Strategy Shifts Without Impasses: A Computational Model of the Sum-to-Min Transition

Randolph M. Jones
Kurt VanLehn

Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
JONES@CS.PITT.EDU
VANLEHN@CS.PITT.EDU

Abstract

The SUM-to-MIN transition that children exhibit when learning to add provides an ideal domain for studying naturally occurring discovery processes. We discuss a computational model that accounts for this transition, including the appropriate intermediate strategies. In order to account for all of these shifts, the model must sometimes learn without the benefit of impasses. Our model smoothly integrates impasse-driven and impasse-free learning in a single, simple learning mechanism.

Introduction

This paper discusses models for the well-known SUM-to-MIN transition (Ashcraft, 1982, 1987; Groen & Parkman, 1972; Groen & Resnick, 1977; Kaye, Post, Hall, & Dineen, 1986; Siegler & Jenkins, 1989; Svenson, 1975). This is an ideal domain for studying naturally occurring discovery processes (Siegler & Jenkins, 1989). When young children first learn to add two small numbers, they use the so-called SUM strategy. They create sets of objects to represent each addend, then count the objects in the union of the two sets. For instance, in order to solve $2 + 3$, the child says, "1, 2" while raising two fingers on the left hand; then "1, 2, 3" while raising three fingers on the right hand; then "1, 2, 3, 4, 5," while counting all the raised fingers. This strategy is called the SUM strategy because its execution time is proportional to the sum of the two addends. Older children use a more efficient strategy, the MIN strategy, whose execution time is proportional to the minimum of the two addends. In following this strategy, the child first announces the value of the larger addend, then counts on from it. For instance, in order to solve $2 + 3$, the child would say "3" then say "4,5" while raising two fingers on one hand.

Although the SUM strategy is taught in school, the MIN strategy appears to be invented by the children themselves. The best evidence for this comes from a longitudinal study by Siegler and Jenkins (1989). They interviewed 8 children weekly for 11 weeks, each time asking them to solve about 15 orally presented addition problems. After each problem, the child was asked how they got their answer. The child was also told whether their answer was correct, and was given a gold star if it was. Videotapes were analyzed and the child's behavior on each problem was classified according to the strategy used. As far as Siegler and Jenkins could determine, the only instruction that the subjects received during this period was their school's normal instruction on the SUM strategy. Nonetheless, 7 of the 8 children began to use the MIN strategy. Moreover, they appear to have discovered it during the video-taped sessions. The tapes make it clear that they received no help from the experimenter, so the MIN strategies appear to have been invented by the subjects themselves.

The issue addressed by this paper is explaining how children discover the MIN strategy. This particular discovery presents a challenge for current theories of learning. Although a variety of learning methods have been proposed, many of them are triggered when the problem solver reaches an impasse, and yet Siegler and Jenkins found no signs of impasses during the discovery of the MIN strategy. The exact definition of "impasse" depends on the problem-solving architecture, but roughly speaking, an impasse occurs whenever the solver has a goal that cannot be achieved by any operator that is believed to be relevant to the task at hand. The essential idea of impasse-driven learning is to resolve the impasse somehow, then store the resulting experience in such a way that future impasses will be avoided or at least handled more efficiently. Many systems use impasse-driven learning, including SWALE (Schank, 1986), OCCAM (Pazzani, Dyer & Flowers, 1986), LPARSIFAL (Berwick, 1985), SOAR (Newell, 1990), SIERRA (VanLehn, 1990), and CASCADE (VanLehn & Jones, in press).

Siegler and Jenkins looked specifically for signs of impasses and found none. In particular, they designed some of the problems to cause impasses by making one of the addends very large (e.g., $23 + 1$). They found that "The specific problems on which the children first used the MIN strategy were $2+5$, $4+1$, $3+1$, $1+24$, $5+2$, and $4+3$. These problems did not deviate from the characteristics of the overall set in any notable way."(p. 67) In fact, some of the children had earlier successfully solved exactly the same problem that they were working on when they discovered the MIN strategy. Although the impasse problems did cause subjects who had already invented the MIN strategy to start using it more frequently, the problems did not cause those who had not invented the strategy to do so.

Siegler and Jenkins sought signs of impasses by examining solution times and errors in the vicinity of the discovery events. Solution times were longer than normal for the problems where the discovery occurred and for the problems immediately preceding the discovery trial. This might suggest some kind of impasse, but error rates indicated that the problems themselves were not particularly difficult. Siegler and Jenkins suggest

This research benefited from discussions with Jeff Schlimmer and Bob Siegler. It was supported in part by contract N00014-88-K-0080 from the Office of Naval Research, Cognitive Sciences Division, and a postdoctoral training grant from the Department of Health and Human Services.

a reconciliation between their findings and impasse-driven learning theories:

> Two types of strategy changes can be distinguished: changes in which the main difference between the strategies is in the answers themselves, and changes in which the main differences are not in the answers that are generated but rather in the efficiency with which answers are generated and/or the aesthetic appeal of the procedures. The first type of strategy change may occur primarily as a result of encountering impasses, but the second may typically occur for other reasons. (p. 104)

One model in the literature effects a strategy change whenever it detects an opportunity for improving the efficiency of a procedure. HPM (Neches, 1987) keeps a complete trace of its processing, which is constantly monitored by heuristics such as: "If a subprocedure produces an output, but no other subprocedure receives that result by the time the overall procedure finishes, then modify the overall procedure to eliminate the superfluous computation." Neches demonstrated that this heuristic and two others sufficed for changing the SUM strategy into the MIN strategy.

Enroute, two transitional strategies are necessarily produced by HPM. Siegler and Jenkins sought evidence for these transitional strategies in their data. One of the strategies occurred 6 times, all in the protocol of the same subject. Moreover, all these instances occurred *after* the MIN strategy was invented. The second transitional strategy predicted by Neches did not appear at all. These unfulfilled predictions cast doubt on the HPM model. The model itself, as Neches noted, "assumes the relative accessibility of extremely detailed information about both on-going process and related past experiences. How can this be reconciled with known limitations on the ability to report this information?" (p. 213) Although HPM is computationally sufficient to produce the SUM-to-MIN transition, it makes dubious empirical and mneumonic assumptions.

The objective of this research has been to generate a computationally sufficient account of the SUM-to-MIN transition that produces only observed transitional strategies and makes plausible demands on memory. The result is a problem solver called GIPS (General Inductive Problem Solver). This paper describes GIPS and its account of the strategy shifts observed by Siegler and Jenkins (1989).

GIPS is a generalized means-ends analysis (MEA) problem solver (Jones, 1989) whose primary learning mechanism is based on Schlimmer's (1987; Schlimmer & Granger, 1986a, 1986b) STAGGER system, which uses a probabilistic induction technique to learn concept descriptions from examples. Other systems have combined inductive concept learners with problem solvers (e.g., Langley, 1985; Mitchell, Utgoff & Banerji, 1983), but they acquire only search-control knowledge: concepts that indicate which operator to select when several operators match the current goal. GIPS modifies the descriptions of the operators themselves as well as the heuristics for selecting operators. Both types of learning play crucial roles in making the SUM-to-MIN transition.

Following a brief description of GIPS (see Jones & VanLehn, 1991, for a complete description) is the main section of this paper, which presents the GIPS account of the SUM-to-MIN transition.

The General Inductive Problem Solver

GIPS' basic problem-solving algorithm is a generalized version of MEA borrowed from the EUREKA system (Jones, 1989). It is based on trying to achieve a state change. The desired change is represented by a TRANSFORM, which is simply a pair consisting of the current state and some goal conditions. In order to achieve this transformation, GIPS selects an operator and attempts to apply it. If the operator's preconditions are met, it is executed and the current state changes. If some of the preconditions are not met, a new TRANSFORM is created with them as the goals. When this transformation is achieved, GIPS returns to the old TRANSFORM and attempts again to apply the operator. So far, this is just the standard MEA algorithm, but GIPS adds two important differences.

In standard MEA, operators are selected if they reduce the difference between the current and goal states. In GIPS, selection is determined by *selection concepts*. Each operator has a concept that indicates when it should be selected. If the concept depends mostly on the current state of the TRANSFORM, then the operator will act like a forward-chaining inference rule and fire whenever the state is appropriate, regardless of the current goals. If the concept depends mostly on the goals of the TRANSFORM, then it will act like a backward-chaining inference rule. Typically, forward and backward operators intermingle during problem solving, yielding a psychologically plausible blend of goal-directed and opportunistic behavior.

Each operator has a selection concept. The concept is represented as set of literals (predicates that may or may not be negated), and each literal has two values associated with it: its sufficiency and its necessity. In order to evaluate the worth of selecting an operator, GIPS matches the literals against the current TRANSFORM. It determines the subset of literals that match (M) and fail to match (F), then calculates

$$Odds(C) \prod_{L \in M} sufficiency(L) \prod_{L \in F} necessity(L),$$

where $Odds(C)$ is the prior probability that the concept is worth selecting. This is the same formula used by STAGGER, Schlimmer's (1987) propositional concept formation system, to estimate the probability that a given object is an instance of a particular concept. When all the operators have been matched and their worth has been calculated, GIPS chooses the one with the highest rating.

GIPS adjusts its selection concepts on the basis of its successes and failures while solving problems. When a problem is finally solved, for each operator along the solution path, GIPS adjusts the sufficiency and necessity values so that the operator will be rated even higher the next time a similar TRANSFORM occurs. For each operator that initiated a failure path, GIPS adjusts the values in its selection concept so that it will receive a lower value next time. In order to learn, GIPS must store the solution path and every operator that led off it. However, as soon as the problem is finished and the updating is completed, this information can be forgotten. HPM (Neches, 1987) must store the whole search tree, not just the solution path, for an indefinite period. Holland, Holyoak, Nisbett, and Thagard (1986) describe a technique, called the bucket brigade algorithm, that achieves the same kind

of update as GIPS without storing any of the solution path.

The mechanism described so far can learn search-control knowledge, as do many other machine learning systems. In order to make the SUM-to-MIN transition, the system must modify the preconditions of operators as well. In standard MEA, after an operator has been selected, its preconditions are matched to the current state. If all of them match, the operator is executed. If some do not match, they become subgoals. Thus, preconditions determine the goal structure of the problem solving. GIPS takes an indirect approach to adjusting preconditions. In addition to the selection concept, GIPS provides a second concept, called the *execution concept*, which serves as a repository for its experience in attempting to execute the operator. When values in the execution concept cross a threshold, appropriate modifications are made to the preconditions. Thus, GIPS only modifies preconditions when warranted by a great deal of experience. The next two paragraphs describe exactly how execution concepts are used.

When an operator is selected, the system matches the execution concept to the current TRANSFORM in the same way that selection concepts are matched. If the calculation returns a value greater than 1, GIPS attempts to execute the operator. If the value is less than 1, GIPS follows the standard MEA practice of matching the preconditions of the operator to the current state and setting up subgoals for the unmatched ones.

When GIPS attempts to execute an operator, it has no arms and eyes so it cannot tell if its attempt succeeds, thus it asks the user. If an operator succeeds, the values in the execution concept are updated appropriately and the new state becomes current. If an operator fails, the values are updated and the system follows the standard MEA routine of matching preconditions and setting up subgoals.

When the sufficiency value for a literal in the execution concept crosses a threshold, that literal is added to the operator's preconditions. Thus, GIPS only adds a literal when that literal has been found time and time again to be present when the operator executes. However, GIPS is quick to remove a literal from the preconditions if it ever finds that the literal is not actually necessary for the execution of the operator. If the system successfully executes an operator and not all of the preconditions are satisfied, it removes the unmatched literals. There is no sense continuing to subgoal for an unmatched literal if that literal is not in fact necessary for executing the operator.

Representation of the Addition Domain

GIPS describes the world as a set of relations between objects. In the addition domain, these objects and relations include the numbers that are part of the problem, the state of the problem solver's "hands" while it is adding, and the value of a counter that the problem solver keeps "in its head."

GIPS requires 16 operators to represent the addition domain. There are two particular operators, which we refer to as the END-COUNT operators, that are involved in most of the strategy shifts. For future reference, the series of preconditions that the LEFT-END-COUNT operator acquires appears in Table 1. The 16 operators' selection concepts were initialized so that the system generates the SUM strategy. The literals of each operator's selection concept were the preconditions and the goals that the operator could satisfy. The necessity and sufficiency of these literals were set so that they would be retrieved in either a backward-chaining or forward-chaining fashion, depending on the role of the operator in the domain.

Table 1. A series of preconditions for LEFT-END-COUNT.

SUM strategy (a):
`Raising(Lefthand)`
`Counting(Lefthand)`
`Assigned(Lefthand,=Value)`
`Counter-value(=Value)`

SUM strategy (b):
`Raising(Lefthand)`
`Counting(Lefthand)`
`Assigned(Lefthand,=Value)`
`Counter-value(=Value)`
`Raised-fingers(Lefthand,=Value)`

SHORTCUT SUM strategy (c):
`Raising(Lefthand)`
`Counting(Lefthand)`
`Assigned(Lefthand,=Value)`
`Raised-fingers(Lefthand,=Value)`

FIRST strategy (d):
`Raising(Lefthand)`
`Counting(Lefthand)`
`Assigned(Lefthand,=Value)`

Strategy Acquisition in the Addition Domain

This section presents GIPS' behavior through a series of different strategies for adding numbers. These strategy shifts arise from the learning algorithm incorporated into the system, and they correspond well with the shifts observed by Siegler and Jenkins. Siegler and Jenkins classified their subjects' behavior into 8 strategies, of which 4 were based on counting (the others involved various kinds of recognition and guessing). In this section, we describe each of the 4 counting strategies in the order in which they generally appear. However, it is important to note that children always intermingle their strategies, sometimes even on a trial by trial basis. We will discuss the issue of strategy variability in the next section.

The SUM Strategy

GIPS' initial strategy for addition is the SUM strategy. The first thing the system does is assign an addend to each hand. For example, when adding 2 and 3, the system may assign the number 2 to the left hand and the number 3 to the right hand. However, in this strategy the order of the addends does not make a difference, so it could just as easily have switched them.

Next, the system begins its procedure of counting out a set of fingers on each hand. To accomplish this task the END-COUNT operators initially use a counter to determine when a hand is finished being counted out. For example, the preconditions of LEFT-END-COUNT demand that the system be raising fingers on the left hand, and that the value of the counter be

equal to the value of the left-hand addend. These preconditions are set up as subgoals, causing the selection of the START-RAISE and START-COUNT operators, which initialize the forward-chaining procedure of raising and counting fingers one at a time. These operators execute alternately until LEFT-END-COUNT can execute, when the correct number of fingers have been counted on the left hand.

After the left hand has been counted, the CLOBBER-COUNTER operator immediately executes. This operator executes when all the fingers of a hand have been raised along with a running count. Its effects are to zero the value of the counter to prepare it for the next hand, and to mark the current hand as *uncounted*, because the counter's value has been changed. This entire procedure then repeats with the right hand.

After both hands have been counted, DETERMINE-ANSWER checks whether it can execute. It can only execute if both hands are marked as *counted*, but execution of CLOBBER-COUNTER has caused this to be false. Therefore, the system again attempts to count up fingers on each hand, this time marking fingers that are already raised. For this procedure no CLOBBER-COUNTER is necessary, because the number of raised fingers (rather than the value of the counter) is used to terminate the count for each hand. Finally, after each hand has been counted for the second time, GIPS announces the answer.

As the system repeatedly solves addition problems, it continuously updates the execution concepts for the END-COUNT operators. After a while, the concept encodes several regularities that are always true when these operators execute. For example, there are always two addends in the problem description, and the number of "marked" fingers is always zero. Most importantly, however, the concept encodes the number of raised fingers is always equal to the counter value (which in turn is equal to the goal value for counting an addend). Thus, these literals eventually get added into the preconditions for the END-COUNT operators (see Table 1(b)). This action alone does not change the system's outward behavior, but it proves important for later strategies.

The SHORTCUT SUM Strategy

After new preconditions have been added and a number of addition problems have been solved, the new literals in the system's execution concepts for LEFT-END-COUNT and RIGHT-END-COUNT become strong enough that GIPS attempts to execute the operators earlier than usual. At some point, it thinks that the operators should execute when the number of fingers raised on a hand is equal to the goal value even though the system has not yet incremented its count for the last finger. It turns out that the system can successfully solve the addition problem even if it executes this operator prematurely, so it deletes the condition that the current counter value must be equal to the goal value in the preconditions of the END-COUNT operators (see Table 1(c)).

This change has a direct effect on GIPS' behavior. When attempting to apply LEFT-END-COUNT, the value of the counter no longer appears in the preconditions, so it is not posted as a subgoal. This means that the START-COUNT operator is no longer selected. Thus, a running count is still kept while raising fingers, but the counter is not marked for use as the termination criterion. This means that CLOBBER-COUNTER will not fire, and that leads to two changes in strategy. First, the counter is not reset to zero after counting the left hand, and counting continues from the left hand's final value. Second, the hands are not marked as *uncounted*, so there is no need to count up the raised fingers again after the two hands have initially been counted. This behavior corresponds to the SHORTCUT SUM strategy, which was invented by all 8 of Siegler and Jenkins' subjects.

The SHORTCUT MIN Strategy

The next shift leads to an intermediate strategy between SHORTCUT SUM and MIN, which we call SHORTCUT MIN. Although Siegler and Jenkins do not classify SHORTCUT MIN as a distinct strategy from SHORTCUT SUM, they do note (p. 119) that some of their subjects begin to switch addends during SHORTCUT SUM so that they start counting with the larger addend on the left hand, rather than just picking whichever addend appears first in the problem. GIPS can also account for this behavior.

An important feature of the SHORTCUT SUM strategy is that the problem solver's counter value is not equal to the number of fingers being raised on the right hand (i.e., the second hand). We hypothesize that this causes interference and subsequent failure. Such interference would not occur with the left hand, because the number of raised fingers in the SHORTCUT SUM strategy is always equal to the value of the counter for that hand. We simulated interference between the value of the counter and the number of fingers raised on the right hand by causing GIPS to fail sometimes during the SHORTCUT SUM strategy when it decided to count the larger addend on its right hand. This caused the system to update the selection concept for the operator that initially assigns an addend to each hand, so that it would prefer to count the smaller addend on the right hand.

The MIN Strategy

The final strategy shift occurs in a similar manner to the shift from SUM to SHORTCUT SUM. At this point, GIPS has attempted to execute the END-COUNT operators at various times and has been given feedback each time as to whether it would be able to solve the current problem if it executed the operator at that time. Thus, it is slowly learning a "good" concept for when the END-COUNT operators are executable. One of the things that proves to be true every time these operators are executed is that the goal value for counting out a hand is equal to the addend assigned to that hand.

Eventually, the system attempts to fire the LEFT-END-COUNT operator without having raised any fingers at all. When it succeeds by doing this, it deletes the precondition that the number of fingers raised on the hand be equal to the goal value (see Table 1(d)). The system has learned that it can simply start counting from the goal value for the left hand rather than starting from zero. GIPS also attempts to execute the RIGHT-END-COUNT operator early, but this leads to failure. Thus, the system begins to exhibit the MIN strategy, in which the largest number (the left-hand number) is simply announced and used to continue counting the smaller number as in the SHORTCUT MIN strategy.

The FIRST Strategy

The only other counting strategy found by Siegler and Jenkins is the FIRST strategy. It was used on only 6 trials, all by the same subject. FIRST is similar to the MIN strategy, except that it does not assign the larger addend to the left hand. Rather, it starts with whichever addend is presented first, and continues counting with the second. In GIPS, this strategy follows from the SHORTCUT SUM strategy when the system does not learn about ordering the addends. While using the FIRST strategy, the system can still eventually generate the MIN strategy through the same type of failure-driven learning that leads from SHORTCUT SUM to SHORTCUT MIN.

Summary and Discussion

Both the SUM strategy and the MIN strategy have three main subgoals: to represent one addend, to represent the the other addend, and to count the union of the representations. The SUM-to-MIN transition involves three independent modifications to the SUM strategy: (1) The subgoal of representing one addend changes from explicitly constructing a set of objects to simply saying the addend. (2) The order of addends is made conditional on their size so that the larger addend is represented by the easier process. (3) The process of representing the other addend is run in parallel with counting up the union. In the SUM strategy, representing the other addend is finished before counting up the union begins.

The GIPS account for each of these transitions is as follows. The first transition is caused by correlational learning of preconditions. GIPS keeps track of which literals in the situation are correlated with the final achievement of the goal of representing the first addend. Eventually, it considers these correlated literals to be just as essential as the originally specified preconditions. It eventually discovers that the originally specified preconditions can be ignored as long as the correlated literals are achieved.

The second transition is caused by normal failure-driven learning. The system uses two apparently equivalent methods, but persistent errors in one of them causes the other one to eventually dominate.

The third transition again involves a correlational type of learning. The COUNT operator is responsible for incrementing the oral counter. Initially, it is selected only when the subgoal of counting-up an addend is present. Eventually, correlated relations that are present in the current state (i.e., that a finger has just been raised) come to dominate the selection concept, and the operator becomes a forward-chaining operator. Basically, the person has developed the habit of counting whenever they raise a finger even if that count doesn't serve any direct purpose. Although GIPS could learn this habit, we actually gave COUNT a forward-chaining selection concept in our experiments in order to save time. Given this habit, it serendipitously achieves the goal of counting the union even when the counter is no longer used to represent the second addend.

Although this summary leaves out some crucial details, it makes it clear that correlational learning is crucial to the GIPS account for the first and third transitions. Ordinary failure driven learning can handle the second.

Our analysis with GIPS helps clarify several important, general issues about strategy change. Siegler and Jenkins observe that, "Not one child adopted a strategy that could be classified as indicating a lack of understanding of the goals of addition." (p. 107) In this respect, the subjects are similar to those of Gelman and Gallistel (1978) who found that very young children would invent correct strategies for counting a set of objects even when unusual constraints were placed on them to thwart their normal strategy.

The initial knowledge given to GIPS does not include any explicit principles of addition or counting. As far as it is concerned, the SUM strategy is just a song that has to be sung the right way. How then does it avoid developing bad strategies? In the Siegler and Jenkins study, students were told after each trial whether they got the problem right. This kind of feedback is crucial to GIPS' learning. GIPS occasionally attempts to execute an operator in situations that would produce a wrong answer. If it were not told that the execution was wrong, it would develop wrong strategies. This demonstrates that an innate understanding of addition is not necessary for a computationally sufficient account of the observed competence.

A common misconception about discovery is that a newly discovered strategy or concept instantly and totally supplants its predecessor. In all protocol-based studies of discovery (e.g., Kuhn, Amsel, & O'Laughlin, 1988; Siegler & Jenkins, 1989; VanLehn, in press), the transition between the old strategy and the new one is gradual. We have not tried to model the gradual transition to the use of the MIN strategy with GIPS because doing it right would require implementing several memory-based strategies. However, it is clear that the probabilistic nature of GIPS' selection and execution concepts would tend to predict a gradual transition.

Starting in the eighth week of the study, Siegler and Jenkins began including "impasse problems," such as 2+23. They had hoped that these would encourage discovery of the MIN strategy, but they did not, for no child first used the MIN strategy on an impasse problem. However, children who had already discovered the MIN strategy began to use it much more frequently on the impasse problems and even on the non-impasse problems that followed the eighth week. GIPS would tend to do the same thing if it were given impasse problems. The larger addend would invite errors during the SHORTCUT SUM and FIRST strategies, which would lower the values of their selection concepts. The inclusion of impasse problems would not effect the error rate of the MIN strategy, so it would gradually become the preferred strategy for all counting trials.

Siegler and Jenkins noticed that some children were consciously aware that they had invented a new strategy in that they could explain it on the first trial where they used it, and some even recognized that it was a "smart answer," in the words of one child. Other children denied using the MIN strategy even when the videotape showed that they had used it. Siegler and Jenkins divided children into those who seemed conscious of the strategy and those who did not, and measured the frequency of their subsequent usage of the MIN strategy. The high awareness group used the MIN strategy on about 60% of the trials where they used any counting strategy. The low awareness group used the MIN strategy on less than 10% of the trials. This suggests that being aware of a newly discovered strategy facilitates subsequent usage of it.

This finding cannot be modeled by GIPS because GIPS has no way to distinguish a strategy that can be explained from one that is inaccessible to consciousness. However, the finding could probably be modeled by combining GIPS with a symbolic example-learning system such as CASCADE (VanLehn & Jones, in press; VanLehn, Jones, & Chi, 1991). In the new system, GIPS would discover a strategy and store a trace of the strategy's actions in memory. This trace would be used as an example to be explained by the second system. If enough of the trace can be recalled for the explanation to succeed, it annotates the steps in the trace and perhaps the operators whose executions produced the steps. These elaborations would make it easier to retrieve the modified operators from memory, and perhaps help in assigning credit and blame, thus speeding the adjustment of the preconditions, selection, and execution concepts. These influences would increase the usage of the new strategy on subsequent problems.

To summarize, GIPS achieves its main research objective, providing a computational account of the several strategy shifts observed during the SUM-to-MIN transition. It uses plausible local processes, rather than global optimization techniques as required by the HPM system. In addition, GIPS uses modest amounts of storage, in contrast to HPM, which stores complete solution traces for indefinite periods. Most importantly, GIPS produces all and only the transitional strategies observed in the Siegler and Jenkins study. The GIPS analysis solves a number of puzzles raised by the Siegler and Jenkins study. These include the ability to make significant strategy shifts without impasse-driven learning, and to avoid inventing bad strategies without assuming innate knowledge of the principles of addition. Thus, GIPS provides a plausible, computationally sufficient account of the discovery of the MIN strategy. However, Siegler and Jenkins produced a second set of findings on the gradual increase in usage of the newly discovered strategy. We have not yet tried to model these findings, but GIPS seems to provide an appropriate framework for doing so.

References

Ashcraft, M. H. (1982). The development of mental arithmetic: A chronometric approach. *Developmental Review, 2*, 213–236.

Ashcraft, M. H. (1987). Children's knowledge of simple arithmetic: A developmental model and simulation. In C. J. Brainerd, R. Kail, & J. Bisanz (Eds.), *Formal methods in developmental psychology.* New York: Springer-Verlag.

Berwick, R. (1985). *The acquisition of syntactic knowledge.* Cambridge, MA: MIT Press.

Gelman, R., & Gallistel, C. R. (1978). *The child's understanding of number.* Cambridge, MA: Harvard University Press.

Groen, G. J., & Parkman, J. M. (1972). A chronometric analysis of simple addition. *Psychological Review, 79*, 329–343.

Groen, G., & Resnick, L. B. (1977). Can preschool children invent addition algorithms? *Journal of Educational Psychology, 69*, 645–652.

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

Jones, R. M. (1989). *A model of retrieval in problem solving.* Doctoral dissertation, University of California, Irvine.

Jones, R. M., & VanLehn, K. (1991). *The Gips model of strategy acquisition.* Manuscript submitted for publication.

Kaye, D. B., Post, T. A., Hall, V. C., & Dineen, J. T. (1986). The emergence of information retrieval strategies in numerical cognition: A development study. *Cognition and Instruction, 3*, 137–166.

Kuhn, D., Amsel, E., & O'Laughlin, M. (1988). *The development of scientific thinking skills.* New York: Academic Press.

Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science, 9*, 217–260.

Mitchell, T. M., Utgoff, P. E., & Banerji, R. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In R. S. Michalski, J. G. Carbonell, T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

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

Newell, A. (1990). *Unified theories of cognition: The William James lectures.* Cambridge, MA: Harvard University Press.

Pazzani, M., Dyer, M., & Flowers, M. (1986). The role of prior causal theories in generalization. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 545–550). Philadelphia: Morgan Kaufmann.

Schank, R. (1986). *Explanation patterns: Understanding mechanically and creatively.* Hillsdale, NJ: Lawrence Erlbaum.

Schlimmer, J. C. (1987). Incremental adjustment of representations for learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 79–90). Irvine, CA: Morgan Kaufmann.

Schlimmer, J. C., & Granger, R. H., Jr. (1986a). Beyond incremental processing: Tracking concept drift. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 502–507). Philadelphia: Morgan Kaufmann.

Schlimmer, J. C., & Granger, R. H., Jr. (1986b). Incremental learning from noisy data. *Machine Learning, 1*, 317–354.

Siegler, R. S., & Jenkins, E. (1989). *How children discover new strategies.* Hillsdale, NJ: Lawrence Erlbaum.

Svenson, O. (1975). Analysis of time required by children for simple additions. *Acta Psychologica, 39*, 289–302.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions.* Cambridge, MA: MIT Press.

VanLehn, K. (in press). Rule acquisition events in the discovery of problem solving strategies. *Cognitive Science.*

VanLehn, K., Jones, R. M., & Chi, M. T. H. (1991). *A model of the self-explanation effect.* Manuscript submitted for publication.

VanLehn, K., & Jones, R. M. (in press). Integration of explanation-based learning of correctness and analogical search control. In S. Minton & P. Langley (Eds.), *Proceedings of the symposium on learning, planning and scheduling.*