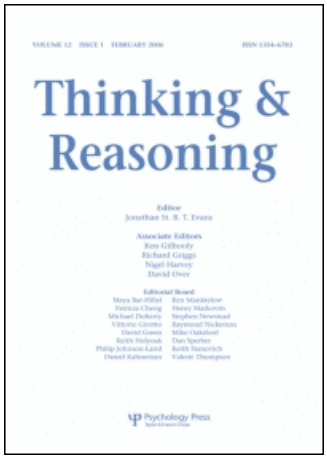


Thinking & Reasoning
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713685607

Cognitive components of troubleshooting strategies

To cite this Article: , 'Cognitive components of troubleshooting strategies', Thinking & Reasoning, 13:2, 134 - 163
To link to this article: DOI: 10.1080/13546780600750641
URL: http://dx.doi.org/10.1080/13546780600750641

THINKING & REASONING, 2007, 13 (2), 134–163

Psychology Press
Taylor & Francis Group

Cognitive components of troubleshooting strategies

Leo Gugerty
Clemson University, Clemson, SC, USA

This study investigated the kinds of knowledge necessary to learn an important troubleshooting strategy, elimination. A total of 50 college-level students searched for the source of failures in simple digital networks. Production system modelling suggested that students using a common but simpler backtracking strategy would learn the more advanced elimination strategy if they applied certain domain-specific knowledge and the general-purpose problem-solving strategy of reductio ad absurdum. In an experiment, students solved network troubleshooting problems after being trained with either the domain-specific knowledge, the reductio ad absurdum strategy, both types of knowledge, or neither. Students needed both the domain-specific and general knowledge identified by the models in order to significantly increase their elimination use.

What are the cognitive components of problem-solving strategies? And how do people learn new strategies? The research described here focuses on cognitive components and learning of key troubleshooting strategies. Troubleshooting is an important kind of problem solving that is frequent in everyday life, since few of us have achieved perfection yet. We are doing troubleshooting whenever we notice, find the cause of, and fix a problem in a system or device. To get an idea of the troubleshooting strategies investigated here, consider the following problem: "You turn your desk lamp on and it does not work. The lamp and a radio are both connected to the wall outlet by a single extension cord". One strategy applicable to this problem is *backtracking*, in which one reasons backwards from the

Correspondence should be address to Leo Gugerty, 418 Brackett Hall, Psychology Department, Clemson University, Clemson, SC 29634-1355, USA.
E-mail: gugerty@clemson.edu

I would like to thank Gary Olson, Dave Meyer, and John Laird for their help in this research, and Martin Ippel, Valerie Shute, and Joshua Hurwitz for comments on the paper. This study was part of a doctoral dissertation at the University of Michigan, Department of Psychology.

http://www.psypress.com/tar DOI: 10.1080/13546780600750641

symptoms (i.e., faulty system outputs), checking system components that are causally connected to the symptoms. Thus, you might check the lightbulb, the power cord, the extension cord, the wall outlet, and the circuit breaker. However, a more efficient strategy is available. Suppose you turn on the radio and it works. This *eliminates* from consideration the extension cord and any component prior to it, thus saving you the trouble of checking a number of components.

Notice the efficiency of the elimination strategy as compared to backtracking. A quick test of the system's outputs (turning on the radio) can save you the trouble of more time-consuming tests and replacements of system components. When troubleshooting complex devices and systems that involve many components, strategies like elimination are essential to efficient troubleshooting. Despite this, pilot testing for this study showed that many college students failed to use elimination on a simple troubleshooting task, and instead relied on backtracking. In a similar vein, Kurland and Tenney (1988) found that novice radar technicians sometimes failed to use elimination, while experts used it regularly. Given these findings and the importance of the elimination strategy to effective troubleshooting, the study presented here focused on how college students who know backtracking can learn elimination. This study identified some of the knowledge, or cognitive components, needed to learn elimination and showed that teaching this knowledge does increase use of elimination.

Before describing the study, some of the theoretical and methodological issues that guided its design will be outlined. One theoretical issue concerns the mechanisms of strategy learning; in particular, the contribution of domain-specific and general-purpose knowledge to strategy learning.

DOMAIN-SPECIFIC VS GENERAL KNOWLEDGE

Psychologists have debated the relative importance of domain-specific and general-purpose knowledge in problem-solving skill (Glaser, 1984; Sternberg, 1985). In the case of troubleshooting, this issue leads to questions such as whether general-purpose troubleshooting skills exist that are applicable in diverse domains, and how domain-specific knowledge affects troubleshooting performance. A considerable amount of research points to the importance of domain-specific knowledge in problem solving. Studies by Chi, Glaser, and Rees (1981) and Chase and Simon (1973) suggest that expert – novice differences in performance on physics and chess problems are due to differences in domain knowledge, rather than general problem-solving skill. However, even experts cannot rely on automatic application of domain knowledge when they face unfamiliar domain problems or learn new problem-solving strategies applicable in the domain. Psychologists have

recognised that situations such as these can only be understood in terms of the interaction of general and domain-specific knowledge (Alexander & Judy, 1988; Kuhn, 1991; Siegler, 1989; Sternberg, 1989; Zimmerman, 2000). For example, Siegler (1989) has developed a model of strategy choice that has been successfully applied to a variety of domains (e.g., arithmetic, reading, time telling). According to Siegler, the general strategy-choice mechanism is the same in all these domains, yet using this mechanism to make choices in a particular domain requires domain-specific knowledge.

Troubleshooting is an excellent type of problem solving for studying the interaction of domain-specific and general knowledge. Troubleshooting strategies such as backtracking and elimination are general purpose, in the sense that they can be applied in any troubleshooting domain, including medical diagnosis, computer program debugging, and electronic troubleshooting. However, domain-specific knowledge is necessary to apply these strategies in a particular domain. The domain-specific knowledge needed for elimination consists of mental-model knowledge of the structure (topography), functions, and internal states of the malfunctioning system and its components. Based on observational studies of technical troubleshooters, Schaafstal, Schraagen, and van Berlo (2000) suggest that novice troubleshooters need additional training in both domain-specific knowledge and general troubleshooting strategies.

EMPIRICAL STUDIES OF STRATEGY LEARNING

Studies of strategy-learning mechanisms focus on two ways to learn new strategies, via discovery learning (Siegler & Jenkins, 1989, VanLehn, 1991) or instruction (Pressley, Woloshyn, Lysynchuk, Martin, Wood, & Willoughby, 1990). Studies of strategy discovery in the domains of children's arithmetic (Siegler & Jenkins, 1989), puzzle problems (VanLehn, 1989, 1991), and scientific reasoning about gears (Metz, 1985) have found that the initial use of a strategy is preceded by pauses and signs of cognitive effort, and followed by a long period in which the strategy is used infrequently, with earlier strategies still being used.

A number of studies have been conducted in which problem-solving strategies were explicitly instructed (Carlson, Lundy, & Schneider, 1992; Rouse & Hunt, 1984; Shepherd, Marshall, Turner, & Duncan, 1977). Two of these studies have shown that explicit training in elimination improves participants' performance on the type of troubleshooting task used in this research (Brooke, Cook, & Duncan, 1983; Goldbeck, Bernstein, Hillex, & Marx, 1953). The Goldbeck et al. study found that training in the half-split strategy did not improve troubleshooting performance unless it was combined with training in elimination. In the half-split strategy, if elimination results in a single chain of possibly faulty components, one

tests first in the middle of the chain. This study showed the importance of elimination for the use of other advanced strategies.

These instructional studies show the effectiveness and importance of the elimination strategy. However, they give little information about the psychological mechanisms underlying strategy learning. To date, more information about strategy learning mechanisms has come from studies of strategy discovery than from instructional studies. Perhaps this is why most theoretical models of strategy learning focus on strategy-discovery mechanisms, rather than the effect of instruction. These models are discussed in the next subsection.

MODELS OF STRATEGY LEARNING

Before discussing models of strategy learning, I give my definition of a reasoning strategy and address the issue of implicit vs explicit knowledge in strategies. Like Schaeken, de Vooght, Veneierendonck, and d'Ydewalle (2000), I see strategies as cognitive control processes that involve both explicit, conscious knowledge and implicit, unconscious knowledge. And like Evans and Over (1999), I see the explicit knowledge used in strategies as requiring working memory resources and as being essential for hypothetical thinking. As will be discussed shortly, explicit hypothetical thinking is essential for learning the troubleshooting strategies considered here. The production-system models of troubleshooting strategies to be presented shortly make specific claims about what strategic knowledge is explicit vs implicit. In these models, knowledge about the states of the problem during problem solving (e.g., "component A may be giving a bad output") is explicit and conscious, whereas rule-based knowledge represented in productions is more implicit. Finally, in line with theories of the acquisition of cognitive skill (Anderson, 1982), I assume that during the early stages of learning a strategy, strategic knowledge is more conscious, and when a strategy is well practised, strategic knowledge is less available to consciousness.

Strategy-discovery models include rule-based, production-system models (Anzai & Simon, 1979; Jones & VanLehn, 1991; Laird & Newell, 1983; Langley, 1985; Ruiz & Newell, 1989; VanLehn, 1991) and connectionist models (McClelland & Jenkins, 1991). The model of Newell and colleagues (Laird & Newell, 1983; Ruiz & Newell, 1989) is especially relevant to the current study. They used the SOAR model to simulate strategy discovery in the Tower of Hanoi problem. Their model eventually used the general-purpose strategy of means–ends analysis to solve the problem, but did not assume that participants have prior knowledge of this strategy. Instead, SOAR induced a version of means–ends analysis applicable to the problem using: (1) general learning and problem-solving mechanisms, including

chunking and subgoaling; and (2) domain-specific knowledge about spatial-manipulation problems. Thus, these researchers suggest that instead of applying a strategy like means–ends analysis to a new domain, people may induce a new (or initial) version of the strategy using general learning and problem-solving mechanisms along with domain-specific knowledge.

A major problem with all of the strategy-discovery models mentioned above is a lack of empirical validation. All seven models were developed at least partially post-hoc to explain pre-existing data. In four of the studies, these data came from two or fewer participants. The current studies use production-system modelling, but rectify some of the problems these models have concerning empirical validation.

In the current study, separate production-system models were developed for the backtracking and elimination strategies. A comparison of these models suggested certain domain-specific and general knowledge needed to use elimination, given knowledge of backtracking. To evaluate this suggestion, the domain-specific and general knowledge suggested by the modelling was taught to college students who knew backtracking. Then the students' performance on a number of troubleshooting problems was observed to see if the new knowledge led to increased use of elimination. This study used an instructional manipulation, but it also included a pure discovery-learning condition with no instruction. Also, the instructional conditions in the study had a discovery-learning flavour in the sense that participants were not taught the elimination strategy explicitly, but instead were taught potential cognitive components of elimination and allowed to discover the strategy. The next subsections describe the task and strategies studied in the experiment and the production-system models of strategy use.

TROUBLESHOOTING TASK

Since the planned experiment required a high degree of experimental control over the participants' domain-specific knowledge, a task was chosen that would be novel for the participants (college students). Participants had to find the broken nodes in simple networks similar to digital circuits (see Figure 1). The network shown passes 0s and 1s from left to right. The nodes act as AND gates. That is, when working correctly, nodes only pass on 1s if all their inputs are 1s; otherwise they pass on 0s. When nodes break, they pass on 0s regardless of their inputs. The participants searched for the broken node by testing particular connections between nodes to see if they were passing 0s or 1s, and by replacing nodes. Tests and replacements were assigned costs (in imaginary money), with replacements costing four times more than tests. Participants were asked to keep costs to a minimum. The network task was taken from the work of Rouse (1978).

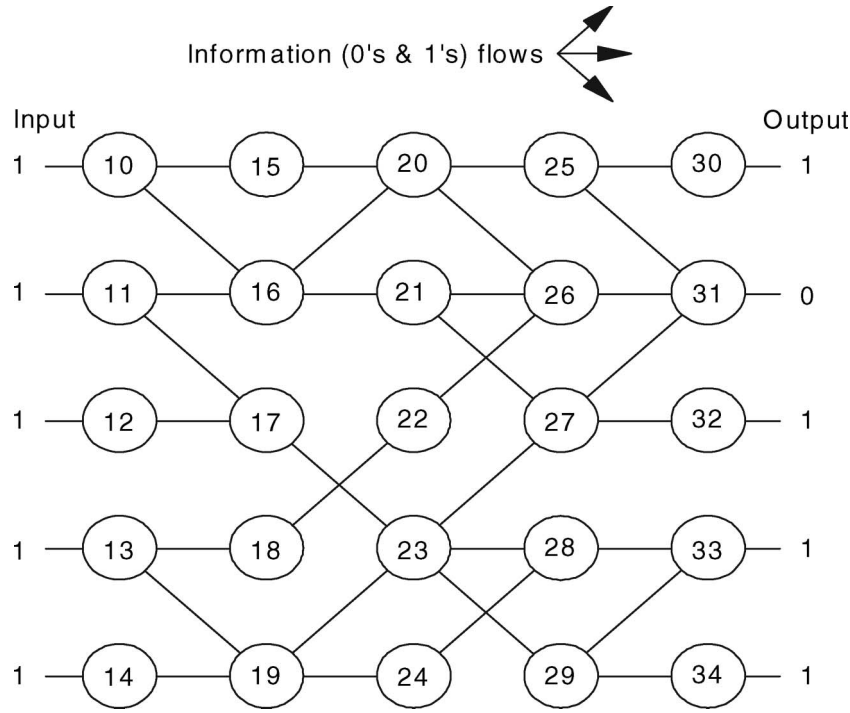


Figure 1. Example network.

Strategies used on the network task

General backtracking. In this strategy, the problem-solver only tests connections that lead into the 0 network output. In the network in Figure 1, this strategy leads initially to considering 18 nodes as possibly faulty. When a test reveals an internal network connection that is passing a 0, a new (smaller) set of possibly faulty nodes is developed that leads into the leftmost connection passing a 0. Then the search continues within this new set. When a node is found with a 0 output and all inputs of 1, it is replaced.

Right-to-left backtracking. This is like the general backtracking strategy, with the additional constraint that connections are tested in a right-to-left order. That is, for whatever set of possibly faulty nodes currently under consideration, the problem solver first tests the rightmost connections leading into the connection passing a 0. This contrasts with the general backtracking strategy, in which the problem solver may jump around in selecting tests within the possibly faulty set of nodes.

Initial elimination. The problem solver follows the general or right-to-left backtracking strategy, but before testing any connection, first checks to see if the node on the left end of the connection leads into any outputs of 1. If it does, then the connection in question is not tested, because it must be passing a 1. For example, upon considering whether to test connection 25–31 in Figure 1, this connection can be eliminated (not tested) because node 25 leads to an output of 1 via node 30. In initial elimination, nodes and connections that lead into 1s are not eliminated until they are considered as candidates for testing based on one of the backtracking strategies.

Advanced elimination. Here, the problem solver systematically eliminates all nodes that lead into connections passing a 1 before considering any tests. For the problem in Figure 1, this strategy initially leaves a possibly faulty set of only 4 nodes (18, 22, 26, 31), much smaller than the set of 18 nodes considered in backtracking. The sub-strategies for choosing tests from within this set include random choice, right-to-left testing, and half split. The most efficient of these, half split, cannot be used with any of the three previous strategies.

If either of the backtracking strategies or the advanced elimination strategy is well practised, then people should be able to follow it without being conscious of the reasoning behind their troubleshooting actions. However, when using initial elimination, people would be expected to be more conscious of their reasoning.

Pilot testing showed that college students usually did not use either of the elimination strategies when troubleshooting the network problems. Most resorted to the less efficient backtracking strategies.

One advantage of the network task is that it allows detailed and objective measurement of strategy use by looking at the moves (tests and replacements) participants make. For example, participants who consistently confine their tests to the small set of possibly faulty nodes remaining after using advanced elimination can be assumed to be using this strategy. Also, since the task is performed on a computer, inter-move latencies can be collected easily. These detailed data on troubleshooting moves and latencies allow a detailed analysis of strategy-discovery processes without using verbal protocols.

PRODUCTION-SYSTEM MODELS OF TROUBLESHOOTING STRATEGIES

In the production-system models for this study, I did not attempt to model the learning process by which someone who knows backtracking transitions to elimination. Instead, separate production-system models were created for

the general backtracking, the right-to-left backtracking, and the initial elimination strategies, because these seemed to encompass the critical stages in the transition from backtracking to elimination. By looking at the knowledge that had to be added to the backtracking models to obtain the initial elimination model, I could infer the knowledge a person using only backtracking would need in order to begin using elimination. To begin describing the new knowledge that enables elimination use, I will informally describe the transition from the right-to-left backtracking to the initial elimination strategy.

When the right-to-left backtracking model finds a node outputting a 0, it generates hypotheses that each of the node's input connections are passing 0s and tests these by making actual tests of the network. For example, given the network in Figure 1, it might generate the hypothesis that connection 25–31 is passing a 0 and immediately test this. In the initial elimination model, this testing process is modified. Instead of immediately testing the hypothesis that 25–31 is 0, the elimination model uses knowledge of how the nodes work and a "what if" reasoning process to propagate the effects of the hypothesis through the network. In this case, propagating the hypothesis that 25–31 is 0 leads to the further hypotheses that 25–30 is passing a 0 and that node 30 is outputting a 0. Since this last hypothesis is contradicted by the fact that node 30 is outputting a 1, the original hypothesis is judged to be false. Thus, the model concludes that 25–31 is passing a 1 and this connection is not tested on the computer. If, after propagation, the original hypothesis is found to only *agree with* known network information, the model goes ahead and makes an actual test of the connection associated with this hypothesis.

The initial elimination model uses two kinds of knowledge in this reasoning process that are not in the right-to-left backtracking model. The first of these is the domain-specific knowledge of how the nodes work that is used to propagate a hypothesis through the network. The model uses two key rules about the nodes, both of which are used in the previous example. These are the rules that all outputs of a node are equal, and that if a node has a 0 input, it will output 0s. The second kind of new knowledge used in the elimination model is the overall process of deciding to propagate a hypothesis, noticing contradictions, and falsifying the original hypothesis. This is the reasoning process known as *reductio ad absurdum* (*RAA*).

The node knowledge used by the elimination model is domain specific. On the other hand, the general-purpose nature of RAA should be emphasised. It is a key part of a number of theories of human deductive and inductive reasoning (Braine, 1990; Rips, 1983). In fact, some of the productions used to model RAA here are closely analogous to schemas used in Braine's (1990) natural-logic theory of reasoning. Polya (1957) included

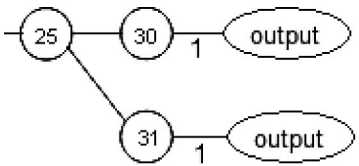
RAA in his book on general problem-solving skills. Ohlsson and Robin (1994) pointed out the use of RAA by mathematicians and scientists such as Euclid and Galilei, and observed students using RAA while solving mechanical motion problems. Handley and Evans (2000) and Braine, Reiser and Rumain (1984) provided evidence that adults sometimes perform poorly on RAA problems, but both of these studies used complex, abstract problems that are not representative of the type of RAA problems that arise during the more concrete task of troubleshooting a physical system.

The domain-specific and general knowledge added to create the elimination model will now be described in more detail. Table 1 informally describes the main productions that had to be added to the backtracking model (either right-to-left or general backtracking) in order for it to solve network problems using the initial elimination strategy. Given the initial information in working memory (the hypothesis that connection 25–31 is passing a 0), first production 4, which is part of RAA, fires and sets a goal of propagating information from this hypothesis. Then the domain-specific productions 1, 2, and 3 actually propagate the information by adding to working memory the further hypotheses that 25–30 is 0, that connection 30–output is passing a 0, and that 30–output is not passing a 1. At this point, RAA-related production 5 notes that the last hypothesis added to working memory contradicts the perceived information that 30–output is passing a 1. Finally, given this contradiction, RAA production 6 falsifies the original hypothesis.

These production system models show how a general-purpose troubleshooting strategy—elimination—can be induced from even more general knowledge—RAA—and domain-specific knowledge. Thus, they are similar to Newell's SOAR model (Laird & Newell, 1983; Ruiz & Newell, 1989) in which the general strategy of means–ends analysis was induced from more general knowledge and domain knowledge.

One more point about the cognitive mechanism underlying RAA is in order. Handley and Evans (2000) note that RAA is an important example of suppositional reasoning, and present evidence that some types of suppositional reasoning are based on mental models of the problem domain and not on the kinds of mental rules hypothesised by Braine (1990) and Rips (1983). The mental model view might also be seen as conflicting with the rule-based production system used to model RAA and other troubleshooting processes in this study. However, when the production system proposed here is used to solve an actual troubleshooting problem, the modelled problem solver builds up knowledge in working memory about the actual and hypothesised states of a specific physical system. This specific knowledge in working memory can be seen as a mental model of the physical system. Thus the production system approach seems to blend the rule-based and mental model approaches to modelling reasoning processes.

TABLE 1
Key productions added to backtracking models

Partial network problem	*Visually available info (VIS) about the problem*	*Initial state of working memory (WM)*
	in VIS: 25–30 passing an-unknown-value in VIS: 25–31 passing an-unknown-value in VIS: 30–output passing 1 in VIS: 31–output passing 1	in WM: hypothesis that 25–31 passing 0

Productions representing domain-specific knowledge (partial)	*Productions representing reductio ad absurdum (partial)*
#1. All outputs of a node are equal IF in WM: goal is to propagate info from a hypothesis & in WM: hypothesis that *leftnode–rightnode* passing *X* & in VIS: *leftnode–othernode* passing *anyvalue* & *othernode* is not equal to *rightnode* THEN add to WM: hypothesis that *leftnode–othernode* passing *X*	#4. Start propagating info about a hypothesis IF in WM: hypothesis that *leftnode–rightnode* passing *X* THEN add to WM: goal is to propagate info from a hypothesis & add to WM: propagating hypothesis *leftnode–rightnode* passing *X*
#2. Nodes pass 0s forward IF in WM: hypothesis that *leftnode–node1* passing 0 & in VIS: *node1–othernode* passing *anyvalue* THEN add to WM: hypothesis that *node1–othernode* passing 0	#5. Notice propagated hypothesis contradicts perceived info IF in WM: hypothesis that *leftnode–rightnode* not passing *X* & in VIS: *leftnode–rightnode* passing *X* THEN add to WM: found a contradiction
#3. If a node is passing a 0, it is not passing a 1 IF in WM: hypothesis that *leftnode–rightnode* passing 0 THEN add to WM: hypothesis that *leftnode–rightnode* not passing 1	#6. Given a contradiction, falsify original hypothesis IF in WM: propagating hypothesis *leftnode–rightnode* passing 0 & in WM: found a contradiction THEN remove from WM: propagating hypothesis *leftnode–rightnode* passing 0 & remove from WM: hypothesis that *leftnode–rightnode* passing 0 & add to WM: fact that *leftnode–rightnode* passing 1

English description of the key productions added to the backtracking models in order for them to solve network problems using the initial elimination strategy. Also shown is a partial network problem and visually available and working memory information about that problem. Italicised words in productions represent variables that can be bound to specific values in working memory or visually available information. Given the information in working memory, these productions will fire in the order: #4, 1, 2, 3, 5, 6.

To summarise, the production-system modelling led to the hypothesis that people using only backtracking will learn elimination if they apply: (1) particular domain-specific knowledge about how the nodes work, and (2) the general RAA reasoning strategy. According to the models, *both* of these kinds of knowledge are needed for learning elimination. In addition, the production system models do not suggest that overall improvement in domain-specific knowledge will lead to the elimination strategy. Two, quite specific, rules about how the nodes work are used in the model that can use elimination. Other domain-specific knowledge would not be expected to lead to elimination. The next section describes the experiment that tested these hypotheses.

EXPERIMENTAL DESIGN AND PREDICTIONS

To test the models, an experiment was conducted with five conditions, based on five ways of training participants to do the network task. In each condition, participants first received brief instruction in how the nodes and networks worked and how to perform the troubleshooting task. The initial instruction contained enough information for the participants to induce the elimination strategy, but this information was not highlighted. After the initial training, each participant completed a pretest of four network problems, received one of the five kinds of extra training, and then completed a post-test of 24 network problems.

The purpose of this experiment dictated the type of training that was given. The purpose was not simply to see how well participants could learn the initial or advanced form of elimination described earlier, and how this would affect their performance. Rather, the purpose was to test whether the types of domain-specific and general knowledge identified by the models were really sufficient to allow participants to learn elimination. In a sense, the experiment tested whether the production system models constituted an accurate task analysis of the task of using the elimination strategy. Because of these goals, the training given to participants had a discovery-learning, rather than a direct-instruction, flavour. Participants were not explicitly taught the initial or advanced form of elimination. Rather, they were taught the kinds of knowledge highlighted by the models as important for learning this strategy. Then their performance was assessed to see whether this knowledge allowed them to induce the elimination strategy.

In particular, the training conditions were designed to test whether either of the two types of knowledge highlighted by the model (domain-specific and general) could facilitate elimination by itself, or whether both had to be taught together. Thus, conditions were included in which participants received (a) no extra training (*baseline*), (b) only the domain-specific node knowledge suggested by the model (*relevant domain-specific*), or (c) both the

relevant domain-specific knowledge and the general RAA strategy (*relevant domain-specific/RAA*) (see Table 2). In the relevant domain-specific/RAA condition, participants learned RAA in the context of the network problems.

In the three conditions described so far, participants received either no extra training, relevant domain-specific training only, or relevant domain-specific training plus RAA training. To fully test whether the domain-specific and RAA training were helpful separately as well as together, an RAA-only condition was needed in which participants learned RAA but not the relevant domain-specific knowledge. Since people are often bad at transferring knowledge across domains, it was important that participants in the RAA-only condition learned RAA in the context of the network problems, as did the participants who learned both RAA and relevant domain-specific knowledge. Thus, for the RAA-only condition, RAA was taught in the context of the network problems, but using domain-specific knowledge that was, according to the model, irrelevant to learning elimination. This was the *irrelevant domain-specific/RAA* condition. The domain-specific knowledge taught—both relevant and irrelevant to elimination—is shown in Table 3. It is important to stress that "relevant" and "irrelevant", as used here, refer only to relevance to learning elimination, as predicted by the model, and not to relevance to the network task. In both the relevant

TABLE 2
Training conditions used

Type of domain-specific training	*Reductio-ad-absurdum training given?*	
	No	*Yes*
None	Baseline	
Irrelevant	Irrelevant domain-specific	Irrelevant domain-specific/RAA
Relevant	Relevant domain-specific	Relevant domain-specific/RAA

TABLE 3
Domain-specific knowledge taught

Relevant domain-specific knowledge:
(1) 0s are always passed forward through a node
(2) all outputs of a node are equal

Irrelevant domain-specific knowledge:
(1) 1s are passed forward through a node, if the node is working and all the inputs are 1s
(2) if a working node has a 0 output, it must have at least one 0 input

and irrelevant domain-specific training, the rules participants learned accurately described how the networks worked.

Finally, an *irrelevant domain-specific* condition was included, in which participants learned only the irrelevant domain-specific knowledge. This condition allowed a test of whether the "relevant" domain-specific knowledge indicated by the model was really essential. By comparing the baseline, the irrelevant domain-specific, and relevant domain-specific conditions, I could determine whether any increase in elimination use after training in domain-specific knowledge was due to the particular domain-specific knowledge highlighted by the model as relevant to elimination or to general familiarity with how the nodes worked.

The main prediction following from the production-system models was that participants in the relevant domain-specific/RAA condition were expected to show the greatest use of elimination, since these participants were explicitly taught all the knowledge (both domain-specific and general) identified by the modelling as used in elimination. Participants in the other four conditions were predicted to show little improvement in their use of elimination, since they were not taught all of the knowledge used in elimination. This prediction was tested first by looking at the percentage of participants' troubleshooting actions that were consistent with the elimination strategy. Since the elimination model solved network problems using fewer network tests than the backtracking models, the prediction was also tested using the number of tests. Finally, since the elimination model used considerably more productions than the backtracking model to make each problem-solving move, it was expected that participants receiving relevant domain-specific/RAA training would take longer to make each problem-solving move than those in the other training conditions.

METHOD

Design

A between-subjects design was used. The independent variable was the type of extra training participants received, beyond the basic training necessary to do the task.

Participants

The 62 participants were all University of Michigan undergraduate students who were paid or who participated for course credit. The average class level of the participants was second year. People who were experienced at computer programming were excluded immediately from being participants, since this task involves extensive troubleshooting of abstract systems and

thus is similar to the network task. The criterion for exclusion was having more than 1 month of programming experience in college or more than two semesters in high school.

Data for 12 of the 62 participants were not used in the final analyses. Prior to the experiment, it was decided that participants whose performance indicated that they already knew the elimination strategy would be excluded, as they would not be able to benefit from any of the training. Thus, participants who made 100% elimination tests on at least three of the four pretest problems were dropped. Five participants were excluded by this criterion. Four participants were dropped for lack of effort in performing the network task. Two of these participants answered a post-experimental question about their effort on the post-test by choosing 1 (''not hard at all''), one gave up on two of the first four post-test problems, and another did not follow the instructions to minimise the amount of money spent on each problem. Two participants were excluded because they had completed only half the post-tests in the 2.5-hour time limit, which was more than enough for the other participants to complete all the problems. One participant was dropped because, in response to a question during the initial training, the experimenter inadvertently gave him a hint that suggested the use of the elimination strategy.

The remaining 50 participants were randomly assigned to the five conditions, such that each condition contained six females and four males.

Materials and task

The networks were presented to the participants on paper, in the format shown in Figure 1. The 4 pretest and 24 post-test networks were all either five nodes × five nodes or six × six. The faulty node was located approximately equally often in each of the rows and columns. For each network problem, participants made tests and replacements using the computer. For example, to test the line connecting nodes 22 and 26 in Figure 1, a participant would type in the two node numbers and press a key labelled ''Test''. The computer would display the test result (0 or 1) immediately to the right of the node numbers. To replace node 22, a participant would type in the number and press a key labelled ''Repl.''. The computer would then display the words ''Replacement Correct'' or ''Replacement Incorrect'' to the right of the node number. The results of all tests or replacements for the current problem (up to a limit of 37) were always displayed on the monitor. In the very few cases where 37 tests or replacements were made, the participant was asked to go on to the next problem. A message reminding participants of the costs for tests ($10) and replacements ($40) was displayed on the monitor whenever participants were working on a network problem. After each test or replacement, a running

total of how much had been spent so far on that network was updated. Thus, after each network problem, participants received feedback about the correctness of their solution and the amount of money they had spent.

After each of the post-test networks (but not the pretest networks), participants were given additional feedback indicating the minimum number of tests required for that network. The minimum number of tests for a network was determined by first applying the elimination strategy and then the half-split strategy. The minimum number of tests ranged from 0 to 5. The feedback regarding minimum number of tests involved telling participants how much money they should have spent on each problem, including the $40 cost of the correct replacement. For example, if the minimum number of tests was 2 and the participant had spent $80, the monitor would display: "You spent $80 for this network. A good score for this network would be to spend $60 or less." The purpose of this feedback was to motivate participants to use efficient strategies. This feedback was not presented on pretest problems in order to reduce cognitive load on participants during their initial exposure to the network task.

Procedure

Participants were run in individual sessions lasting from 1.0 to 2.5 hours. The overall sequence of events during a session was: initial training in the network task, pretest (4 network problems), domain-specific training (depending on the condition), RAA training (depending on the condition), post-test (24 network problems), and debriefing. The experimenter was unaware of what condition a participant was in until after the pretest.

Initial training. Participants were given instruction in how the networks worked and how to do the task, and were told that each test cost $10 and each replacement cost $40, in imaginary money.

Pretest. Before the pretest, the participants read instructions which emphasised that there would be only one broken node per network, that they should try to solve the network problems using the minimum amount of money, and that time was not a factor for the problems. The participants then solved four network problems.

Domain-specific training. The goal of the domain-specific training was to teach the participants particular rules about how the nodes worked. Participants saw diagrams of individual nodes like those in Figure 2a and 2b, with some of the node inputs or outputs shown. The participants' task was to indicate, if possible, what information was being passed along the line with the question mark. The answer was either 0, 1, or "can't tell from

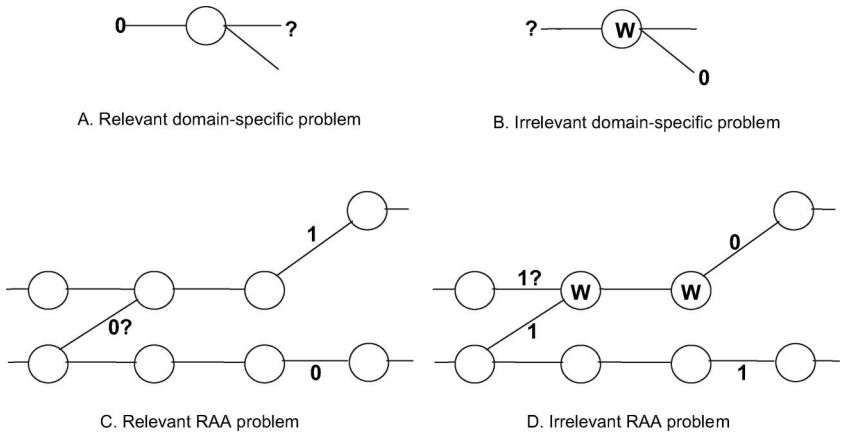


Figure 2. Examples of domain-specific and RAA training problems.

the information given''. (To reiterate, all of the domain-specific rules accurately described the networks; the terms relevant and irrelevant domain-specific training refer to relevance to the elimination strategy based on the models.)

Participants learned two rules in the relevant domain-specific training conditions and two rules in the irrelevant domain-specific training conditions, as shown in Table 3. Both of the irrelevant rules are only true if the node is working. Thus, in all the irrelevant node stimuli, the node had a ''W'' in it to show that it was working. In addition to problems demonstrating these rules, participants saw distractor problems that required an answer of ''can't tell''.

In the initial phase of domain-specific training, participants saw examples on the computer monitor for each of the rules they were to learn. Each example contained a node diagram, the correct answer, and the reason for the answer (e.g., for Figure 2a, the answer is 0 and the reason is ''0s are always passed forward through a node''). In the test phase, participants saw only the node diagrams. They indicated their answers by pressing keys labelled ''0'', ''1'', or ''Can't Tell''. If participants answered incorrectly, the correct answer and reason were displayed. All participants in a domain-specific-training condition did 96 problems, presented in one of two random orders, with half the participants seeing each order.

RAA training. The RAA training always followed the domain-specific training, because it used knowledge taught in the domain-specific training. The goal of this part of the training was to teach participants the RAA reasoning strategy in the context of the network task. Participants saw

diagrams of partial networks like those in Figure 2c and 2d. The diagrams contained information about the values passed on certain lines and, for some problems, whether certain nodes were working. They also contained a hypothesis about the value passed on one line, indicated by a question mark. The participants' task was to indicate whether the line with the hypothesis was actually passing a 0 or a 1, or whether this was impossible to determine from the information given.

In half of the RAA problems, the hypothesis was contradicted by the known information shown in the diagram. In these problems, the correct answer was that the hypothesis line was passing the opposite of the hypothesised value. In the other half of the problems, the hypothesis agreed with the known information. For these problems the correct answer was "can't tell". In the example in Figure 2d, the hypothesis that the upper left-hand line is passing a 1 leads to the hypothesis that the upper right-hand line in also passing a 1, which is contradicted by the known information. Thus, the correct answer is that the upper left-hand line is passing a 0.

The RAA problems were presented on paper. Participants in a RAA training condition did 24 problems in one of two random orders, with half the participants seeing each order. For the participants, the RAA strategy was given the name "Looking for Contradictions". In the initial training phase, participants first saw a written statement of the strategy. The experimenter then demonstrated the strategy on three or four examples. If the hypothesis led to a contradiction, participants were told to cross out the initial hypothesis and write down what value was actually being passed on that line (0 or 1). If the hypothesis was not contradicted by any of the known information, participants were to write down "CT" (for can't tell) next to the initial hypothesis. The Appendix contains a full statement of the RAA strategies taught in the relevant and irrelevant conditions, which differed slightly.

In the test phase of the RAA training, the experimenter told the participants immediately after each problem whether they had got the problem wrong, but did not explain their errors. After the RAA test problems, the experimenter verbally reminded the participants that using the RAA strategy instead of immediately testing network lines on the computer would help them achieve their goal of solving the network problems using the minimum amount of money. Also, participants were given a card with the written statement of the RAA strategy (as presented in the Appendix) to use as a reference during the rest of the experiment.

Post-test. The 24 post-test networks were presented to participants in two orders to reduce any effects associated with a particular problem sequence. Half the participants saw each order.

Debriefing test. After the post-test, participants rated, on a scale of 1 to 5, how hard they tried to solve the network problems using the minimum amount of money.

RESULTS AND DISCUSSION

Before presenting the results, precise definitions will be given for the dependent variables used to measure participants' performance on the network task. The *percentage of backtracking moves* was defined as the percentage of tests and replacements consistent with the general backtracking strategy modelled by the production system. A move was considered a general backtracking move if it was a test of a connection between two nodes in the current backtracking set, or a replacement of a node in the current backtracking set, where the current backtracking set is the set of nodes leading into the leftmost connection known to be passing a 0. That is, the current backtracking set is updated (made smaller) during a problem whenever possible, based on information from tests, as in the production system model.

The *percentage of elimination moves* was the percentage of tests and replacements consistent with the modelled elimination strategy. Elimination moves were those within the current elimination set, where the current elimination set is the set of nodes and connections that lead into the leftmost 0 output, but not into a connection or network output known to be passing a 1. At the beginning of the problem in Figure 1, the current elimination set consists of nodes 18, 22, 26, and 31, and connections 18–22, 22–26, and 26–31.

It is important to note that participants using less effective strategies than elimination, such as backtracking, will make some moves in the current elimination set by chance. Therefore, to get a more accurate estimate of the amount of elimination use, the percentage of elimination moves used by a group of participants was compared to the percentage of elimination moves expected, given use of a less effective strategy. Since backtracking was a very common strategy prior to the training manipulation, participants' percentage of elimination moves was compared to the percentage of elimination moves expected given use of backtracking.

Effectiveness of strategy training

Pretest data. Except for 2 (of 200) problems, participants were able to solve all the pretest problems. An important question concerning the pretest data is whether there were any pre-existing differences between conditions on any dependent variables related to strategy use. In Table 4, participants' pretest performance, broken down by condition, is characterised in terms of percentage of elimination moves, percentage of general backtracking moves,

TABLE 4
Pretest strategy use data

Strategy measurement variable	*Average pretest performance on strategy measurement variables by training condition*					*Predicted pretest performance on strategy measurement variables for each strategy*[1]			
	Base-line	*Irrelevant domain-specific*	*Relevant domain-specific*	*Irrelevant DS/RAA*	*Relevant DS/RAA*	*Random testing*	*General back-tracking*	*Right to Left back-tracking*	*Initial elimination*
% General backtracking moves	91.0	88.6	95.1	91.3	97.2	52	100	100	100
% Elimination moves	48.2	46.8	53.9	53.3	58.9	13	22	52	100
Number of tests	6.9	8.9	5.5	6.3	5.1	26.6	15.1	6.0	2.5
Time per move(s)	26.3	20.8	17.0	27.6	25.5	na	na	na	na

Strategy use data for each training condition averaged over the four pretest problems, along with strategy use data predicted from various modelled strategies.
Average SE for % general backtracking moves was 3.0, for % elimination moves was 5.0, for number of tests was 1.1, and for time per move was 4.7 s.
DS = domain-specific.
[1]Expected values of the strategy performance variables were calculated by running each of the production system models, and a random testing model, 1000 times on each of the four pretest problems.

number of tests, and time per move. The percentages of elimination and backtracking moves do not add to 100% because the set of permissible elimination moves is always a subset of the set of backtracking moves; thus participants using backtracking will by chance make some elimination moves. For example, the modelled right-to-left backtracking strategy makes about 52% elimination moves by chance on the pretest problems. This explains why the participants in each training condition showed a high percentage of backtracking moves (above 88%) as well as 47–59% elimination moves.

The data for these four variables were averaged over the pretest networks. As the table shows, the groups did not differ much across conditions on any of these variables. ANOVAs confirmed that the condition effect did not approach significance for any of these dependent variables ($p > .3$ for each variable.) Table 4 also compares participants' actual performance on the pretest to the predicted performance given consistent use of the three modelled strategies—general backtracking, right-to-left backtracking, and initial elimination—and a random testing strategy. Paticipants' percentage of elimination and of backtracking moves and number of tests fit most closely with the predicted values for the right-to-left backtracking strategy, and do not fit the predicted values of the other three strategies. Thus, prior to training, most of the college students in this study probably used a right-to-left backtracking strategy.

In addition to the pretest data, the participants' scores on standardised tests of mathematical ability (SAT and ACT) were used as a measure of whether the experimental groups differed prior to the experiment in terms of skills related to the experimental task. ACT scores were converted to SAT equivalents. The mean mathematics SAT score for the baseline, irrelevant domain-specific, relevant domain-specific, irrelevant domain-specific/RAA, and relevant domain-specific/RAA conditions were 551 (57), 555 (58), 558 (39), 570 (47), and 593 (119), respectively (with the half width of a 95% confidence interval in parentheses). None of the means is significantly different from the others.

To deal with any pre-existing individual differences in ability to perform the network task, participants' pretest scores on the dependent variable of interest were used as a covariate when analysing the post-test data.

Domain-specific and RAA training data. Participants did quite well on the domain-specific and RAA training problems. The mean percent correct was 99% for the participants receiving relevant domain-specific training and 95% for those with irrelevant domain-specific training. These means were not significantly different, $t(18) = 1.47$, $p > .10$. The participants receiving relevant RAA and irrelevant RAA training answered correctly an average of 91% and 90%, respectively.

Post-test data. The participants solved all of the post-test problems except in the case of one participant who used more than the maximum number of 37 tests on two of the problems. The dependent variables describing post-test performance were averaged over the 24 post-test problems. For each dependent variable, an analysis of covariance (ANCOVA), was conducted on the post-test variable, with training condition as the independent variable and average pretest score on that variable as a covariate. All of the post-test data (including graphs) are presented in terms of the adjusted means from the ANCOVA. These are the means that would be expected if there were no pre-existing (i.e., pretest) differences among the participants on the dependent variable. The hypothesis that only the relevant/domain-specific training would lead to noticeable improvements in use of the elimination strategy was tested by a statistical contrast comparing that training condition with the average of the other four conditions.

The percentage of elimination moves for each training condition is shown in Figure 3. Training condition significantly affected the percentage of elimination moves, $F(4,\ 44) = 2.99,\ MSE = 434.9,\ p < .05$. As predicted,

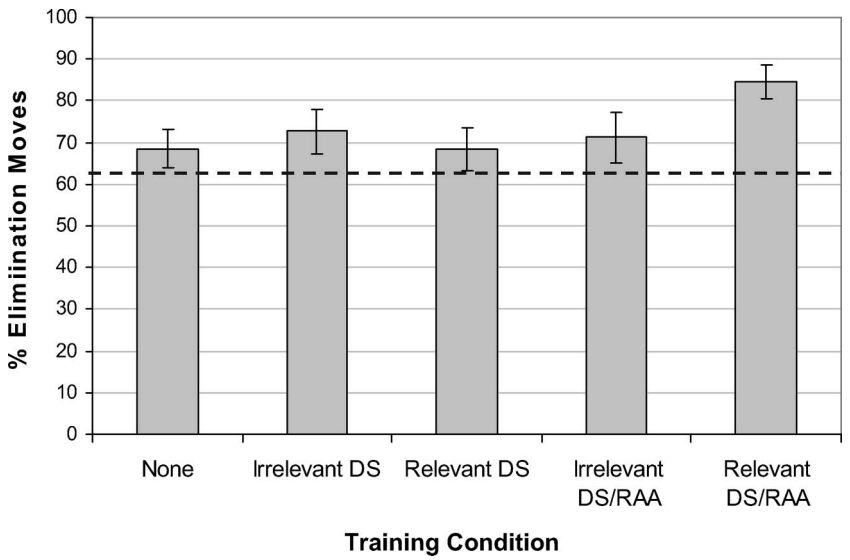


Figure 3. Percentage of elimination moves (adjusted means from ANCOVA) for each training condition averaged across the post-test problems, with standard errors. Dotted black line shows the percentage of elimination moves expected given use of the modelled right-to-left backtracking strategy. 100% elimination moves are expected given the initial elimination strategy (DS = domain-specific; RAA = Reduction ad absurdum).

participants in the relevant domain-specific/RAA condition had significantly higher elimination use than those in the other conditions, which were approximately equal, contrast $F(1, 44) = 11.0$, $MSE = 1596$, $p < .01$. Thus, the training most closely patterned after the elimination model (i.e., the *model-based training* in both relevant domain-specific knowledge and RAA) led to greater elimination use than other kinds of training. The smallest effect size when comparing the model-based training to each of the other four training conditions was a Cohen's d of 0.80. This is a large effect size in Cohen's (1988) terms. For comparison, the figure also shows the percentage of elimination moves expected given consistent use of the right-to-left backtracking strategy. Although the model-based training group did not show the 100% use of elimination predicted by the model, it came closer to this level than the other training conditions; and the other conditions were not much above the percentage of elimination moves expected given use of the right-to-left backtracking strategy.

It was also predicted that participants who used elimination would make fewer troubleshooting tests. This prediction was supported by a high negative correlation of $-.90$ ($p < .001$) between percentage of elimination moves and number of tests. Figure 4 shows the number of tests for the

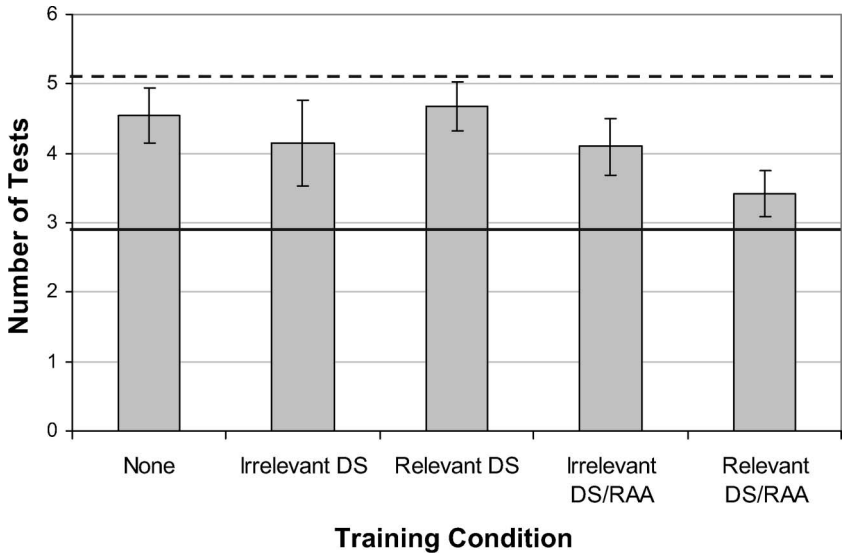


Figure 4. Number of tests (adjusted means from ANCOVA) for each training condition averaged across the post-test problems, with standard errors. Dotted and solid black lines show the percentage of elimination moves expected given use of the modelled right-to-left backtracking and initial elimination strategies, respectively (DS = domain-specific; RAA = Reduction ad absurdum).

different training conditions. The main effect of training condition was significant for number of tests, $F(4, 44) = 2.64$, $MSE = 2.43$, $p < .05$. Participants in the relevant domain-specific/RAA condition had significantly fewer tests ($M = 3.4$) than the average of the other conditions ($M = 4.4$), contrast $F(1, 44) = 7.66$, $MSE = 7.04$, $p < .01$, and approached closer to the number of tests given consistent use of the modelled elimination strategy (2.9) than the other conditions. Effect sizes for comparing model-based to other training conditions ranged from medium to large (Cohen's d from 0.47 to 1.15).

Participants who used elimination were also predicted to take more time for each problem-solving move. This prediction was supported by the fact that elimination use correlated .76 ($p < .001$) with time per move. The time per move data was positively skewed, so before conducting inferential statistics a Box-Cox transformation ($\lambda = -0.71$) was applied, which reduced the skewness of the pretest and post-test time per move data to 0.18 and -0.17, respectively. Figure 5 shows the time per move for the different training conditions (using untransformed times for clarity). The effect of training condition on time per move approached significance, $F(4, 44) = 2.36$, $MSE = 0.004$, $p = .07$; and a contrast comparing the times for the relevant domain-specific/RAA condition to the average of the other four conditions was significant, $F(1, 44) = 8.04$ $MSE = 0.004$, $p < .01$. Effect sizes

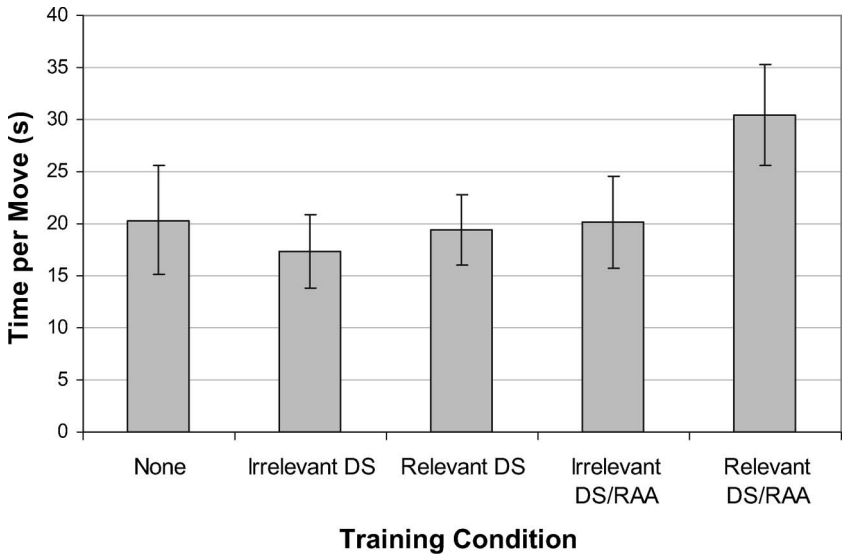


Figure 5. Untransformed time per move (adjusted means from ANCOVA) for each training condition averaged across the posttest problems, with standard errors (DS = domain-specific; RAA = Reduction ad absurdum).

for comparing model-based to other training conditions ranged from medium to large (Cohen's *d* from 0.63 to 1.07).

In agreement with the production system modelling, these data suggest that model-based training did lead to increased elimination use by college students, and that this use was reflected in the type, number, and timing of students troubleshooting actions. In order to induce the elimination strategy on the network task, college students needed training that conveyed *both* domain-specific knowledge of how the nodes work and knowledge of the general RAA strategy. Training in only one of these kinds of knowledge did not lead to induction of elimination.

The models also predicted that only particular kinds of domain knowledge would facilitate elimination use. A comparison of the irrelevant domain-specific/RAA and relevant domain-specific/RAA conditions shows that pairing RAA training with any kind of training in domain-specific knowledge is not enough to allow learning of elimination. Only the specific domain knowledge highlighted by the models increased elimination use, when paired with the appropriate general strategy.

An objection can be raised to the conclusion that participants who received model-based training (i.e., the relevant domain-specific/RAA condition) were using elimination more than other participants. The fact that participants receiving model-based training made fewer tests but took longer to make moves could mean that they were simply being more reflective, and not using a different strategy from other participants. Two findings argue against this conclusion. First, by the last six problems of the post-test, participants in the model-based condition were working as fast as the baseline (no training) participants. Yet, for these problems, the percentage of elimination use for the model-based training group was at its highest point. Second, participants who showed a high percentage of elimination use often crossed off network nodes that could be inferred via elimination to be working, whereas participants with a low percentage of elimination use did this much less often. Many participants using elimination systematically crossed off all nodes that led to an output of 1 before making their first move. This suggests they were using the advanced form of elimination mentioned previously. These findings suggest that participants with a high percentage of elimination moves (who were predominantly in the model-based training condition) really were using a different strategy from participants using backtracking. Also, the behaviour of crossing off eliminated nodes provides evidence that, at least at this stage of its development, elimination was an explicit, conscious strategy.

The fact that participants in the model-based training group took longer than other groups to solve the post-test problems could be seen as evidence for the cognitive costs of the elimination strategy. However, the fact that the model-based training group solved problems as quickly as other groups by

the end of the post-test shows that the cognitive costs of elimination are considerably reduced during the short practice period of solving the 24 post-test problems. It would be expected that any cognitive costs due to the extra inferences required for the initial elimination strategy would be further reduced as elimination becomes well practised and less demanding of conscious, working memory resources. One reason for this is that with practice the initial elimination strategy would likely shift into the advanced elimination strategy, which probably has lower cognitive costs because nodes leading to good outputs are eliminated without going through the extensive inferences of RAA.

GENERAL DISCUSSION

The questions motivating this research concerned how people learn new troubleshooting strategies. I focused on elimination because this strategy is essential to effective troubleshooting, and because data from maintenance technicians (Kurland & Tenney, 1988) and my pilot data showed that novice troubleshooters often fail to use this strategy. Prior research suggested that learning new problem-solving strategies requires students to integrate domain-specific and general-purpose knowledge. In this study, production system modelling was used to focus on the particular kinds of domain-specific and general knowledge needed for learning elimination, given prior knowledge of the common precursor strategy, backtracking. The modelling highlighted the following kinds of knowledge as necessary: (1) domain-specific knowledge about how the malfunctioning system works (e.g., all outputs of an AND gate are equal), and (2) the general RAA strategy. The experiment used a training manipulation to test whether both of the kinds of knowledge highlighted by the model were necessary for learning elimination.

Implications of the strategy training findings

The experiment showed that only college students who received both the domain-specific and the general knowledge (RAA) identified by the modelling increased their use of elimination relative to an untrained control group. The model-based training group had a higher percentage of moves consistent with elimination, made fewer tests, and took longer to make moves than the other training groups. These differences reflected medium to large effect sizes.

These findings support the production-system models of elimination learning. Both the general RAA strategy and certain domain-specific knowledge are needed to induce elimination. These findings also support the models of strategy learning and discovery developed with SOAR, in which very general problem-solving mechanisms are combined with

domain-specific knowledge to induce a problem-solving strategy applicable to a new domain (Laird & Newell, 1983; Ruiz & Newell, 1989). Finally, these findings fit with claims that developing expertise involves the interaction of general and domain-specific knowledge (Kuhn, 1991; Siegler, 1989; Sternberg, 1989; Zimmerman, 2000). An advantage of the production-system modelling is that it enabled the global claim of "interaction of general and specific knowledge" to be particularised. The modelling identified the particular kinds of general and specific knowledge needed for learning elimination.

Two findings from these experiments—the high performance of students on the RAA training problems, and the effectiveness of RAA training in increasing elimination for these students—stand in contrast to the difficulties experienced by Handley and Evans' (2000) and Braine et al.'s (1984) adult participants in using RAA. However, the current results do not contradict these previous findings, because the RAA problems in the current studies were simpler and more concrete than in the previous studies. In terms of the cognitive mechanism underlying RAA, the production system models used in this project integrate rule-based and mental-model approaches to modelling reasoning processes. Also, the high accuracy (90% correct) of students on the RAA training problems suggests that the students were not learning RAA for the first time in this study. It seems more reasonable that the RAA training was priming prior knowledge about RAA that students had not applied during the pretest problems (inert knowledge).

Similarly, the high performance of students on the domain-specific training tasks (about 97% correct) suggests that this training may have been priming prior knowledge that was available in the initial network-task instructions. Thus, my conclusion from this experiment is not that the model-based training taught the cognitive components of the elimination strategy, but rather that this training focused students' attention on the specific kinds of knowledge that could be used to learn elimination.

In the introduction, I emphasised the general-purpose nature of RAA, and suggested that the elimination strategy is also applicable to a variety of troubleshooting tasks and domains. Other research supports this suggestion. Klahr and Carver (1988) trained children in computer-program trouble-shooting procedures, and found that this training led to improved troubleshooting in other tasks such as following directions with a map. Lee (1993) found that participants who were experienced in computer programming but not electronics used the same advanced troubleshooting strategies, including elimination, with electronic circuits as they did with computer programs, as long as the participants were given domain-specific information about how the circuits worked.

The current study and the studies just reviewed suggest that people can learn reasoning schemas that are abstract and general purpose; that is, not

tied exclusively to specific task domains. In general, Nisbett (1993) and Case (1993) have surveyed a wide range of studies supporting the conclusion that people have reasoning schemas that are intermediate in generality between domain-specific knowledge and very general cognitive processes. Examples of these intermediate-level reasoning schemas include: the law of large numbers, modus ponens, and causal reasoning schemas (Smith, Langston, & Nisbett, 1992). Given the research findings cited above concerning cross-domain transfer, and the effectiveness of the model-based training with college students in this experiment, it seems reasonable to hypothesise that general RAA training would improve elimination use in complex, real-world troubleshooting tasks, when combined with appropriate training in domain-specific knowledge.

What kind of domain-specific knowledge would be needed for learning elimination in a real-world troubleshooting domain? The domain-specific knowledge that was found to be effective in these studies (the relevant domain-specific training) included the information about system structure (topography) contained in the network diagrams as well as the following rules: Rule 1 – a component receiving any bad input will transmit bad outputs, and Rule 2 – the outputs of a component are either all good or all bad. In a real-world system, components do not always malfunction in the all-or-none fashion described in these rules. Thus, these rules would need to be modified to fit the domain. Real-world troubleshooters would need to know, for Rule 1, what inputs of individual components will cause them to transmit bad outputs, and for Rule 2, which particular outputs of components will be bad when they receive some bad input. This knowledge would allow a troubleshooter to infer how possibly faulty components will affect other, "downstream" system components, as is needed to use the RAA strategy.

The effectiveness of the model-based training in these experiments demonstrates that cognitive modelling can serve as a method of cognitive task analysis that can lead to improved training systems (Means & Gott, 1988). That is, modelling enables identification of the content knowledge and cognitive processes that students should be taught (or reminded of) for a particular task. In this case, modelling allowed me to specify the particular kinds of domain-specific and general knowledge needed to learn the elimination strategy. This kind of detailed cognitive task analysis should be useful in developing training for complex troubleshooting and maintenance tasks.

The above discussion has focused on the *contents* of troubleshooting knowledge; that is, the particular types of knowledge needed to learn the elimination strategy. What implications does this study have for *methods* of teaching troubleshooting strategies? The training method in this study was intermediate between explicit instruction and discovery learning. That is, students were explicitly taught (or reminded of) strategy component

knowledge and allowed to discover the target strategy if they could from the component knowledge. This method was effective at improving use of the target elimination strategy. However, since the current study used a single method of strategy training, I cannot make conclusions about the effectiveness of this method relative to others. Klahr and Nigam (2004) conducted an interesting study regarding methods of strategy instruction, which suggests that both direct instruction and discovery learning of reasoning strategies can transfer to "authentic" domains, and that the level of mastery of a strategy during initial training is a much better predictor of transfer than the method of training.

In conclusion, the production system models helped to identify some of the key knowledge needed to learn an important troubleshooting strategy—elimination. The experiment showed that people need both the domain-specific and the general RAA knowledge identified by the models in order to learn elimination.

Manuscript received 6 October 2005
Revised manuscript received 3 April 2006
First published online 20 July 2006

REFERENCES

Alexander, P. A., & Judy, J. E. (1988). The interaction of domain-specific and strategic knowledge in academic performance. *Review of Educational Research*, *58*(4), 375–404.

Anderson, J. (1982). Acquisition of cognitive skill. *Psychological Review*, *89*(4), 369–406.

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

Braine, M. (1990). The "natural logic" approach to reasoning. In W. Overton (Ed.), *Reasoning, necessity, and logic: Developmental perspectives*. Hillsdale, NJ: Erlbaum.

Braine, M., Reiser, B., & Rumain, B. (1984). Some empirical justification for a theory of natural propositional logic. *The Psychology of Learning and Motivation*, *18*, 313–371.

Brooke, J., Cook, J., & Duncan, K. (1983). Effects of computer aiding and pre-training on fault location. *Ergonomics*, *26*(7), 669–686.

Carlson, R., Lundy, D., & Schneider, W. (1992). Strategy guidance and memory aiding in learning a problem-solving skill. *Human Factors*, *34*(2), 129–145.

Case, R. (1993). *The mind's staircase*. Hillsdale, NJ: Erlbaum.

Chase, W., & Simon, H. (1973). The mind's eye in chess. In W. Chase (Ed.), *Visual information processing*. New York: Academic Press.

Chi, M., Glaser, R., & Rees, E. (1981). Expertise in problem solving. In R. J. Sternberg (Ed.), *Advances in the psychology of human intelligence* (Vol. 1). Hillsdale, NJ: Erlbaum.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences*. Mahwah, NJ: Erlbaum.

Evans, J. St. B. T., & Over, D. E. (1999). Explicit representations in hypothetical thinking. *Behavioral and Brain Sciences*, *22*(5), 763–764.

Glaser, R. (1984). Education and thinking: The role of knowledge. *American Psychologist*, *39*(2), 93–104.

Goldbeck, R., Bernstein, B., Hillex, W., & Marx, M. (1953). Application of the half-split technique to problem-solving tasks. *Journal of Experimental Psychology*, *53*(5), 330–338.

Handley, S. J., & Evans, J. St. B. T. (2000). Supposition and representation in human reasoning. *Thinking and Reasoning*, *6*(4), 273–311.

Jones, R., & VanLehn, K. (1991). Strategy shifts without impasses: A computational model of the sum-to-min transition. In K. Hammond & D. Gentner (Eds.), *Program of the Thirteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Klahr, D., & Carver, S. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, *20*, 362–404.

Klahr, D., & Nigam, M. (2004). The equivalence of learning paths in early science instruction: Effects of direct instruction and discovery learning. *Psychological Science*, *15*(10), 661–667.

Kuhn, D. (1991). *The skills of argument*. New York: Cambridge University Press.

Kurland, L., & Tenney, Y. (1988). Issues in developing an intelligent tutor for a real-world domain: Training in radar mechanics. In J. Psotka, L. D. Massey, & S. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned*. Hillsdale, NJ: Erlbaum.

Laird, J. E., & Newell, A. (1983). *A universal weak method* [Technical Report No. CMU-CS-83-141]. Pittsburgh, PA: Carnegie-Mellon University, Department of Computer Science.

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

Lee, A. (1993). Learning computer programming: A route to general reasoning skills? In C. Cook, J. Schultz, & J. Spohrer (Eds.), *Empirical studies of programmers: Fifth workshop*. Norwood, NJ: Ablex.

Means, B., & Gott, S. (1988). Cognitive task analysis as a basis for tutor development: Articulating abstract knowledge representations. In J. Psotka, L. D. Massey, & S. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned* (pp. 35–58). Hillsdale, NJ: Erlbaum.

McClelland, J., & Jenkins, E. (1991). Nature, nurture and connections: Implications of connectionist models for cognitive development. In K. VanLehn (Ed.), *Architectures for intelligence: The Twenty Second Carnegie Mellon University Symposium on Cognition*. Hillsdale, NJ: Erlbaum.

Metz, K. (1985). The development of children's problem solving in a gears task: A problem space perspective. *Cognitive Science*, *9*, 431–471.

Nisbett, R. (1993). *Rules for reasoning*. Hillsdale, NJ: Erlbaum.

Ohlsson, S., & Robin, N. (1994). The power of negative thinking: The central role of modus tollens in human cognition. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Polya, G. (1957). *How to solve it*. Princeton, NJ: Princeton University Press.

Pressley, M., Woloshyn, V., Lysynchuk, L., Martin, V., Wood, E., & Willoughby, T. (1990). A primer of research on cognitive strategy instruction: The important issues and how to address them. *Educational Psychology Review*, *2*(1), 1–58.

Rips, L. (1983). Cognitive processes in propositional reasoning. *Psychological Review*, *90*, 38–71.

Rouse, W. (1978). Human problem-solving performance in a fault diagnosis task. *IEEE Transactions on Systems, Man & Cybernetics, SMC-8*, 258–271.

Rouse, W., & Hunt, R. (1984). Human problem solving in fault diagnosis tasks. In W. Rouse (Ed.), *Advances in man–machine systems research* (Vol. 1, pp. 195–222). Greenwich, CT: JAI Press.

Ruiz, D., & Newell, A. (1989). Tower-noticing triggers strategy-change in the Tower of Hanoi: A SOAR model. In G. Olson & E. Smith (Eds.), *Program of the Eleventh Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Schaafstal, A. M., Schraagen, J. M., & van Berlo, M. (2000). Cognitive task analysis and innovation of training: The case of structured troubleshooting. *Human Factors*, *42*(1), 75–86.

Schaeken, W., De Vooght, G., Vandierendonck, A., & d'Ydewalle, G. (2000). Strategies and tactics in deductive reasoning. In W. Schaeken, G. De Vooght, A. Vandierendonck, & G. d'Ydewalle (Eds.), *Deductive reasoning and strategies*. Mahwah, NJ: Erlbaum.

Shepherd, A., Marshall, E., Turner, A., & Duncan, K. (1977). Diagnosis of plant failures from a control panel: A comparison of three training methods. *Ergonomics*, *20*(4), 347–361.
Siegler, R. (1989). How domain-general and domain-specific knowledge interact to produce strategy choices. *Merrill-Palmer Quarterly*, *35*(1), 1–26.
Siegler, R., & Jenkins, E. (1989). *How children discover new strategies*. Hillsdale, NJ: Erlbaum.
Smith, E., Langston, C., & Nisbett, R. (1992). The case for rules in reasoning. *Cognitive Science*, *16*, 99–102.
Sternberg, R. (1985). All's well that ends well, but it's a sad tale that begins at the end: A reply to Glaser. *American Psychologist*, *40*(5), 571–572.
Sternberg, R. (1989). Domain-generality versus domain-specificity: The life and impending death of a false dichotomy. *Merrill-Palmer Quarterly*, *35*(1), 115–130.
VanLehn, K. (1989). Learning events in the acquisition of three skills. In G. Olson & E. Smith (Eds.), *Program of the Eleventh Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.
VanLehn, K. (1991). Rule acquisition events in the discovery of problem solving strategies. *Cognitive Science*, *15*, 1–47.
Zimmerman, C. (2000). The development of scientific reasoning skills. *Developmental Review*, *20*, 99–149.

APPENDIX

Examples of domain-specific and RAA training materials for the experiment

Relevant RAA training. For the relevant RAA training condition, participants were given the following written statement of the RAA strategy as part of their training:

> If you have a hypothesis about what information is being passed along a certain line in the network, find out whether or not your hypothesis is contradicted by any information you know to be true about the network. If the hypothesis is contradicted by any information you know to be true, then the hypothesis is false. If the hypothesis only agrees with all the information you know to be true, then you can't tell whether it is true or false.

Irrelevant RAA training. The statement for the irrelevant RAA training contained only the first two of the above sentences. The last sentence was dropped because irrelevant RAA problems with an answer of "can't tell" did not involve agreement of the hypothesis with the known information, but rather insufficient information to tell whether the hypothesis contradicted known information. However, before doing the irrelevant RAA problems, participants in this condition received a clear explanation (based on two example problems demonstrated by the experimenter) of when to answer "can't tell". This explanation seemed to be sufficient, because participants in the experiment did just as well on the irrelevant RAA problems (90% correct) as on the relevant RAA problems (91%).