# LEARNING A TROUBLESHOOTING STRATEGY: THE ROLES OF DOMAIN SPECIFIC KNOWLEDGE AND GENERAL PROBLEM-SOLVING STRATEGIES

LEO GUGERTY
EDUCATIONAL TESTING SERVICE
PRINCETON, NJ

ABSTRACT

This research investigated how college students learned an efficient troubleshooting strategy, elimination. The subjects' task was to find the broken components in networks that were similar to digital circuits. With only minimal training in this task, subjects usually used a strategy of backtracking from the incorrect network outputs, instead of the more efficient elimination strategy, which involves backtracking but also eliminating (ignoring) components that lead into good network outputs. Computer simulation modeling suggested that in order for subjects to learn the elimination strategy on their own, they needed to apply (1) certain key domain-specific knowledge about how the components worked, and (2) the general reductio-ad-absurdum problem-solving strategy. An experiment showed that these two kinds of knowledge do enable students to increase their use of elimination, thus supporting the model.

## INTRODUCTION

Imagine you have rented a one-room cottage for the weekend. Upon arriving, you turn on the radio and no sound comes out, even with the volume control up high. A little investigating shows that the radio and a desk lamp are plugged into a well-worn extension cord, which is plugged into a wall outlet. Nearby is a fuse box with some replacement fuses and a new extension cord. What would you do first to find the cause of the silent radio?

There are a number of problem-solving strategies you could use in this situation. Based on your past troubleshooting experience, you could try to recall some of the likely causes of problems with radios. Or you could focus on the current situation and backtrack from the evident problem (the silent radio). This might lead you to replace the extension cord or some of the fuses. However, a more efficient strategy than simple backtracking would be to turn on the lamp. If it works, you can then eliminate the extension cord and fuse box as possible causes of the problem. In this situation, the elimination strategy isolates the problem to the radio or its cord. More precisely, in backtracking, the set of possibly faulty components consists of all those components that lead into the bad system output. In elimination, this set of possibly faulty components is reduced by ignoring the components that lead into good system outputs. The elimination strategy has also been called dependency-directed backtracking (Stallman & Sussman, 1977).

This research focused on how people learn the elimination strategy in a novel troubleshooting task. In particular, it considered how domain-specific knowledge and general-purpose problem-solving strategies are used in learning elimination. Psychologists have recently debated the relative importance of domain-specific versus general-purpose knowledge in problem solving (Glaser, 1984; Perkins & Salomon, 1989). Some suggest that all problem-solving strategies are closely tied to the specific domains in which they are used. Others claim that people do have general strategies that they can transfer to novel problems. Those who have written on this topic often have noted the paucity of research examining how domain-specific and general knowledge might interact during problem solving, and have called for such research (Alexander & Judy, 1989).

This study focused explicitly on the interaction between domain-specific and general-purpose knowledge. I first developed a simulation model that highlighted the domain-specific and general knowledge needed for learning elimination. Then I conducted a training experiment to test the model. This experiment determined whether either domain-specific or general knowledge alone would improve use of elimination, or whether both together were necessary.

The task for this research was not taken from everyday experience, as is the electrical troubleshooting problem above. Since I planned an experiment requiring a high degree of experimental control over the subjects' domain-specific knowledge, I chose a task that would be novel for my subjects (college students). Subjects had to find the broken nodes in very simple networks that were similar to digital circuits (see Figure 1). The network shown passes 0's and 1's from left to right. The nodes act as AND gates; when working correctly they only pass on 1's if all their inputs are 1's. When nodes break, they pass on 0's regardless of their inputs. The subjects searched for the broken node by testing particular connections between nodes to see if they were passing 0's or 1's, and by replacing nodes. The networks were presented on paper and subjects made tests and replacements by typing node numbers into a computer. Tests and replacements were assigned costs (in imaginary money), with replacements costing four times more than than tests. Subjects were asked to keep costs to a minimum. The network task was taken from the work of Rouse (1978).

For the network in Figure 1, the backtracking strategy leads to considering the following set of 18 possibly-faulty nodes: 10 through 23, 25 through 27, and 31. The elimination strategy allows many of these nodes to be ignored because they lead to an output of 1, leaving a possibly-faulty set of 4 nodes: 18, 22, 26, and 31. Pilot testing showed that college students usually did not use elimination when troubleshooting the network problems. Most resorted to the less efficient backtracking strategy.

The next section presents the model of the domain-specific and general-purpose knowledge needed to learn elimination.

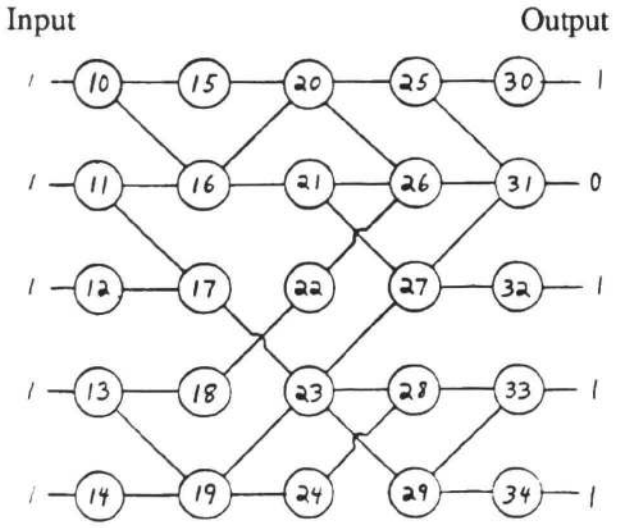


***Figure 1.*** **Example network**

## THE SIMULATION MODEL

I created separate production-system models for the backtracking and the elimination strategies. By looking at the knowledge that had to be added to the backtracking model to obtain the elimination model, I could predict what knowledge a person using only backtracking would need in order to learn elimination.

The backtracking model initially focuses on the node that gives the 0 network output, testing each input to this node (in a random order) until it finds a 0 input or discovers that all the inputs are 1. If all the inputs are 1, the model concludes that the current focus node is broken and replaces it. If a 0 input is found, the node outputting that 0 becomes the next focus node and the process is repeated.

When the backtracking model is looking for 0 inputs to a node outputting a 0, it generates hypotheses that each of the node's input lines are passing 0's and tests these by making actual tests of the network (as a subject would make on the computer). For example, given the network in Figure 1, it might generate the hypothesis that line 25 - 31 is passing a 0 and immediately test this. In the 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 kind of "what if" reasoning to propagate the effects of the hypothesis through the network. Propagation leads to the further hypotheses that line 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 assumed to be false. Thus, the model concludes that line 25 - 31 is passing a 1 and this line 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 tests the line associated with this hypothesis.

The elimination model uses two kinds of knowledge in this reasoning process. The first of these is the 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 0's. The second kind of knowledge used in the elimination model is the overall process of propagating a hypothesis, noticing contradictions, and falsifying the original hypothesis. This is basically the reasoning process known as **reductio ad absurdum (RAA)**.

The general-purpose nature of RAA should be emphasized. It can be used in mathematics, legal reasoning, and science. Polya (1957) included it in his book on general problem-solving skills. On the other hand, the node knowledge used by the elimination model is domain specific. In addition, it should be noted that the model does 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. Other rules about the nodes would not be expected to lead to the reasoning needed for elimination.

To summarize, the model suggested that people using only backtracking will learn elimination if they: (1) learn particular domain-specific knowledge about how the nodes work, and (2) apply the general RAA reasoning strategy. According to the model, both of these kinds of knowledge are needed for learning elimination. The next section describes the experiment that tested the model.

## THE EXPERIMENT

### METHOD

To test the model, I designed an experiment with five conditions, based on five ways of training subjects to do the network task. In each condition, subjects first received brief instruction in how the nodes and networks worked and how to do the task. The initial instruction contained enough information for the subjects to induce the elimination strategy, but this information was not highlighted. After the initial training, each subject received one of five kinds of extra training and then solved 24 network problems.

I designed the training conditions 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 I included conditions in which subjects received either no extra training (**baseline**), only the domain-specific node knowledge suggested by the model (**relevant node**), or both the domain-specific knowledge and the general RAA strategy (**relevant node/RAA**). In the relevant node/RAA condition, subjects learned RAA in the context of the network problems.

I also needed an RAA-only condition in which subjects learned RAA but not the relevant node knowledge. Since people are often bad at transferring knowledge across domains, it was important that subjects in the RAA-only condition learn RAA in the context of the network problems, as did the relevant node/RAA subjects. Thus, for the RAA-only condition, I taught RAA in the context of the network problems, but using domain-specific knowledge that was, according to the model, irrelevant to learning elimination. This is the **irrelevant node/RAA** condition. Finally, I added an **irrelevant node** condition where subjects learned only the irrelevant domain-specific knowledge.

The relevant domain-specific knowledge consisted of the two key rules about how the nodes worked that were used by the model to learn elimination. The irrelevant domain-specific knowledge consisted of rules about how the nodes worked that were true but, according to the model, irrelevant to learning elimination. (An example of an irrelevant node rule is: For a working node to have a 0 output, it must have at least one 0 input.) Comparing these two conditions (relevant node and irrelevant node) allowed me to test whether any increase in elimination after node training was due to the particular node knowledge indicated by the model or to general familiarity with how the nodes worked.

The main prediction from the model was that subjects in the relevant node/RAA condition would show the greatest use of elimination, since these subjects were explicitly taught all the knowledge sufficient for learning this strategy. Use of elimination in the irrelevant node condition was expected to be the same as in the baseline condition; since in both these conditions, subjects practiced none of the key knowledge needed for elimination. The model did not make a clear prediction regarding the relevant node condition and the irrelevant node/RAA condition, where subjects were taught only part of the knowledge sufficient for elimination. If subjects in these two conditions could infer the remaining needed knowledge from the initial training or from prior knowledge, then their use of elimination would increase. If not, it would stay the same.

Ten University of Michigan undergraduates participated in each condition. The overall procedure for each subject was: initial training, pretest (4 network problems), node training (depending on the condition), RAA training (depending on the condition), and post-test (24 network problems). After each post-test problem, subjects were told how much money they had spent and what would

be a "good" amount to spend on that problem. The good scores were based on using elimination plus the half-split strategy, which involves testing near the middle of a chain of possibly faulty nodes. During the pretest, subjects' only feedback consisted of how much money they had spent on each problem.

In the node training, a subject would see, for example, a diagram of an individual node with one of its input lines passing a 0 and one of its output lines marked with a question mark. The subject had to indicate what was being passed along the line with the question mark, either "0", "1", or "Can't tell". The correct answer in this case is "0". This problem exemplifies the relevant node rule that if a node has a 0 input, it outputs 0's. Subjects in the node training conditions did 96 node problems, with immediate feedback after each problem.

In the RAA training, subjects saw diagrams like in Figure 2. They were told to assume that the lines marked with a 1 and a 0 were actually passing those values and that the "0?" was a hypothesis. Their task was to determine, if possible, what value was being passed along the line with the hypothesis. The experimenter first demonstrated on a few problems how the hypothesized value could be propagated, and how sometimes, as in this problem, the hypothesis would lead to a contradiction and could therefore be falsified. The correct answer for this problem is "1". This is an example of relevant RAA training. Subjects in RAA training conditions did 24 RAA problems, with immediate feedback after each problem. The instructions emphasized that subjects could spend less money on the network problems by using RAA to test some of their hypotheses instead of making tests on the computer.

I would like to stress that the RAA training did not directly teach the elimination strategy. In well-practiced use of elimination on the network problems, subjects first step in problem-solving is usually to cross off the nodes that lead into outputs of 1. Then they direct their search for the faulty node to the remaining set of nodes. In the RAA training, subjects did something rather different. They used RAA to determine whether certain hypotheses about the network are true or false.

## RESULTS

A major advantage of the network task is that it allows easy measurement of subjects' strategy use merely by observing the tests they made. The main dependent variable for measuring use of

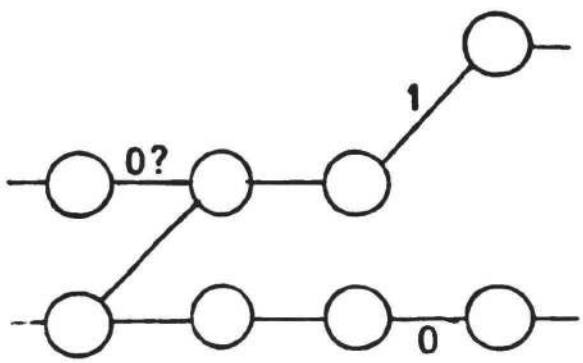


***Figure 2.*** **Example of a relevant RAA training stimuli.**

backtracking was the percentage of tests on each problem that were within the backtracking set A test is in the backtracking set if the line tested leads into the 0 network output. For the elimination strategy, a similar variable was calculated using the elimination set, the set of lines that lead into the 0 network output but not any network outputs of 1. For example, in Figure 1, the only tests consistent with elimination were those of the lines connecting nodes 18, 22, 26, and 31.

However, because the elimination set is contained within the backtracking set, subjects using only backtracking will by chance make some tests in the elimination set. I therefore calculated another variable to represent subjects' use of elimination that factored out elimination tests that would be expected merely by use of the backtracking strategy. For each network, I calculated the percentage of elimination tests that would be expected if a subjects used the backtracking strategy described by the model. This percentage was subtracted from the subjects' actual percentage of elimination tests on that network to give a new variable, called the percentage of elimination tests beyond chance.

On the 4 pretest networks (before any extra training), subjects used the backtracking strategy almost exclusively. Considering all 50 subjects, the average percentage of backtracking tests was 99%; while the average percentage of elimination tests beyond chance was 3%. An analysis of variance showed that the subjects in the five training conditions did not differ significantly in their use of elimination prior to training. However, subjects in some conditions (including the relevant node/RAA condition) did use elimination slightly more on the pretest. To handle these differences, subjects' pretest scores on a dependent variable were used as a covariate when analyzing the post-test data.[1]

Subjects did quite well on the node and the RAA training. They answered 97% of the node problems and 92% of the RAA problems correctly, with no significant differences between the relevant and irrelevant training.

Figure 3 shows the subjects' use of elimination on the post-tests. The data for each subject were averaged over the 24 post-tests. The figure gives the adjusted means from the analysis of covariance, since these are the means that would be expected after any preexisting (i.e., pretest) differences in use of elimination have been factored out. As the figure shows, post-test use of elimination after relevant/RAA training was significantly greater than in the baseline condition ($p < 0.05$). The other three training conditions showed no improvement over the baseline. This was true for both the percentage of elimination tests and the percentage of elimination tests beyond chance.

To put these data in context, the straight line at 48% shows the percentage of elimination tests that would be expected given use of the backtracking strategy as implemented in the model. Using the modeled elimination strategy would lead to 100% elimination tests. The modeled backtracking and elimination strategies would result in values of 0 and 48%, respectively, for the percentage of elimination tests beyond chance. Thus it seems that the two kinds of training suggested by the model, relevant node and RAA training, did help people use elimination more often. In fact, they increased the above-chance use of elimination by at least a factor of two.

---

[1]The difference between pretest and post-test scores was not used as a dependent variable because different kinds of feedback were used on the pretests and post-tests, and there were many fewer pretests than post-tests (4 vs. 24).

This conclusion is further supported by analysis of other aspects of subjects performance on the post-tests. Subjects in the relevant node/RAA condition also made fewer tests and took longer to make their tests than subjects in the other conditions. The overall picture is that subjects in the relevant node/RAA condition were using elimination extensively, while subjects in the other four conditions used it only slightly above the levels expected due to chance.

Thus the experiment shows that in order to learn the elimination strategy in the network task, college students need explicit training that conveys both of the kinds of knowledge suggested by the model, domain-specific knowledge of how the nodes worked and knowledge of the general RAA strategy. The experiment supported the main conclusion based on the model - that learning elimination depends on the interaction of both domain-specific and general knowledge. Furthermore, not any kind of domain-specific knowledge will help. Only the key domain-specific knowledge highlighted by the model leads to increased use of elimination, when paired with the appropriate general strategy.

## CONCLUSION

My initial question concerned how domain-specific and general-purpose knowledge interact to allow learning of problem-solving strategies in novel domains. Simulation modeling proved quite helpful in this research. It allowed precise identification of the kinds of domain-specific and general knowledge that might be involved in learning a particular troubleshooting strategy, elimination. The experiment supported the model and suggested that the general reductio-ad-absurdum strategy and certain key domain-specific knowledge, when learned together, substantially increase the use of the elimination strategy. The experiment reported here also provides empirical support for AI models such as SOAR, which suggest that problem-solving

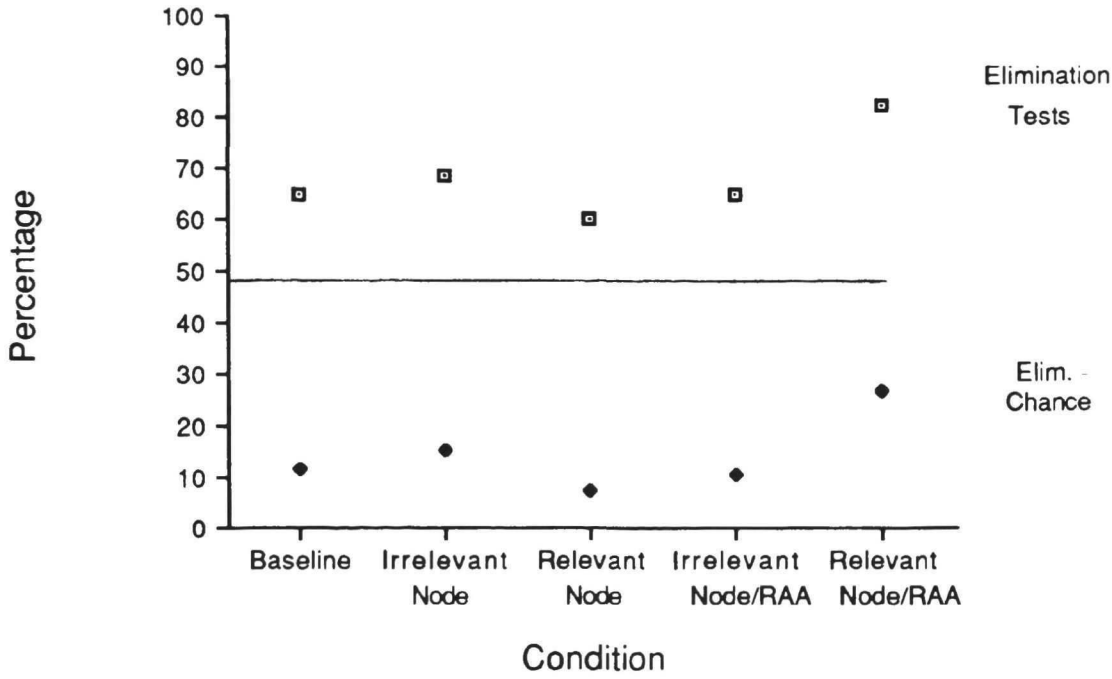


***Figure 3.*** **Post-test use of elimination (adjusted means)**

strategies in a domain can be induced using domain-specific knowledge and very general problem-solving strategies (Laird & Newell, 1983). The students in this experiment were not taught elimination directly; rather, they induced it using domain-specific knowledge and RAA.

Because of the training design used in the experiment, this research can also suggest answers to educational questions concerning how general problem-solving strategies can be taught. One such question is whether general problem-solving strategies can and should be taught independently of domain-specific knowledge. This research argues against an extreme "independence" position, at least for the elimination strategy. Domain-specific knowledge was found to be essential to using the general RAA strategy to learn elimination.

Many avenues are open for further research. If research such as this is conducted on other strategies and other domains, we will begin to fill in the gaps in our understanding of how domain-specific knowledge and general strategies interact in problem solving. Also, one could address the question of whether the students who learned the elimination strategy in this experiment could transfer this knowledge to other domains. Elimination is a general-purpose strategy, which can be used in many kinds of troubleshooting tasks, such as computer-program debugging and medical diagnosis, as well as in searching for lost objects. Perhaps because these students induced the elimination strategy from more basic knowledge, they would have a deep understanding of it and thus be able to apply it in other domains.

---

This report is based on my dissertation, which was submitted to the Psychology Department at the University of Michigan. I would like to thank my doctoral committee for their help, and also Irving Sigel and Drew Gitomer for their comments on this report.

## BIBLIOGRAPHY

Alexander, P. A. & Judy, J. E. (1989). The interaction of domain-specific and strategic knowledge in academic performance. *Review of Educational Research, 58(4),* 375-404.

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

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.

Perkins, D. N. & Salomon, G. (1989). Are cognitive skills context bound? *Educational Researcher, 18(1),* 16-25.

Polya, G. (1957). *How to Solve It.* Princeton, NJ: Princeton University Press.

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

Stallman, R. M. & Sussman, G. J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence, 9,* 135-196.