Abstract Computer Models: Towards a New Method for Theorizing About Adaptive Agents

Stellan Ohlsson and James J. Jewett
Learning Research and Development Center
University of Pittsburgh

Abstract

After a brief flourish in the decade 1979-1989, the study of learning has once again stalled. The main method for theorizing about learning--symbolic computer simulation--is plagued by serious difficulties. *Abstract computer models*, i. e., models that capture the structural features of cognitive processes while ignoring their content, overcome those difficulties. An example of abstract modeling is discussed and a research agenda outlined.

1. A Promise Unfulfilled

During the decade 1979-1989, the newly invented techniques for implementing machine learning systems inspired a wave of theorizing about learning in general and skill acquisition in particular. An interesting variety of learning mechanisms were proposed, including rule generalization, rule discrimination, rule composition, chunking, subgoaling, procedure induction, proceduralization, strengthening and weakening, constraint-based error correction, redundancy elimination, genetic algorithms, analogical transfer and explanation-based learning (Anderson, 1981; Chipman & Meyrowitz, 1993; Holland, Holyoak, Nisbett & Thagard, 1986; Klahr, Langley & Neches, 1987; Kodratoff & Michalski, 1990; Shrager & Langley, 1990).

These hypotheses represent the first attempts at describing, at a fine level of detail, how adaptive agents alter themselves on the basis of experience. Although incomplete, the list nevertheless shows that *more hypotheses about the cognitive mechanisms underlying learning were invented in that decade than in the preceding century*, i. e., since Edward L. Thorndike and his contemporaries posed the explanation of learning as a central scientific problem in the 1890's.

In the mid-1980's, the stage was thus set for a productive research program aiming to invent more learning mechanisms, identify the properties of various mechanisms, derive their implications and use them to explain the behavior of particular adaptive agents. Instead, we now see scattered, isolated efforts at theory, surrounded by a cloud of unprincipled empirical studies that show no signs of crystallizing into a coherent body of knowledge. The production of new ideas about the mechanisms underlying learning has almost stopped and there is little work on analyzing and evaluating the hypotheses already proposed.[1] This impasse is caused, in part, by certain problems with symbolic machine learning systems as vehicles for theorizing.

2. Problems With Symbolic Models

We now have three decades of experience in building symbolic computer simulation models of cognitive processes. Several complexities and problems have appeared (Anderson, 1987; Frijda, 1967; Kieras, 1985; McCloskey, 1991; Neches, 1982; Ohlsson, 1988; Schneider, 1988; Young, 1985). These difficulties include the following:

Knowledge representation. A symbolic simulation model requires an explicit representation of the knowledge that the modeled agent is drawing upon. Such a knowledge base represents a large number of micro-hypotheses about what the modeled agent knows and how that knowledge is encoded. Those assumptions are, in principle, unverifiable. There are no empirical methods that deliver such fine grained information that we can ground the individual nodes and links in a knowledge base independently of each other. Hence, the knowledge base of a symbolic simulation model functions as a giant free parameter. Every process assumption can be made to generate a wide range of different behaviors by 'hacking the representation'. This introduces considerable fuzziness in interfacing such models with data.

Domain specificity. Because a symbolic model requires a knowledge base, its behavioral predictions are only valid for the corresponding task domain. To investigate the behavior of the model in a different domain, one has to implement a new knowledge base. Because knowledge is a major determinant of

[1]The work of John R. Anderson on the ACT theory (Anderson, 1993) and the work of the Soar group on the Soar theory (Newell, 1990) are the main exceptions to this generalization.

the model's behavior, its behavioral predictions may or may not hold up in the new domain. Hence, when a simulation model is claimed to explain a *general* empirical phenomenon (as opposed to a particular experimental result), it is difficult to evaluate the claim: How do we know what the model would do in another task domain? There is no principled way to identify what is general and what is task specific about the model's behavior.

Indeterminate referent. A symbolic model is a single entity, a particular cognitive agent. It has a particular knowledge base, a particular set of processes, particular parameter settings (e. g., activation levels), and so on. What is such a system a model *of*? Obviously, adaptive agents (e. g., people) differ at least slightly in their knowledge of a domain, as well as in their strategies for dealing with it; seldom if ever are two individuals exactly alike. But if all individuals differ from each other, then a simulation model can be a model of at most one of them.

One solution to this problem is to regard a simulation model as a model of the *average* individual. However, in the face of qualitative differences in knowledge or strategy, what is average? How does one take the average of two knowledge bases or two problem solving strategies? There is only one principled way to bridge the gap between the concretion of a symbolic computer program and the generality we desire in a learning theory: To include individual difference variables in the model, to systematically vary those variables and to verify that they cause the model to produce the range of behaviors observed empirically. However, it is difficult to vary a symbolic simulation model systematically.

Brittleness. Symbolic simulation models are no less brittle than other AI systems, i. e., they do not function well at the boundaries of their knowledge, they usually cannot be applied to new problems without extensive re-programming, and the effects of changes to either the knowledge base or the process assumptions are often unpredictable and sometimes disables the program entirely. If a change requires extensive reprogramming, it is difficult to describe in a principled way what changed and what remained the same. Hence, one cannot easily conduct experiments in which particular aspects of such a model are systematically varied.

Principles versus implementation. Symbolic simulation models always include a mixture of theoretically motivated code and what I call *convenience code*, i. e., code that had to be added to make the model run (Ohlsson, 1988). The behavior of the model, and hence its behavioral predictions, depend as much on the convenience code as on the theoretically motivated code. Hence, it is difficult to know which predictions to take seriously and which to dismiss as accidental consequences of the particular implementation. The only principled solution to this problem is to have a theoretical reason behind *every* aspect of the code (Newell, 1990), a methodological dictum that is hard to live by and which increases the amount of labor required to build a model.

Labor intensity. The first AI-based simulations of learning were small and simple (e. g., Anzai & Simon, 1979). However, as the field shifted attention to real life tasks, the complexity of our simulations increased to a point where the implementation of a serious learning model requires several man-years, perhaps even tens of man-years, of work. At the end of that investment, the only product is a formalization of a single point in the theory space, and the only benefit of having the model is that we can derive the behavioral predictions at that point. This lack of proportion between initial investment and ultimate benefit is one reason why only one or two research groups continue to build symbolic models of learning; those groups are already up and running, as it were, so they can make progress through incremental improvements on past achievements. However, there are few, if any, new entrants into the field; the cost of entry is prohibitive.

In short, symbolic simulation models are not convenient tools for theorizing about learning. The interaction between domain specificity and labor intensity is lethal. At the end of several years of work, the theoretician might have a formalization of single cell in the table of all mechanisms versus all tasks. We need look no further for an explanation of why the development of learning theory has stalled.

3. Abstract Computer Models

Continued progress in learning theory requires a theoretical method that combines the abstraction, elegance and simplicity of mathematical models with the flexibility and power of computational models. The solution that I am pursuing is called *abstract computer modeling*. Abstract models are computer programs and hence not limited by mathematical tractability, but they are not AI systems and hence do not require domain specific knowledge or complicated algorithms.

The key step that leads to abstract modeling is to turn a common criticism of AI on its head: It is often said that computer programs are purely syntactic systems, i. e, they do not understand the symbols they operate upon and the intended meaning of those symbols has no impact on the system's behavior. Only the *structure* of the knowledge base and the associated processes are important.

The implication of this observation is that the content (i. e., the knowledge base) of an AI-based model is not needed to generate its behavioral predictions. Hence, the basic idea of abstract modeling is to take the content (knowledge) out of the model. The result is a computer program which mimics the structural properties of the corresponding AI system and *which therefore goes through the same number of steps,* i. e., which makes the same quantitative predictions.

This is not an entirely new idea. For example, within AI it has long been understood that one can compute the number of steps that a particular search algorithm requires to solve a problem, if one knows the length of the desired solution path and the branching factor of the search space, *without actually having to implement the system,* (e. g., Nilsson, 1971). We have taken this insight one step further by applying it to machine learning systems.

An abstract computer model is not an AI-program. It does not carry out the processes it models. An abstract model of problem solving does not solve problems and an abstract model of learning does not learn. Unlike AI models, abstract models do not satisfy the *sufficiency criterion* proposed by Newell, Shaw and Simon (1958, p. 151) as a touchstone for theories of cognition. An abstract model goes through certain motions and counts how many steps (of certain kinds) it takes to do so. For the study of adaptive systems, this is enough. As theoreticians, our task is to derive the behavioral consequences of our hypotheses with as little effort as possible, not to make computers intelligent.

An abstract computer model can be viewed as an abstraction of a particular symbolic model. For example, Rosenbloom (1986; see also Rosenbloom & Newell, 1987) used an abstract model to analyze certain aspects of the XAPS simulation model.[2] Similarly, our first abstract model was an abstraction of the HS simulation model (Ohlsson, 1993). From this point of view, an abstract model is a tool for analyzing the properties of an existing system.

[2]Rosenbloom (1986) called his abstract model a *meta-model*, to emphasize the relation between the abstract model and the AI system, XAPS, that it was abstracted from. Because we want to build abstract models for which there are no prior symbolic models, we find the term "abstract model" more descriptive.

However, this view is too limited. There is no need to limit abstract modeling to points in the theory space for which symbolic models have already been built. On the contrary, for purposes of studying adaptive agents we can *replace* symbolic models with abstract models.

The advantages of replacing symbolic models with abstract models are numerous. First, there is no need to implement and debug either a knowledge base or any AI algorithm. Implementing and debugging an abstract model can therefore be done in a few weeks, sometimes a few days. Second, an abstract model is not tied to a particular task, so its behavioral predictions are general. Third, an abstract model can easily be changed and manipulated.

In summary, one possible escape from the current impasse is to work with *abstract computer models*. Such models are not AI systems and hence do not suffer from the latters' problems and difficulties, but they are computer programs and hence escape from the constraints of mathematical tractability. Abstract models correspond to particular computational mechanisms in the sense of mimicking the structural features of those mechanisms. By going through the same motions as a symbolic model, an abstract model allows us to derive the behavioral predictions of the latter without implementing it. The ease with which abstract models can be built and manipulated encourages systematic search through the theory space and extensive exploration of parameter variations.

4. An Example of Abstract Modeling

Several different learning mechanisms have been modeled abstractly and a number of results derived (Ohlsson & Jewett, 1994). The purpose of this section is to present an illustrative example.

Like a symbolic model, an abstract model can be described as consisting of three components: a task environment, a performance module and one or more learning mechanisms. The main creative step in abstract modeling is to figure out how to represent change in knowledge without an explicit representation of the knowledge base.

4.1 An Abstract Task Environment

The central features of the environments in which adaptive behaviors occur are that (a) they require a *sequence* of actions, as opposed to a single action, and (b) there are *multiple options* at each point in that sequence, only some of which are correct in the sense of being on a path to the desired end state. These features imply that an environment can be conceptualized as a tree structure. Each situation corresponds to a node in the tree and an action that changes situation S_1 into situation S_2 corresponds to a link between nodes S_1 and S_2. This *situation tree* is a map of the task environment.[3]

Because we are abstracting from task content, the nodes and links in the tree structures that we use to model the environment are empty. Each node represents some situation but it contains no information as to *which* situation, i. e., which propositions are true with respect to the represented situation. Similarly, the actions are represented by links between the situations, but the links contain no information about which action is being represented; each link stands for some unspecified action. In short, the nodes are uninterpreted situations and the links are uninterpreted actions.

A situation tree has three additional features. We designate a randomly chosen leaf node as the goal. The fact that a particular node is the goal is modeled by putting the *label* "GOAL" on that node (rather than by writing down a propositional *description* of the goal situation, as in a symbolic model). Links which are on a path between the root node and the goal node are labeled "correct", the others are labeled "incorrect". In addition, each link is associated with a *strength*. Initially, all links have the same (arbitrarily chosen) strength value. (Nothing in the methodology forces equal initial strengths, but it is a natural starting state.)

4.2 Abstract Performance Module

To perform a task is to traverse the corresponding situation tree from the root to the goal. Each situation (node) presents the learner with the problem of deciding which option (link) to traverse. In our abstract models, performance is modeled as in Figure 1.

The program sketched in Figure 1 is simple. Nevertheless, it executes the same number of steps as an AI algorithm operating on a task-specific knowledge base. The abstract mechanism does not apply a strategy, execute operators or

[3]The situation tree obviously owes much to the concept of a *search space* or *problem space* (Newell & Simon, 1972). However, the problem space is an hypothesis about the mind. In contrast, the situation tree is a tool for describing the environment.

apply goal criteria, but it mimics the quantitative properties of a system that does perform those computations.

```
1. Check whether the current node is the goal node;
   a. if yes, exit with the relevant counter(s);
   b. if no, retrieve all outward-bound links (options).
2. Select one link probabilisticly, with the probability of link
   L a function of the strength associated with L.[4]
3. Make the node at the far end of that link the current node
   and update the step counter.
4. Check whether the step was correct or incorrect;
   a. if correct, go to 1;
   b. if incorrect, backup to the previous node and go to 2.
```

Figure 1. Pseudo-code for performance module.

We could derive predictions about steady state performance by running the performance model and study, e. g., how the number of steps required to complete a task (traverse a tree) varies with various parameters (e. g., tree size). However, the function of the performance model for present purposes is to serve as a platform for the learning mechanisms that we want to study.

4.3 A Model of Failure-Driven Learning

One plausible hypothesis about learning is that adaptive agents react to errors, expectation failures, negative feedback, or, in general, *negative outcomes* by revising the performance module in such a way as to avoid similar outcomes in the future. This hypothesis was explored in the (symbolic) HS model (Ohlsson, 1992, 1993, 1994; Ohlsson, Ernst & Rees, 1992; Ohlsson & Rees, 1991a, 1991b).

[4]The exact algorithm used is the following: Each strength value is multiplied with a random number between 0 and 1; the highest product wins. This algorithm gives a strong preference to the link with the highest strength. We are in the process of exploring alternative decision algorithms.

The HS model was based on an analysis of how a knowledge-based agent can detect and correct his or her own mistakes. In brief, the basic principles proposed were that (a) errors are caused by overly general rules which generate actions that are not appropriate or useful; (b) errors are detected by judging successive situations as either consistent or inconsistent with known constraints on 'good' situations; and (c) errors, once detected, are corrected by specializing the responsible rule in such a way that it will not be evoked in situations in which it causes errors; the particular specialization is determined by the learner's casual analysis of the error.

For example, consider a novice driver. (a) Driving on a two-lane highway, a novice driver is likely to have a general tendency to change into the left-hand lane whenever the vehicle in front is traveling too slowly. In its unrestricted form, this disposition is dangerous, because it will lead him or her to change lanes even when there is another car in the left-hand lane. (b) One possible outcome of such an error is screeching tires or irritated honkings from that other car. Even a novice driver knows that such signals indicate an error on his or her part. (c) The causal analysis is in this case obvious: The error arose because there was another car in the left-hand lane. Hence, the needed correction is to specialize the overly general disposition so that it is only evoked in situations in which the left-hand lane is empty.

Our efforts to derive the behavioral implications of the HS model illustrates the difficulties of symbolic modeling. The model was applied to three different, fairly small, task domains: counting, arithmetic and a particular skill in organic chemistry (see Ohlsson, 1993, for an overview of the three applications). Development of the knowledge base took over a year for each domain. Hence, providing minimal evidence for the generality of the learning mechanism took three years of work.

Furthermore, to derive the learning curve predicted by the HS model we ran a quantitative learning experiment in which the model learned a simple symbolic skill in chemistry: to construct Lewis structures (structural formulas) for particular molecules given their sum formulas. HS could learn this skill both by practicing repeatedly on a single molecule and by practicing a sequenced of different molecules. Figure 2 shows a typical result from repeated practice on a single molecule.

The simulation runs required to identify the learning curve of HS took several months to complete. To investigate how the curve is affected by various parameters, e. g., task complexity, we would have to run the simulation repeatedly for

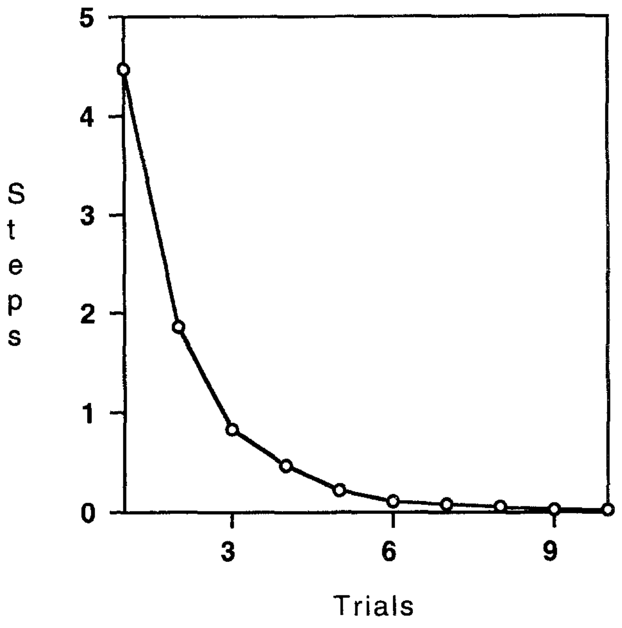

Figure 2. Learning curve for HS model.

each parameter value, making this seemingly simple analysis a multi-year enterprise.

Abstract representation. Abstract modeling provides a more convenient way to derive the implications of the basic hypotheses behind the HS model. The main creative step in constructing an abstract model is to represent the correction of faulty knowledge without an explicit representation of that knowledge. To a first approximation, the effect of detecting and correcting an error is that one avoids that error in the future. (Error correction might have other effects as well, but avoidance of the error is obviously central.) How do we represent this outcome in the abstract?

The effect of error correction is that the unsuccessful option will not be tried again. This effect can be modeled by *deleting the relevant link* from the situation tree. This operation has the same effect as constraining a knowledge structure so that it does not apply in a particular situation; the action generated by that knowledge structure will not be taken again. Similarly, to remove a link from a tree is to make the corresponding option unavailable for the decision making algorithm; but if the link is not considered during decision making, then it will not be selected and hence not traversed. The (abstract and simple) operation of deleting a link mimics the (content-full and complicated) process of correcting an error.

The abstract model that implements this learning mechanism works as shown in Figure 3 (the pseudo-code added to the performance module--see Figure 1--to model error correction is shown in bold faced font). This abstract model mimics the behavior of the symbolic HS model. To determine the learning curve produced by the version of failure-driven learning implemented in HS, we ran a series of simulation experiments with the abstract model.

```
1. Check whether the current node is the goal node;
   a. if yes, exit with the relevant counter(s);
   b. if no, retrieve all outward-bound links (options).
2. Select one link probabilisticly, with the probability of
   link L being proportional to the strength associated
   with L.
3. Make the node at the far end of that link the current
   node and update the step counter.
4. Check whether the step was correct or incorrect;
   a. if correct, go to 1;
   b. if incorrect, backup to the previous node, remove
      the traversed link from the situation tree,
      and go to 2.
```

Figure 3. Pseudo-code for error correction model.

Simulation experiments. Our goal was to make the experiments as similar as possible to empirical studies of skill acquisition. We used situation trees with a branching factor of 10 and a path length of 20. These trees represent the abstract structure of a task that requires 20 steps (actions) to solve and that present the performer with approximately 10 options in each step. Tasks that illustrate this level of complexity include proving a college-level algebra theorem, using a word processor to write a letter and cooking a dish from the classical French cuisine.

Each simulation experiment consists of running the model through a sequence of training trials, i. e., traversals of the situation tree. The model is started in the root node of the tree and cycles through the decision cycle until it reaches the goal node; it is then restarted in the root node; and so on. The training trials continue until mastery. The criterion for mastery is three consecutive error-free traversals of the situation tree.

We run several simulated subjects until criterion, as is standard procedure in experiments on learning. (Because the decision making module is probabilistic, successive runs are not identical.) In the explorations reported below, there were 20 simulated subjects in each simulation experiment.

Finally, we collect data on the behavior of the model, typically the number of steps (link traversals) required to traverse the tree. These numbers are averaged across simulated subjects for each trial. Hence, the procedure by which we construct learning curves corresponds closely to the empirical procedures used to construct the learning curves of human learners.

Selected results. We have performed extensive explorations of the abstract error correcting model (Ohlsson & Jewett, 1994). Figure 4 shows the basic result, plotted with logarithmic coordinates along the y-axis. The behavior of the model is a straight line in this type of plot. This is the hallmark of an *exponential* learning curve. The fit to the exponential is extremely good. Furthermore, the rate of learning is very high. The model requires less than 10 trials to reach criterion. (To judge the psychological plausibility of this achievement, remember that the task requires 20 steps and presents the learner with 10 options in each step.)

For comparison, Figure 5 shows the learning curve for the symbolic version of HS (see Figure 2) replotted in log-normal coordinates. Both the symbolic and the abstract models produce good approximations to straight lines in log-normal coordinates, indicating that both generate exponential (rather than power law) learning curves. The slopes of the two curves (as indicated by the exponents) are also comparable (.289 versus .314). Hence, the abstract model does indeed mimic the quantitative behavior of the symbolic model.

The amount of work required to derive the predicted learning curve with the abstract model was a mere fraction of what it took to make the corresponding derivation with the symbolic version of HS: a few days versus several months. The ease and speed with which the abstract model can be manipulated also enables us to carry out several other analyses that would be completely prohibitive with the symbolic model.

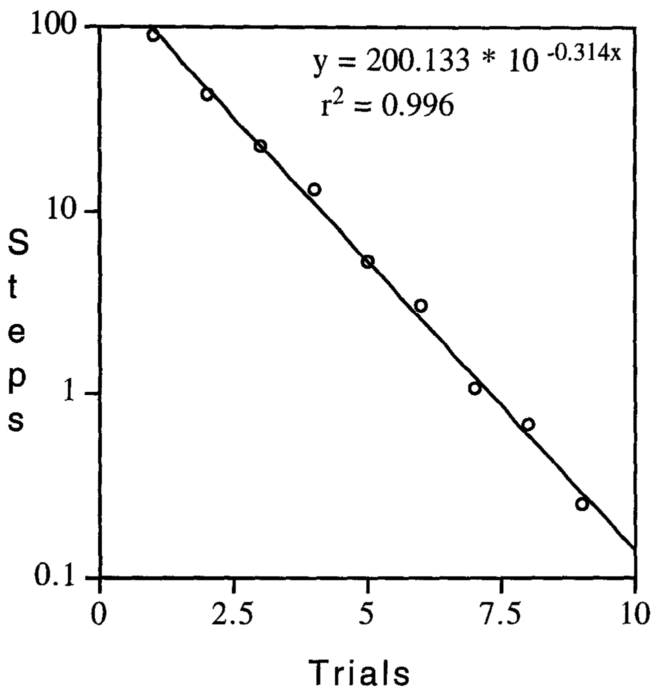

Figure 4. Curve for failure-driven learning.

For example, one natural question is to what extent the learning curve predicted by failure-driven learning is robust across different task environments. Intuition suggests that the complexity of the task might affect the shape of the curve, because more difficult tasks ought to cause slower learning. To investigate this question, we have to perform a *sensitivity experiment* (Schneider, 1988) i. e., derive the learning curve again and again for different values of the relevant complexity parameters (i. e., branching factor and path length.

This 2-by-2 (path length by branching factor) experiment was carried out with the abstract model, with path lengths either 10 or 40 and branching factor either 3 and 17. Figure 6 shows the result. The four learning curves are straight lines in log-normal coordinates. Failure-driven learning is robustly exponential across levels of complexity. Furthermore, the four curves are almost parallel. Hence, manipulating either dimension of task complexity produces parametric displacement of the curve, but leaves its shape unaffected. The two dimensions do not interact. The exponential nature of failure-driven

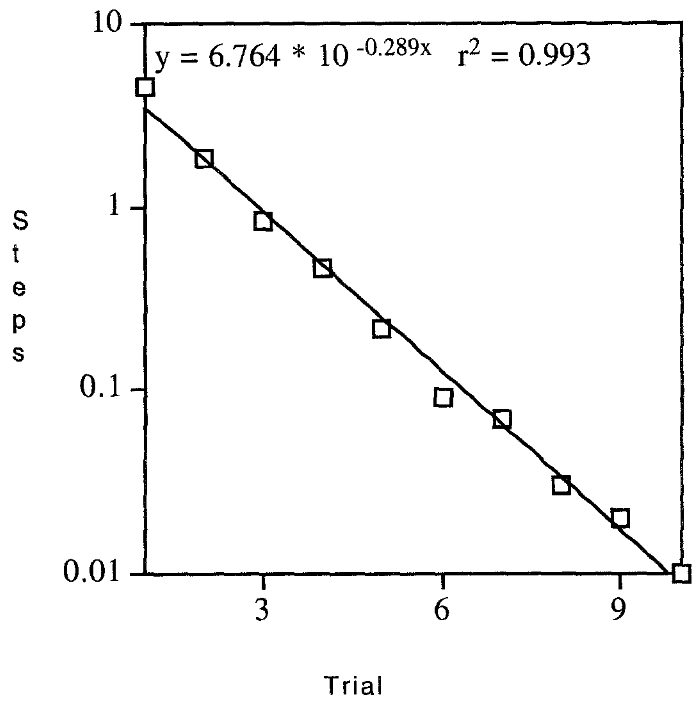

Figure 5. HS curve plotted in log-normal coordinates.

learning is robust across variations in task complexity (as measured by path length and branching factor).

A second natural question is how the efficiency of the learner affects the result. An alert learner has a greater probability of detecting and correcting an error. Intuitively, it is plausible that variation in learner efficiency affects the shape of the learning curve. This question is easily explored with the abstract model. The results presented so far were derived under the assumption that the learner catches and corrects every error he or she makes; let us call this 100% efficiency. Figure 8 shows the learning curve predicted under the alternative assumption that the learner has 50% efficiency, i. e., that he or she only detects or corrects every other error that he or she makes. Once again, we see that the learning curve is a straight line in log-normal coordinates. Hence, failure-driven learning is exponential across levels of learner efficiency.

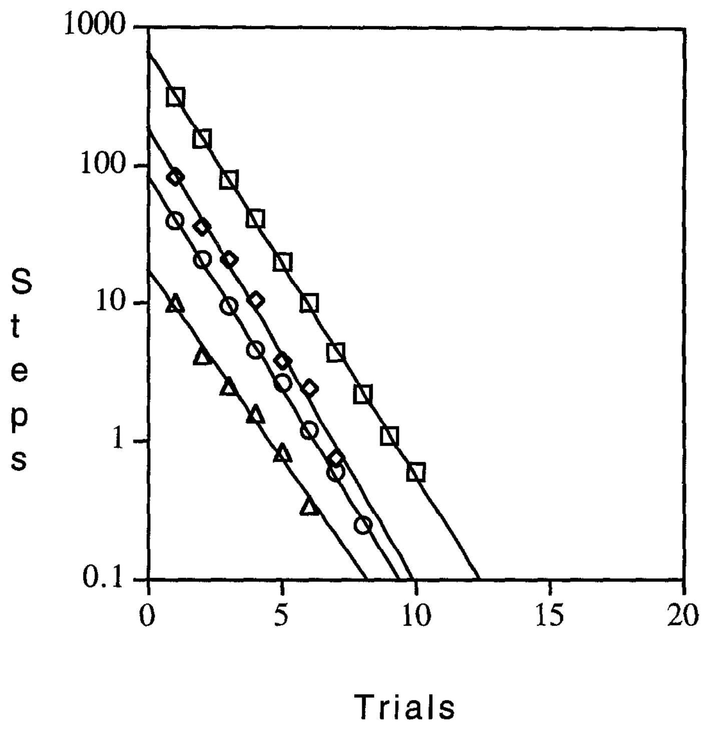

Figure 7. Effect of task complexity.

Discussion. Failure-driven learning appears to be robustly exponential across variations in both task and learner parameters. Empirical studies of skill acquisition (primarily in humans, but also in animals and social structures such as manufacturing plants) have shown that skill acquisition typically follows a power law, not an exponential curve (Lane, 1987; Newell & Rosenbloom, 1981). Hence, these simulations show that pure failure-driven learning does not predict the type of learning curve observed empirically. Contrary to the strong claims sometimes made for failure-driven leaning, people (and perhaps other adaptive agents as well) do not learn *solely* by reacting to their failures. (These results leave open the possibility that adaptive agents learn by some combination of error correction and one or more other learning mechanisms.)

The main point for present purposes is that the methodology of abstract modeling enables extensive exploration of the quantitative properties of particular learning mechanisms. The sensitivity experiments shown in Figures 7 and 8 would

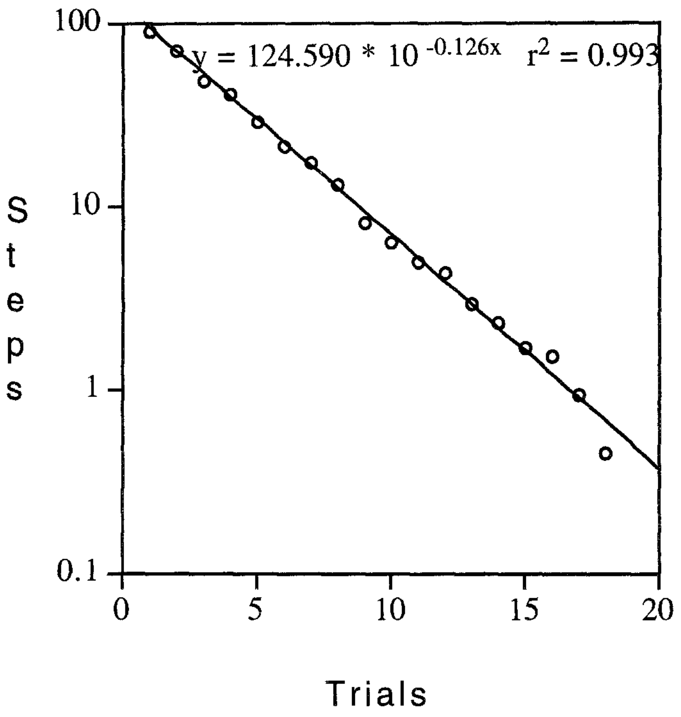

Figure 8. Failure-driven learning with 50% efficiency.

have required at least one man-year of work each if carried out with a symbolic model, but they took only between two and three weeks with the abstract model.

5. Summary

The development of theories of skill acquisition has stalled. Although a rich and interesting variety of hypotheses about the cognitive mechanisms underlying learning emerged in the decade 1979-1989, little current work is devoted to the analysis and evaluation of those ideas and the production of new ideas has slowed down. To transform a set of hypothesized change mechanisms into a theory of adaptive agents we need to explore the properties of those mechanisms, derive their behavioral implications and use them to explain particular regularities in adaptive behavior.

Symbolic computer simulation are not convenient tools for such explorations. In particular, the combination of brittleness and labor intensity discourages experimentation and exploration. In addition, the task specificity that comes with dependence on a knowledge base makes it difficult to interpret a symbolic model. In particular, there is no principled way of deciding which behavioral predictions are general and which are task specific.

Abstract computer models avoid these difficulties by combining the abstraction, elegance and simplicity of mathematical models with the flexibility and power of computational models. An abstract model is a computer program, but it is not an AI system. Specifically, abstract models mimic the structural features (and hence the quantitative behavior) of symbolic models but without explicitly representing the content of the relevant knowledge base. Abstract models are orders of magnitude easier to design, implement, debug and run and hence encourage extensive exploration of the behavioral predictions of particular adaptive mechanisms.

The issues to be explored in future include the following. First, what types of learning curves are generated by other learning mechanisms (chunking, planning, strengthening, etc.)? Second, which, if any, of the learning mechanisms proposed to date correctly predicts some *set* of behavioral regularities (as opposed to a single regularity): forgetting, overlearning, the difference between distributed and massed practice, error distributions and so on? Third, we want to investigate more complex models of task environment. How are various learning mechanisms affected by, for example, multiple correct paths, non-uniform branching or random fluctuations in strengths? Fourth, how do learning mechanisms interact with individual difference variables? Can the mechanisms proposed to date account for so-called aptitude-treatment interactions?

Finally, a particularly important question is how learning mechanisms interact with each other in the production of adaptive behavior. Are learning curves and other regularities invariant over composition of learning mechanisms? If not, what are the effects of combining multiple learning mechanisms? In a multi-mechanism system, observable changes in behavior are not direct expressions of any one of the relevant mechanisms, but a composite result of their interactions. In other sciences, complex interactions between underlying causal mechanisms map onto observable system properties in complex ways. The mapping from genes to phenotypic traits or from molecules to macroscopic properties of substances are good examples. There is no reason to expect the mapping from cognitive mechanisms to observable behavior to be any simpler. We are currently exploring the consequences of composing error correction with other learning mechanisms.

Acknowledgment

Preparation of this manuscript was supported by Grant No. N00014-93-I-1013 from the Cognitive Science Program of the Office of Naval Research (ONR). No endorsement should be inferred. Approved for public release; distribution unlimited.

References

Anderson, J. R., (Ed.), (1981). *Cognitive skills and their acquisition*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1987). Methodologies for studying human knowledge. *Behavioral and Brain Sciences, 10,* 467-505.

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Erlbaum.

Anzai, Y., & Simon, H. A. (1979) The theory of learning by doing. *Psychological Review, 86*, 124-140.

Chipman, S., & Meyrowitz, A., (Eds.), (1993). *Foundations of knowledge acquisition: Cognitive models of complex learning*. Boston, MA: Kluwer.

Frijda, N. J. (1967). Problems of computer simulation. *Behavioral Science, 122,* 59-31.

Holland, J., Holyoak, K., Nisbett, R., & Thagard, P. (1986). *Induction: The processes of inference, learning, and discovery*. Cambridge, MA: MIT Press.

Kieras, D. E. (1985). The why, when, and how of cognitive simulation: A tutorial. *Behavior Research Methods, Instruments, and Computers, 17*, 279-285.

Klahr, D., Langley, P., & Neches, R., (Eds.), (1987). *Production system models of learning and development.* Cambridge, MA: MIT Press.

Kodratoff, Y., & Michalski, R. S., (Eds.), (1990). *Machine learning: An artificial intelligence approach* (Vol 4). San Mateo, CA: Kaufmann.

Lane, N. (1987). *Skill acquisition rates and patterns: Issues and training implications*. New York, NY: Springer-Verlag.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological Science, 2*, 387-395.

Neches, R. T. (1982). Simulation systems for cognitive psychology. *Behavior Research Methods & Instrumentation, 14*, 77-91.

Newell, A. (1990). *Unified theories of cognition.* Cambridge, MA: Havard University Press.

Newell, A., & Rosenbloom, P. (1981). Mechanisms of skill acquisition and the law of practice. In J. Anderson, (Ed.), *Cognitive skills and their acquisition* (pp. 1-55). Hillsdale, NJ: Erlbaum.

Newell, A., Shaw, J. C., & Simon, H. A. (1958). Elements of a theory of problem solving. *Psychological Review, 65*, 151-166.

Newell, A., & Simon, H. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice-Hall.

Nilsson, N. J. (1971). *Problem-solving methods in artificial intelligence.* New York, NY: McGraw-Hill.

Ohlsson, S. (1988). Computer simulation and its impact on educational research and practice. *International Journal of Educational Research, 12*, 5-34.

Ohlsson, S. (1992). Artificial instruction: A method for relating learning theory to instructional design. In M. Jones & P. H. Winne, (Eds.), *Adaptive learning environments: Foundations and frontiers* (pp. 55-83). Berlin, Germany: Springer-Verlag.

Ohlsson, S. (1993). The interaction between knowlede and practice in the acquisition of cognitive skills. In S. Chipman and A. L. Meyrowitz, (Eds.), *Foundations of knowledge acquisition: Cognitive models of complex learning* (pp. 147-208). Boston, MA: Kluwer.

Ohlsson, S. (1994, February). *Learning from performance errors* (Technical Report, KUL-94-02). Pittsburgh, PA: University of Pittsburgh.

Ohlsson, S., Ernst, A. M., & Rees, E. (1992). The cognitive complexity of doing and learning arithmetic. *Journal of Research in Mathematics Eduation, 23*(5), 441-467.

Ohlsson, S., & Jewett, J. J. (1994, December). *Abstract models of learning from success and failure* (Technical Report). Pittsburgh, PA: University of Pittsburgh.

Ohlsson, S., & Rees, E. (1991a). The function of conceptual understanding in the learning of arithmetic procedures. *Cognition and Instruction, 8*, 103-179.

Ohlsson, S., & Rees, E. (1991b). Adaptive search through constraint violation. *Journal of Experimental and Theoretical Artificial Intelligence, 3*, 33-42.

Rosenbloom, P. (1986). The chunking of goal hierarchies. A model of practice and stimulus-response compatibility. In J. Laird, P. Rosenbloom, & A. Newell, *Universal subgoaling and chunking: The automatic generation and learning of goal hierarchies* (pp. 135-1282). Boston, MA: Kluwer.

Rosenbloom, P., & Newell, A. (1987). Learning by chunking: A production system model of practice. In D. Klahr, P. Langley,

& R. Neches, (Eds.), *Production system models of learning and development* (pp. 221-286). Cambridge, MA: MIT Press.

Schneider, W. (1988). Sensitivity analysis in connectionist modeling. *Behavior Research and Methods, Instruments, & Computers, 20*, 282-288.

Shrager, J., & Langley, P., (Eds.), (1990). *Computational models of scientific discovery and theory formation.* San Mateo, CA: Kaufmann.

Young, S. R. (1985). Programming simulations of cognitive processes: An example of building macrostructures. *Behavior Research Methods, Instruments, and Computers, 17*, 286-293.

Young, R., & O'Shea, T. (1981). Errors in children's subtraction. *Cognitive Science, 5*, 153-177.