alone. This *post facto* (and more expensively derived) solution has the advantage of perspicuousness of the knowledge used to generate the solution.

[6]This will tend to happen for relatively long periods of time, since the PWM changes only slowly at each cycle, and it is on the basis of recognizing states of the PWM that the R-module assigns priority to either module.

[7]This relies, of course, on the R-module *also* noting this similarity and being prepared to 'trust' the solution 'proposed' by the C-module. This suggests that the "natural degree" to which the R-module has preferential bias for the C-module might be out of line—either too much or too little—with the rate at which the C-module actually learns; this should make itself readily apparent when testing the system.

Of course, this is hardly the same way in which the original solution was in fact generated. On the other hand, to the extent that the C-module's knowledge is the result of automatization of previous explicitly encoded knowledge in the P-module, such an 'explanation' might not be so inappropriate or unrelated to the automatically derived solution.

It should be noted that with this method it is perfectly possible that the "explanation" would not agree with the original solution. This would hardly seem to be undesirable, however, since if we are to make assumptions on the basis of generalizing from past experience, it is very useful to know when those assumptions disagree with the knowledge acquired directly in the form of rules or other propositions.

Other Types of 'Mixed Reasoning'. The previous description of explanation introduces other possibilities for intermixing the two types of knowledge to obtain satisfactory performance. Such techniques would take the form of the R-module's alternating between the two 'knowledge sources' to modify working memory in the intermediate steps of solving some larger problem. For example, in the circuit analysis example, an initial set of solutions might be 'proposed' by the C-module on the basis of the circuit's similarity to other circuits, which might be followed by an explanation-like process in which the rough possible solutions are checked and/or refined by more expensive (but possibly better supported by reference to 'first principles') categorical reasoning of the P-module.

4. Summary.

This paper proposes a method of integrating *automatic* and *controlled* forms of problem solving by building an architecture in which connectionist and standard symbolic AI implementation techniques complement each other in a single system. This is seen as an important step towards eventually incorporating the propositional (or rule-based) form of knowledge found in AI systems into completely connectionist networks. This architecture is described structurally. The paper describes various cognitive behaviors which derive from the interplay between the two problem solving styles and that this architecture would be able to model. The emphasis of this research is on the way in which expertise can be transformed from "explicit" to "compiled" via automatization, and how the opacity of the resulting automatic behavior can be counterbalanced in a cognitively plausible manner.

5. Bibliography.

1. Anderson, J. A., (1983) "Cognitive and Psychological Computation with Neural Models," *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13, 799-815.

2. Anderson, John R., J. G. Greeno, P. J. Kline & D. M. Neves, (1981) "Acquisition of Problem Solving Skill," in J. R. Anderson (Ed.), *Cognitive Skills and Their Acquisition*, Hillsdale, N.J.: Erlbaum.

3. Derthick, Mark (1986) "A Connectionist Knowledge Representation System", CMU Thesis Proposal.

4. Derthick, Mark & David Plaut, (1986) "Is Distributed Connectionism Compatible with the Physical Symbol System Hypothesis?," Proceedings of the Cognitive Science Society.

5. Gallant, Steven, (1985) "The Automatic Generation of Expert Systems From Examples," Proceedings of Second International Conference on AI Applications, Piscataway, NJ, 313-319.

6. Hinton, Geoffrey, (1986) "Learning Distributed Representations of Concepts", Proceedings of the Cognitive Science Society.

7. McClelland & Kawamoto, (1986) "Mechanisms of Sentence Processing: Assigning Roles to Constituents," *Parallel Distributed Processing, Vol. II*, McClelland & Rumelhart, eds., Cambridge, MA: MIT Press.

8. McClelland, J., (1986) "The Programmable Blackboard Model of Reading", *Parallel Distributed Processing, Volume II*, McClelland & Rumelhart, (Eds.), Cambridge: MIT Press.

9. McClelland & Kawamoto, (1986) "Mechanisms of Sentence Processing: Assigning Roles to Constituents," *Parallel Distributed Processing, Volume II*, McClelland & Rumelhart, (Eds.), Cambridge: MIT Press.

10. Rumelhart, D., G. Hinton & R. Williams, (1986) "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing, Volume I*, Rumelhart & McClelland, (Eds.), Cambridge: MIT Press.

11. Rumelhart, D., P. Smolensky, J. McClelland & G. Hinton, (1986) "Schemata and Sequential Though Processes in PDP Models," *Parallel Distributed Processing, Volume I*, Rumelhart & McClelland, (Eds.), Cambridge: MIT Press.

12. Saund, Eric, (1986) "Abstraction and Representation of Continuous Variables in Connectionist Networks," Proceedings of the AAAI.

13. Shiffrin, Richard M. & Susan T. Dumais, (1981) "The Development of Automatism," in *Cognitive Skills and Their Acquisition*, edited by John Anderson, Hillsdale, NJ: Erlbaum.

14. Schneider, Walter, (1985) "Toward a Model of Attention and the Development of Automatic Processing," in Posner & Marin (Eds.), "Attention and Performance XI," Hillsdale, N.J.: Erlbaum.

15. Schneider, W., S. Dumais & R Shiffrin, (1984) "Automatic and Control Processing and Attention," in *Varieties of Attention,* Academic Press.

16. Schneider & Shiffrin, (1984) "Controlled and Automatic Human Information Processing: I. Detection, Search and Attention," *Psychological Review*, 84.

17. Touretzky and Hinton, (1985) "Symbols Among the Neurons", Proceedings of the International Joint Conference on AI.

18. Touretzky, (1986) "BoltzCONS: Reconciling Connectionism with the Recursive nature of Stacks and Trees," *Proceedings of the Cognitive Science Society.*

19. Treismann, Anne, (1969) "Strategies and Models of Selective Attention." *Psychological Review* 76, 282-299.