# Modeling Analogical Problem Solving in a Production System Architecture

Dario D. Salvucci and John R. Anderson

July 1, 1996
CMU-CS-96-151

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research is supported by a National Science Foundation Fellowship awarded to Dario Salvucci and Office of Naval Research grant N00014-96-1-0491 awarded to John Anderson.

**Keywords**: Cognitive science, cognitive modeling, analogy, ACT-R.

## Abstract

We propose an approach to developing models of analogical reasoning within production system architectures. Though existing theories of analogy have succeeded in capturing many aspects of analogical behavior, they have several limitations. First, the theories use empirical support that focuses almost exclusively on high-level data that illustrate the results of analogy, ignoring low-level data that illustrate step-by-step processes during analogy. Second, the theories cannot fully account for variability prevalent in analogical behavior, nor can they account for the adaptation of analogical strategies in learning. Third, some of the theories cannot be readily incorporated into a unified theory of cognition. We show how production rule models of analogy can address and to some extent overcome these limitations. As our exemplar, we describe empirical and modeling results for a task in which subjects solved simple physics problems by analogy. The empirical results, which include both high- and low-level data, give evidence that subjects use multiple analogical strategies and shift between strategies. The modeling results show that an ACT-R (Anderson, 1993) production rule model of the task can account for much of subjects' observed behavior. We also present a model for a similar analogical task involving picture analogies (Sternberg, 1977) to illustrate how the simple physics model can generalize to other tasks.

# 1. Introduction

Recent times have witnessed the birth and development of several unified cognitive theories which use production rules as the building blocks of problem-solving knowledge (e.g., Anderson, 1993; Just & Carpenter, 1992; Kieras & Meyer, 1995a; Laird, Newell, & Rosenbloom, 1987). Production rule systems allow for a great deal of flexibility in modeling, and can represent both high- and low-level thought processes. The systems also predict observable events in time, so solid empirical support can be gathered to test and evaluate them in a straightforward manner. As these systems have proven useful for modeling a wide range of tasks, we employ them on a phenomenon central to human learning—analogy. After a brief overview of analogy, we consider how a production rule framework can capture analogical behavior. We then examine subject performance on a simple physics problem-solving task, looking at high-level data (total latency, correctness, etc.) as well as low-level data (visual scans, key presses, etc.). We also present a model developed under the ACT-R framework (Anderson, 1993) which succeeds in predicting much of the experimental data collected. We conclude by discussing related work and the implications of our work for other research in analogy.

Numerous theories of analogy and analogical reasoning have emerged in the past two decades (e.g., Gentner, 1983, 1989; Holyoak & Thagard, 1989a, 1989b; Keane, Ledgeway, & Duff, 1994). These theories approach analogy in its most general sense, describing how some process of mapping can infer relations between concepts. Colloquially, people often use the term "analogy" to refer to this process of mapping, or finding correspondences between two conceptual structures; for instance, students often think of analogy in terms of problems like

herd : buffalo : school : ?

for which the student must infer the relation between the first and second objects and apply it to the third to obtain the solution (in this case, fish). Researchers have overwhelmingly agreed that mapping is the core essential component of analogy, though each has taken a slightly different approach to the problem. Gentner's (1983, 1989) influential research has postulated that mapping centers on finding structural relations between concepts, with a certain systematicity that favors higher-order relations. Holyoak and Thagard (1989a) have used similar ideas to implement an analogy mechanism based on constraint satisfaction. Keane et al. (1994) presented an incremental mapping engine that incorporates constraints on working memory. Regardless of their particular approaches, these and other theories have considered mapping the centerpiece of the analogy process.

Successful problem solving by analogy requires more than just mapping, however. First, it is necessary to represent the problem in a way that allows or facilitates analogical mapping. Representation, or encoding, of the mental structures that participate in mapping is crucial to the ability of any theory to construct analogical mappings. Of course, representation is important not only for analogy, but for all problem solving; as just one example, Hinsley, Hayes, and Simon (1977) have discussed the importance of representation for word problem solving. Representation is pervasive in the analogy process, and though we sometimes label it as the first subprocess of analogy (e.g., Reeves & Weisberg, 1994), we may more appropriately think of it as a precondition for successful analogizing.

Another key element of analogical reasoning is the retrieval of an appropriate source analog for mapping. The source may constitute some mental structure stored in memory (e.g., Anderson & Thompson, 1989), or may come from the external world in another form, such as written text (e.g., VanLehn & Jones, 1993). Successful retrieval proves to be extremely difficult for humans in certain situations, as has been manifested in several experiments using a hint/no-hint paradigm. For instance, in Gick and Holyoak's (1980) study based on Duncker's (1945) tumor problem experiment, subjects heard a story of how a general attacked a well-defended fortress and were then asked to determine how a doctor might eradicate a tumor; the solution for the doctor-tumor problem was analogous to the solution for the general-fortress problem. Some subjects were told explicitly that the general-fortress story provided a useful analogous solution, while the others were given no such hint. Gick and Holyoak found that 92% of subjects in the hint condition answered the problem correctly, while only 20% of the no-hint subjects gave a correct response. Such studies provide strong support that the retrieval stage is crucial to analogical mapping, and that retrieving an appropriate source analog can at times be very difficult.

Some researchers have named a fourth essential component of analogy, that of schema induction. Anderson (1993) and Anderson and Thompson (1989) have used analogical reasoning in the creation of new production rules that act as schemata for similar problems. Novick and Holyoak (1991) provided evidence that schema induction is a natural consequence of successful analogical transfer. Ross and Kennedy (1990) found that the cueing of memorized sample problems can facilitate generalization of already known formulas. The literature is somewhat undecided on whether schema induction occurs during mapping, as a separate stage, or along with some other component of analogy. Nevertheless, further use of analogical mappings after the initial application seems to involve inference of some schematic knowledge.

These four components (representation, retrieval, mapping, and induction) all play a role in a person's "strategy of analogy" for a particular task. We will use the term "strategy of analogy," or "analogical strategy," to refer to the step-by-step cognitive processes executed while solving a problem by analogy. For instance, consider the following verbal analogy problem discussed by Grudin (1980):

Thursday : Monday :: Friday : ?

Grudin reported that subjects exhibited two different strategies while solving this problem. Some subjects noted that Friday is the day after Thursday, and thus concluded that the answer is the day after Monday, namely Tuesday. Other subjects computed the number of days between Thursday and Monday, then counted off the days from Friday to determine the solution. Thus, subjects can solve this analogy using one of at least two distinct strategies of analogy, and possibly others. Our goal in modeling analogical reasoning in problems like this one is to account for the analogical strategies exhibited by subjects solving the problem.

Existing theories of analogy have modeled many aspects of analogical behavior. However, most theories tend not to use detailed empirical data concerning what happens during the analogy process, focusing instead on the results of the analogy process. For example, Keane et al. (1994) presented empirical data for subjects' performance on the attribute mapping task. They compared their empirical data to the predictions of three analogical theories: SME (Falkenhainer, Forbus, & Gentner, 1989), ACME (Holyoak & Thagard, 1989a), and their own IAM. Since none of the theories make predictions about directly observable quantities like latency, Keane et al. compared the theories by relating subjects' total time for completion to the number of mappings generated by the models. For comparison purposes, this metric works well in showing how IAM captures several aspects of analogical problem solving in the attribute mapping task. However, arguing for the plausibility of such metrics in general could be difficult, and the metrics may not generalize when considering observables during the analogy process. Ideally, we would like the theory to make predictions about both observable events during the analogy process and observable results of the process. Some work has been done in this vein (e.g., Sternberg, 1977; Sternberg & Gardner, 1983), but there have not been detailed process models of analogical strategies.

Another limitation of many existing theories is their inability to account fully for variability and adaptation of analogical strategies. Researchers have found that people use different strategies when analogizing (e.g., Chi, Feltovich, & Glaser, 1981; Novick, 1988; Spellman & Holyoak, 1993; Whitely & Barnes, 1979) and can adapt these strategies during learning (e.g., Zhu & Simon, 1987). Existing theories that define a single deterministic mechanism to implement analogy can account for some variability, to the extent that the theory expresses the differences in terms of the problem representation. Such theories make the implicit assumption that all humans with the same knowledge representation perform analogy in the same way. However, representation is not the only variant when considering analogy; people with similar backgrounds, and even the same person on separate occasions, may analogize in different ways. While these single-mechanism theories may capture aggregate subject performance, they cannot account for such variability in analogical strategies, nor can they predict adaptation or learning of these strategies. Thus, another goal for a theory of analogy is to account for what analogical strategies subjects utilize and how they shift between strategies.

One further limitation of some theories of analogy is that they are not incorporated into a more general processing system. Analogy in the real world arises in various forms, each of which has unique aspects to its solution, just as any problem-solving task may have domain-specific components. As such, analogy encompasses a broad spectrum of tasks that involve not only analogical reasoning, but a great number of other skills. For example, in Grudin's days-of-the-week analogy above, both of the proposed strategies require domain-specific knowledge; namely, the strategies necessitate knowledge of the days of the week, the order in which they occur, and how to count the days between them. Any approach to analogy which does not incorporate domain-specific knowledge cannot account for such phenomena. Thus, any theory of analogy must lie within a more general processing system, as do some existing theories (e.g., Anderson & Thompson, 1989; Holyoak & Thagard, 1989b).

To summarize, the goals for a theory of analogy include incorporating the theory into a general processing system, creating detailed process models of analogical behavior, and predicting variability and adaptation of analogical strategies. In the remaining sections, we address how production system architectures can help achieve these goals.

# 2. The Production Rule Approach to Analogy

## 2.1. Overview of Production System Architectures

In recent decades, several major cognitive architectures based on production rules have emerged; among others, ACT-R (Anderson, 1993), Soar (Laird, Newell, & Rosenbloom, 1987; Newell, 1990), CAPS (Just & Carpenter, 1992), and EPIC (Kieras & Meyer, 1995a) have enjoyed some success in the modeling world. Though their particular implementations differ, these architectures have several common goals. They all provide a unified approach to general problem solving, that is, they can model any problem-solving task. The architectures also make specific predictions about observable data, allowing direct comparison with experimental results. Given these advantages, the theories have given rise to modeling work in a broad range of areas, including human-computer interaction (John, Vera, & Newell, 1991; Kieras, Wood, & Meyer, 1995b), natural language comprehension (Just & Carpenter, 1992; Lehman, Lewis, & Newell, 1991), working memory and memory span (Anderson & Matessa, in press; Anderson, Reder, & Lebière, in press), visual attention (Anderson, Matessa, & Douglass, 1995), robotic planning and execution (Laird & Rosenbloom, 1990), reasoning (Polk & Newell, 1995), strategy selection (Lovett & Anderson, 1995), and mathematical problem solving (Anderson, Corbett, Koedinger, & Pelletier, 1995).

All production system architectures have production rules as the basic units of skill-related knowledge, represented as a condition-action pair. The condition tests whether the current situation matches the situation in which the rule would be active; conditions may test aspects of both the external world, such as visual information, and internal states, such as the knowledge brought to a task. When the condition holds, the production rule may fire, causing its action side to take effect. Rule actions can modify the internal state of the model, like adding a unit of knowledge, or they can perform some external action, like a motor response. Cognition, then, is simply the sequence of matching rule conditions and firing rule actions.

When the conditions for several productions match the current situation, the architecture must define what subset of the competing productions will fire. Some architectures rely on conflict resolution, or the process by which the system chooses one production to fire. Which production the system selects can depend on many factors: the number of times the production has fired, the recency of its firings, the ease of retrieving matches for the condition, the frequency with which its actions lead to successful states, etc. Some architectures allow firing and execution of productions in parallel, and thus need not decide on one particular production—all competing productions can fire simultaneously.

In a later section, we explore the details of our exemplar production system architecture, ACT-R. For now, though, this overview of production rule systems is sufficient for describing a general approach to analogy within a production rule framework.

## 2.2. Analogy in a Production System Framework

Our production system approach to analogy centers on the belief that analogy is fundamentally a problem-solving skill. Modeling analogy within a production system framework simply means creating a production rule model of an analogical task. The model would incorporate all skills necessary for the task, including analogical and domain-specific skills. In some sense, then, we model analogical tasks as we would any problem-solving task. However, models for analogical tasks all share a specific set of productions that implement analogy. In other words, the common thread that binds models of analogical tasks is the presence of these analogy productions. Production rule models already incorporate this idea for other common processes (see Singley & Anderson, 1989); for instance, two models that require adding numbers could use the same productions to implement addition. Drawing an analogy to computer programming, we can think of the common productions as implementing a subroutine that would be called in any task requiring a specific type of analogy. The analogy "subroutine" could then call domain-specific subroutines that implement mapping, induction, etc. for a particular task.

Production system architectures can address all three goals for a theory of analogy described earlier. First, production system models of analogy are necessarily incorporated into a general processing system, since they are designed within the framework of general problem-solving architectures. The productions that implement analogy can be easily combined with models of domain-specific skills. This allows for the integration of analogical reasoning and other skills necessary to solve problems.

Second, production systems allow detailed, low-level process models that can capture behavior exhibited during the analogy process. Given that production rule models are powerful enough to model this behavior, it is important that comparison of model predictions to empirical data is relatively straightforward. As productions fire, their action sides can execute both unobservable actions, such as modifications to memory chunks, and observable actions, such as motor responses. A model trace of a particular run includes information about what behavior occurs and when it occurs (assuming the architecture predicts real-time latencies). Typically the "behaviors" exhibited in a production model trace are closely related to real-world behaviors; for instance, the model presented herein predicts times and locations of all mouse clicks during the task. Production rule model predictions thus allow easy comparison to both high-and low-level empirical data which illustrate subjects' strategies of analogy.

Third, production system models of analogy can address the issue of variability in analogical strategies. Production rule models have several ways of reproducing variance between and within individuals. Conflict resolution allows for possibly stochastic choices among productions, so that the model can utilize different strategies on different runs (e.g., Lovett & Anderson, 1995). Different strategies can be used for both domain-specific and analogical skills. Thus, production systems allow for multiple implementations of the analogy process and can decide between them probabilistically. Certain architectures also allow for erroneous firings of productions that mimic systematic errors in problem solving, another source of subject variability (e.g., Lebière, Anderson, & Reder, 1994). Such techniques provide methods of generating different traces for each run, such that one run represents the actions of one "simulated individual." Of course, a modeler can easily derive aggregate results for the model by averaging traces from multiple stochastic runs.

Production systems also allow learning and adaptation of analogical strategies. Most architectures allow learning on both a symbolic and subsymbolic level. At the symbolic level, new knowledge units, and even new productions, can be acquired during the learning process. At the subsymbolic level, the system can adjust continuous parameters of the model, such as the activation of knowledge units or the firing strengths of productions. Such learning can help account for strategy

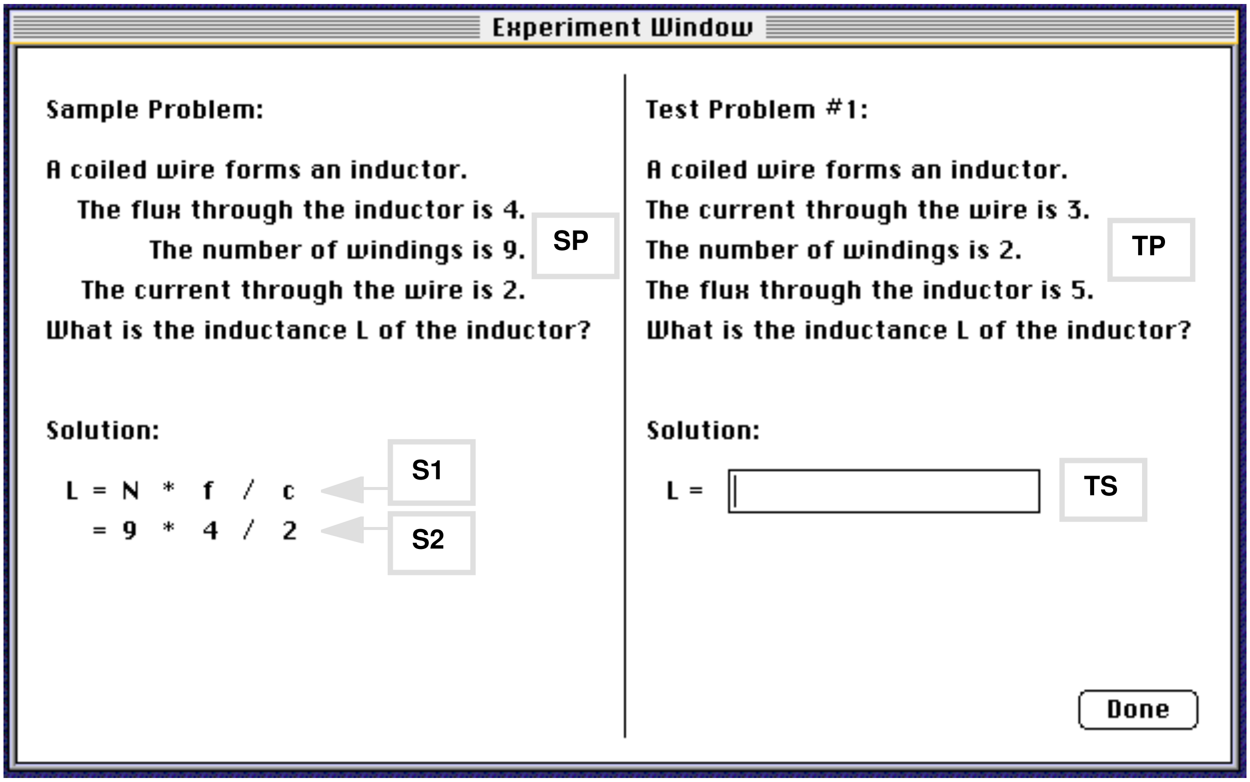


Figure 1. Simple physics experiment screen, without blocks. The labels in grayed boxes indicate the names of the visual areas.

changes in analogical (as well as domain-specific) skills. Thus, production systems can handle not only multiple analogical strategies, but also the shifts between multiple strategies during learning.

Production systems thus provide integration of analogical and domain-specific knowledge, straightforward comparison of process model predictions to empirical data, and the ability to model variability and adaptation of analogical strategies. Interested readers can refer to Schunn & Klahr (in press) for further discussion and general advantages of production system architectures. We now proceed to analyze in detail a particular task in which subjects solved simple physics problems by analogy. Our presentation begins with a description of the task and a discussion of the data collected in the experimental study. We then present the specifics of the production rule model and evaluate how its predictions match the empirical data collected.

## 3. The Simple Physics Task

Our study of the simple physics task examines the analogical strategies utilized by subjects in solving simple physics problems. We consider data that illustrate not only the results of the analogy process (high-level data), but also the strategies exhibited during the process (low-level data).

The experimental task involved solving several sets of simple physics problems by analogy to a given sample problem and solution. We chose this task for several reasons. First, many researchers have used physics problems in exploring the role of analogy (e.g., Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Chi, Feltovich, & Glaser, 1981; VanLehn & Jones, 1993); thus the physics domain is a familiar one in the literature, facilitating comparison of other work to our own. Second, the task allowed relatively terse problems and solutions with a small natural language component, allowing us to ignore language comprehension to a large extent. Third, the task is representative of real-world physics homework from a textbook; we can imagine the student doing a test problem at the end of the chapter, referring back to a worked-out example in an earlier section.

Table 1. Experiment topics and equations.

| Set | Topic | Equations | PC | NC |
|---|---|---|---|---|
| 1 | capacitance | area / spacing | A/s | s/A |
| 2 | thermal resistance | thickness / conductivity | T/c | c/T |
| 3 | contracted length | length / factor | L/f | f/L |
| 4 | force | – constant * distance | –c*d | –d*c |
| 5 | velocity | c / index | c/i | c/i |
| 6 | slit width | number * wavelength / sin angle | n*w/Sa | w*a/Sn |
| 7 | volume | R * temperature / pressure | R*T/p | R*p/T |
| 8 | inductance | number * flux / current | N*f/c | c*N/f |

In the task, problems were presented to subjects on computer screens similar to (but not exactly like) Figure 1. The left half of the screen contained the worked-out sample. The sample problem first gave a description of the problem situation, then listed the relevant quantities (in short phrases) with their respective values. The sample solution used variables (e.g., "P" for pressure), operators (e.g., "+"), and constants (e.g., "c" for speed of light) in the first step; and values (e.g., "4"), operators, and constants in the second step. The right half of the screen contained the test problem to be solved and an editable text field in which to enter the answer. The test problem had a structure similar to the sample problem, except for different quantity values and a possibly different ordering of quantity-value pairs. For future reference, we define the screen's visual areas (labeled in Figure 1 in grayed boxes) as: sample problem quantities and values (SP), sample solution step 1 symbols (S1), sample solution step 2 symbols (S2), test problem quantities and values (TP), and a test solution block (TS).

Subjects were asked to solve sets of five problems in each of eight topics, all dealing with basic physics and all involving instantiation of a single equation. Table 1 shows the equations used for each topic; the first three equations are of equal complexity, while the others range in difficulty from fairly easy to fairly difficult. All topics and equations were taken from a standard physics textbook (Halliday & Resnick, 1988), massaged in some cases to fit the task, but still mathematically correct. Subjects were to type their answers as instantiated equations, without simplifying; for example, they were to type answers such as "3*6/2," instead of simplifying this quantity and entering "9." In summary, the task involved learning a schematic equation such that values could be plugged in for quantities to produce the desired answer.

The task certainly involves analogy, assuming we define analogy by the stages of representation, retrieval, mapping, and induction described earlier. First, subjects must visually process the information on the screen to build up a mental representation of the problems. Second, subjects must retrieve the sources of analogy (i.e., the sample problem and solution) from memory, reading or reviewing parts of the screen if necessary. Third, subjects must infer a mapping between the sample problem and solution, applying the mapping to the test problem to obtain the test solution. Finally, the subject can perform induction by memorizing a schematic equation that facilitates the solution of later problems. We can also think of the task in terms of the standard A:B::C:D schema for analogy problems, where A is the sample problem, B the sample solution, C the test problem, and D the test solution.

To shed light on subjects' behavior during the analogy process, we needed some way of recording subjects' scanning and typing activity during the task. We modified the program so that parts of the screen were covered with opaque blocks, which would disappear when clicked on and reappear on release of the button. An actual experiment screen, namely the screen in Figure 1 with blocks included, is shown in Figure 2. The program concealed each symbol in S1 and S2, allowing us to examine which

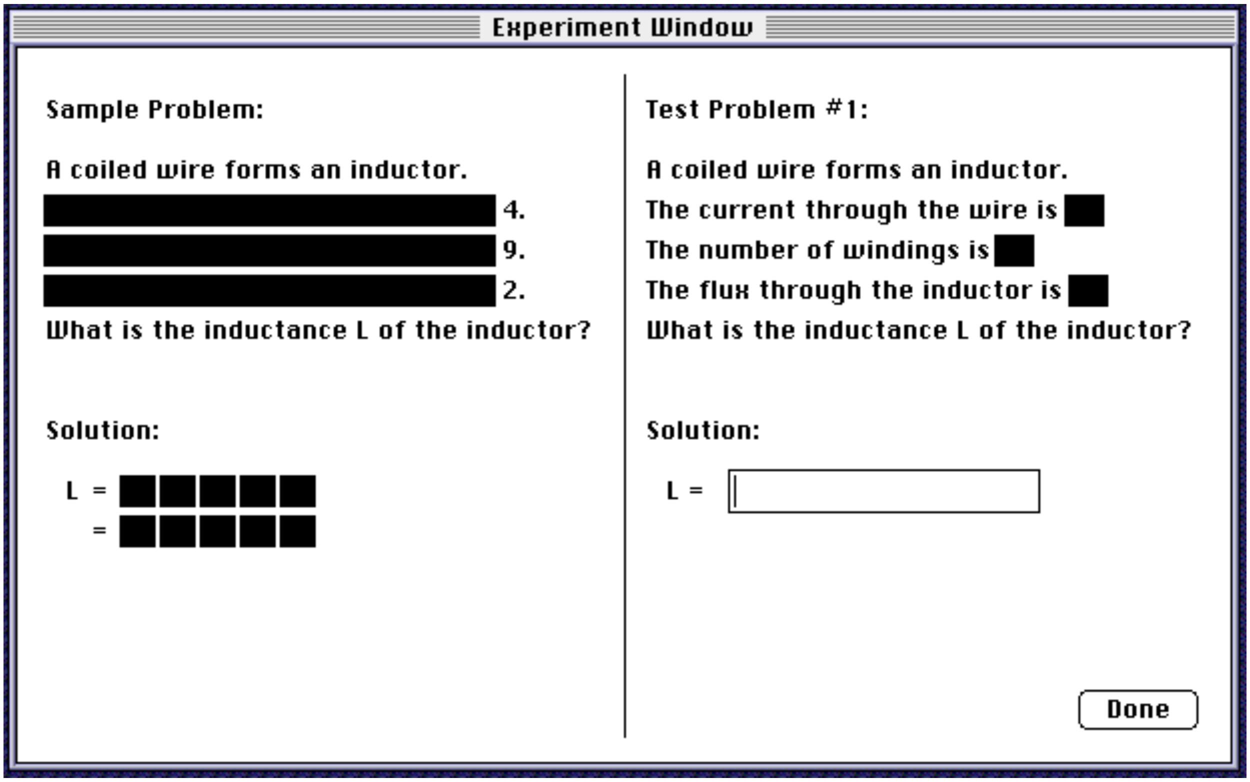


Figure 2. Simple physics experiment screen, as seen by subjects.

equation and symbol subjects were scanning. Blocks also covered the SP quantities and the TP values, but left the SP values and TP quantities uncovered. The reason for leaving the latter pair uncovered centered on the strategies expected of subjects. When a subject reads a value in S2, we expect them to search for that value in SP; to facilitate the search, and to obtain simpler data, we left the SP values uncovered so that subjects can quickly locate the desired value. Similarly, when typing the solution equation, subjects search for a TP quantity to obtain its value; we thus uncovered the TP quantities. One could argue that the setup of the blocks changes the character of the task, since the blocks may alter strategies subjects utilize during the analogy process. Though this argument may be valid, it does not change the fact that the task is fundamentally analogical and can provide rich empirical data concerning strategies of analogy.

The block approach for recording scanning activity is an adaptation of a common information-search paradigm where text or pictures appear on the backs of cards, such that subjects must turn the card over to reveal the information. For example, Payne (1976) used this paradigm in studying strategies of gathering evidence for decision making. Our technique differs from Payne's in that the information is re-concealed after being looked at, whereas the information in Payne's experiment remained visible for the duration of the task. We used the re-concealing technique to record not only the order in which symbols are scanned, but also how often they need to be scanned. A third option for uncovering blocks would not require the subject to click on the block, but only move the pointer to it (e.g., Anderson, Matessa, & Douglass, 1995). We decided against this option to avoid an inevitable flux of unintended references in the data, when subjects incidentally uncover blocks as they move to the intended block.

During an experiment, the program recorded two types of information: the time, location, and duration of each mouse click; and the time and character of each key press. Therefore, the program not only logged subject responses after each problem, but also much of how subjects went about solving the problem. In fact, the experiment program (in a different mode) can read in subject data files and simulate all mouse clicks and key presses, allowing the experimenter to "watch" the subject at work.

In order to induce subjects to learn different analogical strategies, we manipulated the presentation of variables in S1. Subjects were randomly assigned to one of two conditions. In the positive-correlation (PC) condition, each variable was named using the first letter of the quantity it represented; for instance, m would represent mass and L length. In the negative-correlation (NC) condition, each variable was named using the first letter of a quantity it did not represent; for example, L might represent mass and m length. Thus, while the PC variables suggested correct relations, the NC variables deliberately misled the subject by suggesting incorrect relations. The equations used in each condition appear in Table 1. Note that in both conditions, the S2 values did correspond to the correct quantities, and so the solutions remained correct—the only discrepancy between conditions is the mapping from variable to quantity.

Because of the different variable namings in the conditions, we can predict different strategies for the two groups. PC subjects, encountering names they would likely expect, should utilize a "solve-by-variable" strategy. That is, they should eventually tend to focus on S1 rather than S2, using variables more heavily than values for inferring the correct schema.[1] Indeed, the most efficient strategy for PC subjects involves looking only at S1; the subject can infer mappings from variable to quantity by referring to the visible TP quantities. NC subjects should utilize a "solve-by-value" strategy, relying more heavily on SP and S2. Here the most efficient strategy involves finding a corresponding SP quantity for each S2 value and storing the mapping from value to quantity. Of course, since the S1 variable names are misleading, NC subjects can get no (or little) useful information from the variables in the sample solution. In addition, since we expect NC subjects to be deceived initially by the incorrect namings, we also expect them to make more errors in the initial sets of the experiment.

## 3.1. Method

Subjects. A total of 38 Carnegie Mellon undergraduates, 6 women and 32 men, participated for course credit. The subjects were randomly assigned to either the PC or NC condition, with a total of 19 subjects per condition. Three additional subjects participated but were excluded from the analysis; one subject failed to follow instructions, the second produced inexplicable repeated clicks, and the third did not complete the task within the allotted time.

Design. The study's independent factor was the naming of variables in the S1 equation. The PC condition used the expected variable namings, while the NC condition used deliberately misleading namings. The resulting data from the study came in the traces produced by the experiment program for each subject. The traces gave detailed information about mouse clicks, key presses, and final answers observed during the experiment.

Materials. We designed eight sets of five simple physics problems, where each set dealt with a different topic (as shown in Table 1). The equations were adapted from those in a standard introductory physics textbook (Halliday & Resnick, 1988) and simplified to use only basic arithmetic operations. For each topic, we created a sample problem and five test problems. The sample problem's solution comprised a step with the variables and operators, and a step where the variables were replaced with their corresponding values. In the PC condition, the variables were named with the first letter of the represented quantity; in the NC condition, they were named with the first letter of a different quantity. The numeric values used were constrained to one-digit integers, so that typing times would be comparable across all problems. The Appendix lists the sample problems and solutions used in each of the eight topics.

Subjects performed the task using a Macintosh application that presented each problem individually, one per screen. The screen (see Figure 2) comprised the sample problem and solution on

[1] It is feasible that PC subjects might use the S2 values exclusively when analogizing, which would in fact yield correct results. However, the empirical results indicate that very few subjects follow this trend.

the left and the test problem on the right. The program presented the eight sets in the same order, and the five problems within each set in the same order. The same sample problem and solution were used for all problems within a particular set. Also, the order of the quantities in each test problem varied from screen to screen. The program covered certain areas with black blocks, which could be uncovered by clicking and would reappear upon release. The areas covered by blocks were the SP quantities, the S1 and S2 symbols (one block per symbol), and the TP values. During an experiment the program recorded the time and position of all clicks, plus the time and key of all key presses. No feedback was given for either correct or incorrect answers.

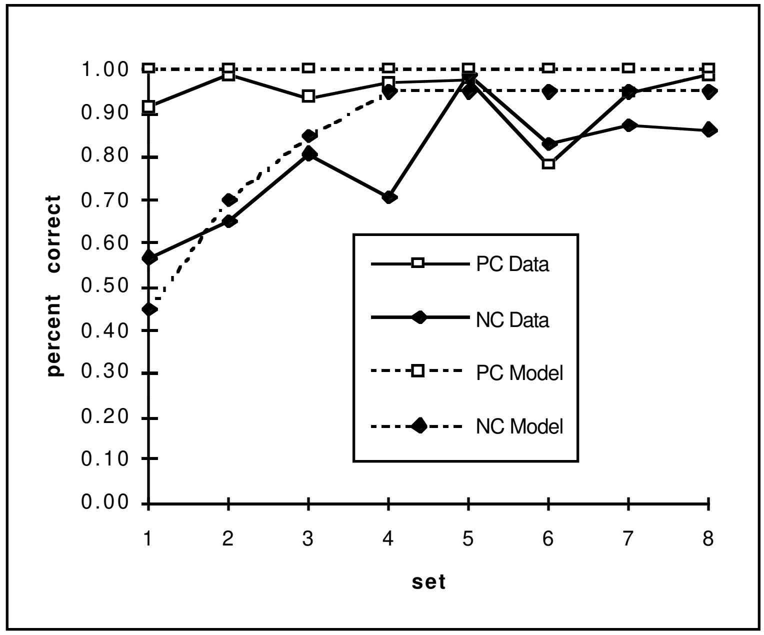


Figure 3. Average correctness for all sets. Correctness denotes the percentage of the five set problems answered correctly. Solid lines show subject data, dashed lines show learning model predictions.

Procedure. The experiment was performed within a 45 minute period. Initially, subjects were given a brief introduction to the task, allowed to practice uncovering blocks, and instructed to enter answers as unsimplified instantiated equations. When ready, they began solving the 40 problems in the task. First, a screen appeared noting that they were beginning a new set of problems. The subjects then solved the five problems in the first set, clicking the lower-right button to indicate completion for each screen. Then subjects were warned of another new topic, and went on to solve the next five problems. They continued solving problems in this manner until all eight sets were completed. Subjects were given a similar but more complex task for the remaining time; we will not address this task in our discussion. Finally, subjects were debriefed and thanked for their participation. During the task subjects received no assistance from the experimenter and were forbidden from using external materials such as pencil and paper.

## 3.2. Results

We discuss the results in two stages. First, we consider high-level data for the entire task, examining total latency and correctness per set of problems. Second, we look at low-level data for the final set, since we expect fairly stable behavior by the end of the experiment. Since the sets were presented in the same order, the final set always corresponded to the final topic in Table 1. On average, subjects required approximately 20 minutes to complete all 40 problems.

We begin by analyzing the correctness summary in Figure 3, where solid lines represent subject data in each condition. PC subjects worked essentially at ceiling, hovering around 95% correct for all sets. NC subjects exhibited more numerous errors in the initial sets, but eventually approached ceiling as well. A two-factor analysis of variance (ANOVA) shows that the effects of set, $F(7,288) = 2.59$, $MSE = 5.63$, condition, $F(1,288) = 20.0$, $MSE = 43.50$, and their interaction, $F(7,288) = 2.50$, $MSE = 5.44$, are all significant, $p < .02$. Particularly, PC and NC errors differ significantly in earlier sets, namely in the first, second, and fourth sets, [2]$t(36) = 3.63$, $t(36) = 3.52$, $t(36) = 2.75$, respectively, $p < .01$; the differences between PC and NC errors on other sets are not significant, $p > .1$.

We explain the correctness results as follows. Both PC and NC subjects initially rely on the S1 variables to induce the meaning of the equations and to solve the test problems. For PC subjects, this method works well, since the variables correspond directly to the first letters of the quantities they represent. For NC subjects, however, this strategy leads to incorrect solutions, given that the variables are mismatched. Eventually, NC subjects experience an epiphany; they notice the discrepancy in the S1 variable namings, and begin to use the S2 values and SP quantities for mapping. The aggregate correctness graph reflects how subjects one by one arrive at this epiphany, and eventually NC subjects too perform at ceiling. It is important to note that when subjects made errors, they typically made them across all five problems in a particular set, so singleton errors appear infrequently. Also, every subject answered at least a few problems correctly, so we cannot attribute all errors in later sets to only a small handful of subjects.

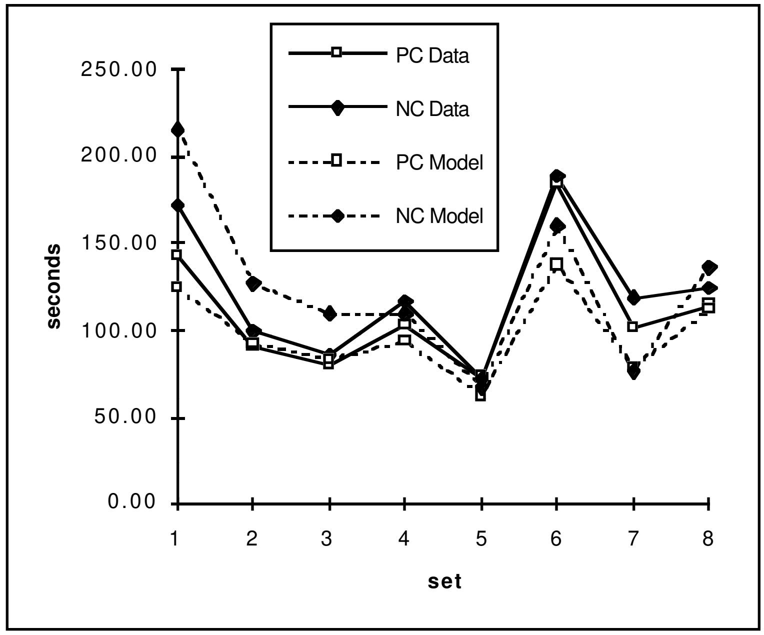


Figure 4. Average set latency for all sets. Set latency denotes the time needed to complete all five problems in the set. Solid lines show subject data, dashed lines show learning model predictions.

We can also examine the total latency per set of five problems. The solid lines in Figure 4 show the average time (in seconds) taken by subjects to complete sets one through eight. The first three points manifest the beginnings of the familiar power learning curve (Newell & Rosenbloom, 1981); recall that the problems in these sets have the same complexity, each having an equation with two variables and one operator. Subjects here were both familiarizing themselves with the system and honing their analogy skills as they pertain to these problems. For the other sets, the total latency corresponds roughly to the complexity of the problems in the set. The sixth and most complex set has the highest latency; in fact, the latency is unexpectedly high given that the complexity difference between the sixth and eighth sets is only one operator. The fifth and least complex set shows the lowest overall latency. A two-factor ANOVA shows a very significant effect of set, $F(7,288) = 48.85$, $MSE = 56193.80$, $p < .001$. The ANOVA also shows a significant effect of condition, $F(1,288) = 8.36$, $MSE = 9615.60$, $p < .01$. This effect arises primarily because NC subjects required additional time to reach their epiphanies and additional mouse clicks to analogize. There is no set-condition interaction, $F(7,288) = .72$, $MSE = 828.12$, $p > .6$.

We now focus on their performance in the eighth and final set, which, roughly speaking, reflects asymptotic performance in the task. We first examine the average latency per problem in the final set, shown as solid lines in Figure 5. Not surprisingly, the subjects took more time in the first problem, as they needed to study the sample and understand the meaning of the equation before solving the test problem. The latency curves flatten out quickly in later problems, most notably in the PC

[2] All t-tests conducted are two-tailed.

condition; subjects were essentially just checking the TP values and enter the solution equation. The problem effects are very significant, $\underline{F}(4,180) = 1.72$, $\underline{MSE} = 10876.01$, $\underline{p} < .001$, while the effects of condition, $\underline{F}(1,180) = 2.64$, $\underline{MSE} = 231.18$, and the interaction of problem and condition, $\underline{F}(4,180) = 1.72$, $\underline{MSE} = 150.92$, are not, $\underline{p} > .1$.

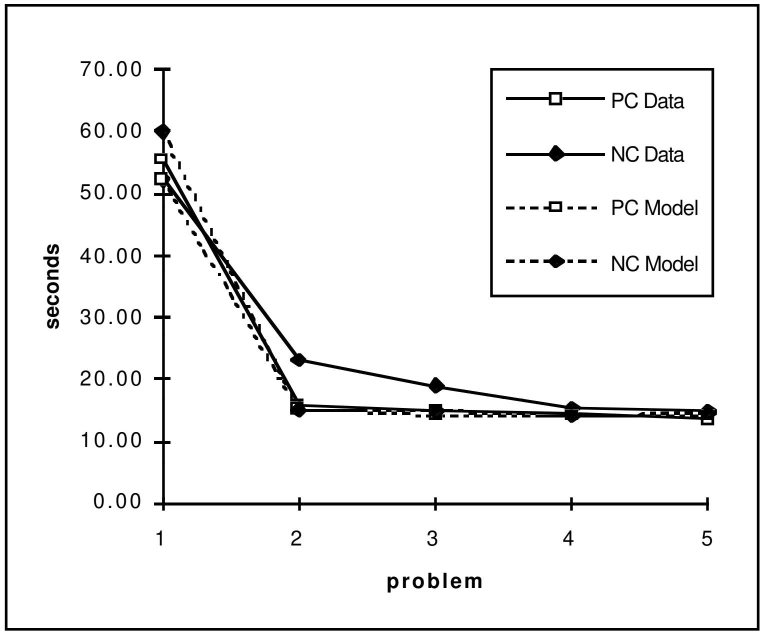


Figure 5. Average problem latency for final set. Problem latency denotes the time needed to complete a single problem. Solid lines show subject data, dashed lines show terminal model predictions.

Subject latency data illustrate approximately when and how often subjects analogized. Visual scanning data, on the other hand, give important clues to how subjects analogized. We first consider our original hypothesis that PC subjects would rely more heavily on the S1 variables, whereas NC subjects would favor the S2 values. Table 2 contains the number of references[3] per area item in the final set, across all subjects. From the table we see that PC subjects did indeed scan S1 more often than NC subjects, who exhibited more references to SP and S2. The effects of area, $\underline{F}(3,144) = 29.31$, $\underline{MSE} = 761.38$, and the area-condition interaction, $\underline{F}(3,144) = 8.21$, $\underline{MSE} = 213.25$, are very significant, $\underline{p} < .001$, while the effect of condition is not, $\underline{F}(1,144) = 1.42$, $\underline{MSE} = 37.01$, $\underline{p} > .2$. Each pair of values across conditions for SP, S1, and S2 differ significantly, $\underline{t}(36) = 2.23$, $\underline{t}(36) = 3.18$, and $\underline{t}(36) = 3.31$, $\underline{p} < .05$.

Though subjects' area references correspond with our predictions, it is interesting to note that overall subjects do not utilize the most efficient strategy. On the one hand, PC subjects could have safely ignored the S2 values if desired and looked only at the S1 variables, yet they still scanned the S2 values. On the other hand, NC subjects needed not look at the S1 variables, since the variables were always misleading; nevertheless, they clicked on the S1 variables approximately half as often as they did on the S2 values. Of course, subjects were not aware that the experimental condition would not change during the task, so these extra clicks can be viewed as checking that mapping conventions remained the same.

We can also analyze the time taken to look at objects in each area, as shown in Table 3. The reference time corresponds to elapsed time between the click on some item and a subsequent click or key press; note that duration of a click may not be an accurate measure, since some subjects may hold the button down while processing information, while others may release the button. The reference times shown in the graph reveal no significant effect of condition, $\underline{F}(1,189) = .78$, $\underline{MSE} = 0.27$, $\underline{p} > .3$, or of the interaction between condition and area, $\underline{F}(5,189) = .35$, $\underline{MSE} = 0.12$, $\underline{p} > .8$. The effect of area, however, is very significant, $\underline{F}(5,189) = 47.80$, $\underline{MSE} = 16.67$, $\underline{p} < .001$. This effect arises partly because of the high latency for the SP area; the SP times reflect the fact that the SP quantity blocks covered not only letters or words but partial sentences. Other significant differences occur in the times between S1

[3] In these and all other reference data reported, multiple references to a single block with no intervening actions were collapsed into one reference. This adjustment was made because of subjects who tended to click several times on the same block.

Table 2. Subject References per Area Item.

| | SP | S1 | S2 | TP |
|---|---|---|---|---|
| PC | 1.74 | 2.11 | 1.13 | 1.11 |
| | (1.75) | (2.30) | (1.00) | (1.00) |
| NC | 2.96 | 1.05 | 2.22 | 1.11 |
| | (3.10) | (1.05) | (2.35) | (1.00) |

Note: Values represent average number of references per item in each area. Unparenthesized values represent subject data, parenthesized values represent terminal model predictions.

Table 3. Subject Reference Times per Area Item.

| | SP | S1 | | S2 | | TP |
|---|---|---|---|---|---|---|
| | | variables | operators | values | operators | |
| PC | 2.74 | 1.77 | 0.94 | 1.57 | 0.79 | 1.38 |
| | (2.60) | (2.32) | (0.91) | (1.77) | (0.85) | (1.25) |
| NC | 2.76 | 1.76 | 0.97 | 1.59 | 1.12 | 1.45 |
| | (2.85) | (1.78) | (0.87) | (1.77) | (0.76) | (1.25) |

Note: Values represent average reference times, in seconds, per reference in each area. Unparenthesized values represent subject data, parenthesized values represent terminal model predictions.

variables and S1 operators and between S2 values and S2 operators, $\underline{t}(34) = 4.46$ and $\underline{t}(28) = 3.98$ for PC, $\underline{t}(21) = 3.11$ and $\underline{t}(35) = 3.90$ for NC, $\underline{p} < .01$. Ostensibly, subjects took only half as much time to process an operator in comparison with a variable or value.

These data give us a general sense of subjects' strategies of analogical behavior during the task, but we can characterize their behavior in more detail. We define a <u>reference path</u> as the sequence of area scans or area groupings that a subject accesses during a problem. An <u>area scan</u>, denoted by a single area such as "S1," represents a scan of all objects within the area, hitting each at least once. An <u>area grouping</u> enclosed in parentheses, such as "(S2 SP)," indicates a "bouncing" scan between two or more groups. A bouncing scan does not necessarily hit every object within the groups. Area groupings with a single area, such as "(S2)," indicate an incomplete scan of a single area bracketed by (complete) area scans; such groupings that represent only one item access are considered spurious and are omitted from our analyses. For example, assume we observe the following subject behavior: scan all S1 symbols, bounce between the S2 symbols and the SP quantities, and bounce between the TP values and the test solution block TS. We can characterize this behavior in the reference path S1 (S2 SP) (TP TS).

Table 4a shows the reference paths for each PC subject in the five final-set problems. The variability between subjects is quite striking; even with this rough characterization of analogical strategies, very few used identical strategies. Only one subject (PC1) used the most efficient strategy, scanning only S1 before proceeding to enter the solution. Approximately half the subjects began with a scan of SP, and generally continued by scanning S1. It seems that these subjects read top-down from the sample problem until they reached the relevant equation for mapping (the S1 equation for PC subjects). Also, most subjects exhibited some bouncing scans between SP and S2 after scanning S1, apparently checking the quantity equation derived from S1 by mapping S2 values to their respective SP quantities. Only one subject (PC15) utilized an obvious solve-by-value strategy, scanning SP and S2 exclusively; this fact provides evidence that subjects tend toward the more efficient (solve-by-variable) strategy in the PC condition. Finally, PC subjects showed very little dependence on the sample problem and solution after the first problem. They seem to have abstracted out the solution during the first presentation.

Table 4b lists the reference paths for NC subjects. We see a fair number of scans of the S1 equation, even though the S1 variables are misleading. Again the subjects seem to manifest the top-down reading behavior, frequently reading SP and S1 before using the variable equation S2. NC subjects also exhibited the (S2 SP) bouncing, but for the purpose of inferring the correct equation rather than for checking the variable equation. Several subjects inferred a mapping by scanning SP and S2 separately rather than bouncing between them. As in the PC condition, only one NC subject (NC16) exhibited the most efficient strategy, scanning SP and S2 once. Use of the test problem and entering of the solution are fairly similar across conditions. Subjects prefer to bounce between the test problem TP and the solution TS, as the behavior (TP TS) lightens working memory load. However, a minority of subjects showed the behavior TP TS, scanning all the test values first, then entering the entire equation into the solution block.

Table 4a. PC subject reference paths.

| | Problem 1 | Problem 2 | Problem 3 | Problem 4 | Problem 5 |
|---|---|---|---|---|---|
| PC1 | S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| PC2 | SP S1 (S2 SP S1) (TP TS) | (SP S1 TP TS) | (TP TS) | (TP TS) | (TP TS) |
| PC3 | SP (S2 SP S1) S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| PC4 | SP S1 (S2 SP S1 TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| PC5 | S1 SP S1 TP TS | TP (TP TS) | TP TS | TP TS | TP TS |
| PC6 | (SP) S1 SP (S2 SP S1 TP TS) | (S1 TP TS) | (S1 TP TS) | (TP TS) TS | (TP TS) |
| PC7 | SP S1 (S2 SP S1) (TP TS) | (TP TS) | S2 (TP TS) | (TP TS) | (TP TS) |
| PC8 | SP S1 SP S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| PC9 | SP S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| PC10 | SP S1 S2 (S2 SP) (TP TS) | SP S2 (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| PC11 | S1 SP S2 TP TS | TP TS | TP (TP TS) | TP TS (S1) | TP TS |
| PC12 | SP (S2 SP S1) TP (TP TS) S1 TS | TP TS TP TS | TP TS (TP) | TP TS | TP TS |
| PC13 | SP S1 (TP TS) | (TP TS) (S2) S1 | TP TS | TP TS | (TP TS) |
| PC14 | (S2 SP S1) (TP TS) | TP TS | TP TS | TP TS | TP TS |
| PC15 | S2 SP S2 TP TS | TP TS | TP TS | TP TS | TP TS |
| PC16 | S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| PC17 | SP S1 (S2 S1) S1 (S1 TP TS) | S1 (TP TS) | (TP TS) | S1 (TP TS) | (TP TS) |
| PC18 | SP (S2 S1) SP S2 S1 (TP TS) | (TP TS) | (TP TS) | S1 (TP TS) | (TP TS) |
| PC19 | SP S1 S2 (S2 SP) (TP TS) | (TP TS) | TP (TP TS) | (TP TS) | (TP TS) |

Table 4b. NC subject reference paths.

| | Problem 1 | Problem 2 | Problem 3 | Problem 4 | Problem 5 |
|---|---|---|---|---|---|
| NC1 | SP S2 (SP TP) TP TS | TP TS | TP TS | TP TS | TP TS |
| NC2 | (S2 S1) SP (S2 SP TP TS) | SP (S2 SP TP TS) | (S2 SP TP TS) | (TP TS) | (TP TS) |
| NC3 | S1 (TP TS) | (TP TS) (S2 SP) | TP (S1) S2 (S2 SP TP TS) | (TP TS) | (TP TS) |
| NC4 | SP S1 (S2 S1 TP TS) | S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| NC5 | SP (S2 SP) SP (S2 SP TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| NC6 | SP (S2 SP S1 TP TS) | (S1) (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| NC7 | SP (S2 SP TP TS) | (S2 SP TP TS) | (TP TS) | (S2 SP TP TS) | (TP TS) |
| NC8 | SP (S1) S2 (SP TP TS) | (S2 TP TS) | (SP TP TS) | (TP TS) | (TP TS) |
| NC9 | S2 SP S2 SP (TP TS) | (TP TS) | (TP TS) | (TP TS) TS (TP TS) | (TP TS) |
| NC10 | S2 (S2 SP TP TS) | S2 (S2 SP TP TS) | (S2 SP TP TS) | S2 (S2 SP TP TS) | (S2 SP TP TS) |
| NC11 | SP S2 (SP TP TS) | (SP TP TS) | (S2 SP TP TS) | TP TS | TP TS |
| NC12 | SP (S2 SP TP TS) TP | (TP TS) S1 TP | (TP TS) | (TP TS) | (TP TS) |
| NC13 | SP S1 (S1) S2 (S1) TP TS | TP TS | TP TS | TP TS | TP TS |
| NC14 | S1 (S2 SP) TP TS | TP TS | TP (TP TS) | TP TS | TP TS |
| NC15 | (S1) S2 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| NC16 | SP S2 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| NC17 | SP (S2 SP S1) SP (S2 S1) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| NC18 | SP S1 (S2 SP TP TS) | (S2 SP TP TS) | (TP TS) | (S2 SP TP TS) | (TP TS) |
| NC19 | SP S1 (S2 SP S1) S2 SP (TP TS) | (S2) SP (S2) (SP TP TS) S2 SP TP | (TP TS) | (TP TS) | (TP TS) |

In summary, PC subjects generally followed the solve-by-variable strategy, while NC subjects followed the solve-by-value strategy. We have also seen that subjects exhibited a great deal of variability in analogizing, and adapted their strategy of analogy to either solve-by-variable or solve-by-value according to experimental condition. In the discussion of the model for these empirical data, we will characterize subjects' behavior more formally and show how a production rule model can provide good fits to the data. Before doing so, however, we need to explain further the production rule framework in which we design our model—ACT-R.

## 4. The ACT-R Architecture

ACT-R is a unified theory of cognition based on production rules. The theory postulates two major components of knowledge, declarative and procedural. Declarative knowledge comprises individual units of information that can be recalled and explicitly reported. Procedural knowledge comprises production rules, that is, the system's knowledge of how to carry out various tasks. In this section we describe the specifics of how ACT-R represents and manipulates these two bodies of knowledge. Refer to Anderson (1993) for an extensive treatment of the theory.

In ACT-R, declarative knowledge contains individual chunks that encode single facts; for instance, one chunk might state that Harrisburg is the capital of Pennsylvania. To represent such facts, each chunk is associated with a chunk type that defines its structure. A chunk type defines several slots which, for each instance of the type, are filled in with the names of other chunks. Returning to our example, we might create a "state-capital-fact" chunk type with two slots, "state" and "capital", and create a new chunk:

```
Pennsylvania-capital  isa   state-capital-fact
    state      Pennsylvania
    capital    Harrisburg
```

Here, "Pennsylvania-capital" is the name of the chunk, the "isa" description defines its type, and the names "Pennsylvania" and "Harrisburg" represent other chunks. Each chunk has a real-valued activation that indicates its ease of retrieval. Chunks also spread activation to other chunks, namely, the chunks specified in their slots (see Anderson, Reder, & Lebière, in press). A chunk's total activation is the sum of its base-level activation, representing how often the chunk is accessed, and the spreading activation that comes from chunks in the current context.

ACT-R's production rules, like those of similar architectures, represent condition-action pairs. The condition contains one or more patterns that match chunks, where the first pattern always matches the current goal. Thus for the condition to hold, it must match both the current goal and any chunk retrievals specified. Also, the chunks matched from declarative memory must be readily retrievable, so chunks with very low activations will not match. Production actions can create new chunks and modify existing ones; the system never loses chunks, although chunks that go unretrieved for long periods of time are so difficult to retrieve that they are effectively lost. Actions can also push and pop goals from the goal stack, which is essentially a LIFO (last in, first out) queue of chunks that represent the goal currently being worked on and other goals waiting to be executed.

There are several continuous parameters that influence the performance of a production rule. The strength of the production determines how easily the production can be used; production strength is in some ways analogous to chunk activation, as strength depends on the frequency and recency of the production's firings. Productions also have two "cost" parameters, measured in seconds, that represent the immediate cost $a$ of firing the production and the estimated eventual cost $b$ of later production firings. Similarly, productions have two "success" parameters that state the probability $q$ that the production will succeed immediately (i.e., have its intended effect), and the estimated probability $r$ that firing the production will eventually lead to success.

During a simulation, ACT-R must choose which productions to fire, and must define how much time each firing takes. Conflict resolution, or the selection of an appropriate production to fire, is guided by the expected gain of firing each production. Assume that achieving the current goal has some value $\underline{G}$. The probability $\underline{P}$ of achieving the goal with a particular production is $\underline{qr}$, and the expected value of the goal is $\underline{PG}$. To calculate expected gain, we need to subtract from the expected value of the goal the expected cost of achieving the goal; for a given production, the total cost $\underline{C}$ is $\underline{a} + \underline{b}$. In summary, we can calculate the expected gain for each production as $\underline{PG} - \underline{C}$. Thus when two or more productions may fire in a given situation, we simply choose the production that maximizes our expected gain.

Once we have chosen the production, we can compute its total latency. Let $\underline{A_i}$ represent the activation of each chunk $\underline{i}$ that matches the rule condition, and let $\underline{S}$ be the strength of the rule. The latency $\underline{T}$ of the production can then be computed as:

$$T = F\sum_i e^{-f(A_i+S)} \tag{1}$$

The scaling parameters $\underline{F}$ and $\underline{f}$ default to 1. The actual details of rule latencies and conflict resolution are slightly more complicated, but this treatment will suffice for our description of the simple physics model.

Having discussed some of the low-level details of ACT-R, let us take a step back and consider what goes into a typical simulation. The simulation begins by pushing an initial goal onto the goal stack. The system then evaluates all productions that operate for that goal, and using conflict resolution selects one to fire. The production's actions may change the slot values of chunks, add new chunks, push new goals onto the stack, or pop the current goal from the stack. Execution continues with the current (top-most) goal on the goal stack, and terminates when the stack is empty or no production can fire.

Thus far we have neglected one of the central components of the architecture—learning. ACT-R can learn on several levels, but we will limit our discussion to the two learning mechanisms relevant to our task: base-level learning and strength learning. Base-level learning adjusts chunk base-level activations based on the frequency and recency of uses of the chunk. As mentioned earlier, the base-level activation is added into the total activation $\underline{A_i}$ of a chunk. Specifically, the system computes base-level activation as

$$B = \log\left(\sum_j t_j^{-d}\right) \tag{2}$$

where the $\underline{t_j}$'s represent the time since each use of the chunk, and $\underline{d}$ represents a decay constant. This equation captures the fact that repeated uses of the chunk increase its activation, thus facilitating future retrievals. It also gives a higher priority to recently used chunks. In the next section, we will see how base-level learning plays a central role in determining how many times the model needs to review the mapping formed during analogy.

The other learning mechanism relevant to this discussion is the learning of production rule strengths. There are many parallels between strength learning and base-level learning; in fact, their defining equations are almost identical:

$$S = \log\left(\sum_j t_j^{-d}\right) \tag{3}$$

Here the $\underline{t_j}$'s represent the time since each firing of the production, and $\underline{d}$ represents a constant decay parameter. Frequency and recency again play a key role, as more frequent and recent firings help to increase a production's strength. Both strength learning and base-level learning influence the timings of production firings as described by Equation 1; strength learning estimates the $\underline{S}$ value, while base-level learning influences the $\underline{A_i}$ values.[4]

Many of ACT-R's computations are noisy in order to help model individual performance differences. Our model adds noise to the computation of the expected gains of productions and the activations of chunks. The expected gain noise is added to the $\underline{PG} - \underline{C}$ calculation during conflict resolution, allowing the model to make different choices of which production to fire. The activation noise affects both the latency and selection of productions. Since the latency of a production depends on the activation of the retrieved chunks, activation noise raises or lowers the timings of rule firings. Also, since matched chunks must be readily retrievable, activation noise affects whether chunks match at all. The combination of these effects allows ACT-R to produce varying traces on different runs, such that each trace can be interpreted as a distinct simulated subject.

# 5. The Simple Physics ACT-R Model

We now know enough about the workings of ACT-R to describe the production rule model of the simple physics task. The major test for this model is its ability to predict the high- and low-level data presented earlier. Recall that these data reflect much variability in both the results of the analogy process and the actions of subjects during the process. We intend to capture as many aspects of the data as possible. Since the final model is quite large (over 120 production rules), we cannot simply present the rules as they are.[5] We will, however, give a sense of how we designed and implemented the model at a high level, and illuminate its most crucial elements. For ease of explanation, we introduce the model as two "terminal" models that represent the asymptotic behavior of subjects. That is, the terminal models capture the behavior of subjects after the subjects have optimized their strategy for a particular condition. Thus we present one terminal model that executes the solve-by-variable strategy (for the PC condition), and another that executes the solve-by-value strategy (for the NC condition). After describing the terminal models and their predictions, we will merge the two terminal models into a single "learning" model. The learning model can account for the evolution to the appropriate terminal model based on the condition (PC vs. NC) of the simulation.

## 5.1. Terminal Models

As the first step in creating the terminal models, we need to perform a task analysis to determine the overall strategy that subjects undertake in analogizing. The data support our original predictions that PC subjects focus on S1 whereas NC subjects focus on S2 and SP. The correctness data suggests that subjects begin using the solve-by-variable strategy, and eventually adjust to the appropriate strategy for their condition. The reference paths manifest, for some subjects, a tendency to read the sample problem from the top down to the area relevant for analogy. Finally, the data show that PC subjects seem to look at the S2 equation to check that the equation derived from the variables is correct. Taking all these facts into account, we can propose a general strategy for subjects upon first seeing the problem. The general strategy applies to both the PC and NC terminal models, but as we will see, is implemented slightly differently for each model.

The overall strategy is illustrated in Figure 6; the control flow for the initial problem in a set appears on the left, while the graph for subsequent problems appears on the right. First, subjects either read top-down to the relevant area, or skip directly to the area. For PC subjects, the relevant area is the S1 equation; for NC subjects, it is the S2 equation. Second, subjects use the equation to build a mental

---

[4] In addition to base-level learning, ACT-R can learn inter-associative strengths between chunks which can influence the values of $\underline{A_i}$. We do not discuss inter-associative learning here.

[5] The full model and the program needed to run the model can be found on the World Wide Web at: http://www.cs.cmu.edu/afs/cs.cmu.edu/user/dario/www/analogy/analogy.html.

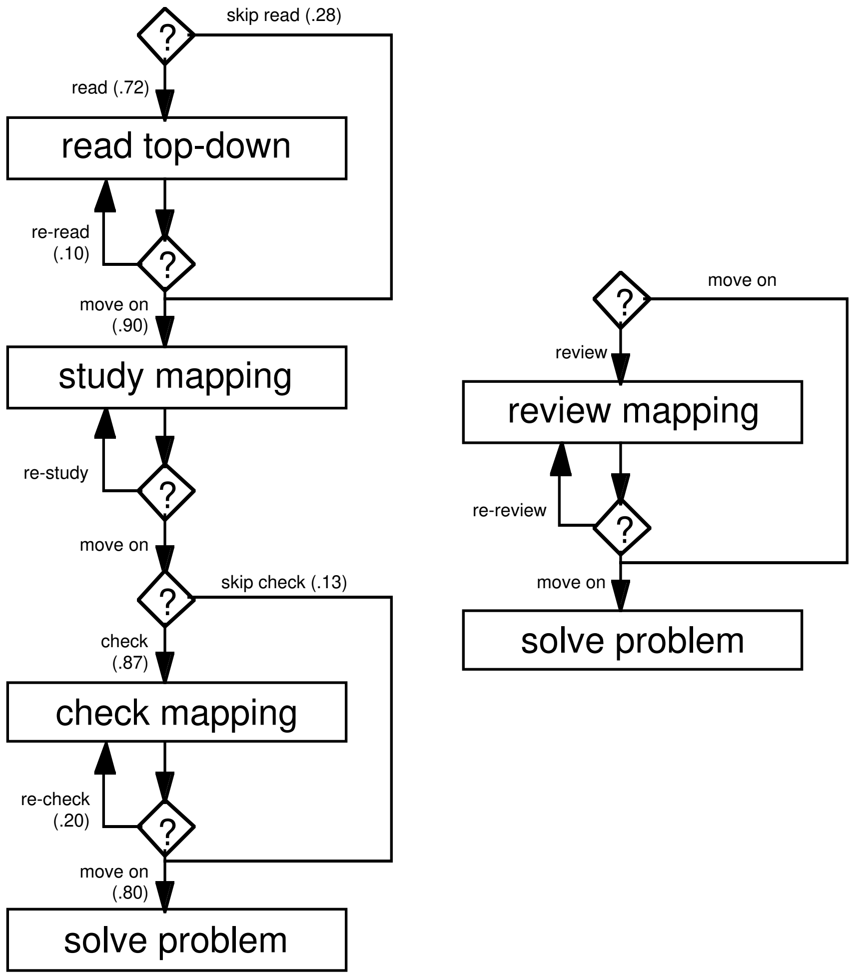


Figure 6. Main control flow graph for the simple physics model. The graph on the left illustrates the control flow for the initial problem, and the graph on the right represents the flow for subsequent problems. Percentages in parentheses indicate statically-computable frequencies for their respective paths.

representation of the mapping or "true" equation. This true equation, which we call the quantity equation, contains quantity names in place of the variables or values, such that the representation could be used to solve the test problem. For instance, for the induction problem shown in Figure 1, the equation "N*f/c" would map to the quantity equation "number*flux/current." Third, subjects either check the quantity equation or skip the check. PC subjects would check the quantity equation against the equation represented by the S2 equation, while NC subjects cannot perform any check, since the variable names are known to be misleading. Fourth, subjects solve the test problem using the quantity equation and the TP values. For subsequent problems, the quantity equation may need to be reviewed before solving the problem, and this review would constitute re-studying the equation.

The terminal models implement this general strategy in their highest level production rules, summarized in Table 5. The first stage allows two options, reading top-down to the study equation or skipping ahead. Reading top-down means reviewing material in the problem that appears before the critical material, that is, the S1 equation for PC subjects and the S2 equation for NC subjects. To decide between reading top-down and skipping ahead, the model has two productions that compete through conflict resolution. If the model opts to read top-down, this subgoal is pushed onto the goal stack. For

Table 5. Simple physics general strategy productions.

**read-top-down**
IF current stage is read-top-down
THEN set subgoal to read top-down
and move to stage study-mapping

**skip-ahead**
IF current stage is read-top-down
THEN move to stage study-mapping

**study-mapping**
IF current stage is study-mapping
THEN set subgoal to study mapping

**retrieve-mapping**
IF current stage is study-mapping
and mapping can be retrieved
THEN move to stage check-mapping

**check-mapping**
IF current stage is check-mapping
THEN set subgoal to check mapping
and move to stage solve-problem

**skip-check**
IF current stage is check-mapping
THEN move to stage solve-problem

**solve-problem**
IF current stage is solve-problem
THEN set subgoal to solve-problem
and move to stage review-mapping

**review-mapping**
IF current stage is review-mapping
THEN set subgoal to study mapping

**stop-review**
IF current stage is review-mapping
and mapping can be retrieved
THEN move to stage solve-problem

PC subjects, reading top-down involves simply reading the SP quantities and values in order. For NC subjects, reading top-down involves reading SP and S1.

The second stage requires studying the mapping, namely the quantity equation. The PC terminal model uses S1 to infer the quantity equation. The model performs a different action for each S1 symbol type (variable, constant, or operator). For each S1 variable, the model scans TP for the quantity that begins with the variable letter.[6] It then creates a knowledge chunk that stores the quantity name along with its position within the equation. For each S1 constant, the model similarly scans TP, but

[6] We could ask why subjects would scan TP for the quantity names, rather than SP. Since the SP quantity names are covered by blocks, searching them would take make more time, thus searching TP is more efficient. The reference paths for PC subjects exhibit little bouncing between S1 and SP, confirming these conclusions.

finds no quantity with the same first letter. Thus it assumes that the letter is a constant and stores the constant along with its position. For each S1 operator, the model simply stores the operator with its position. In contrast to the PC terminal model, the NC terminal model focuses on S2 and SP to infer the quantity equation. For each S2 value, the model finds the value in SP and clicks on its covered block, exposing the corresponding SP quantity. Constants and operators are immediately recognized, and all three types are again stored with their position.

In summary, the study-mapping stage uses either S1 or S2 and SP to build a representation of the quantity equation, of which each element (quantity, constant, or operator) is associated with its position in the equation. After the model studies the equation, it must decide whether to stop studying or to review. If the equation can be readily retrieved, the model advances to the checking stage and studying terminates. Otherwise, the model reviews (or re-studies) the equation. Base-level learning (see Equation 2) controls how easily the equation can be retrieved; as the equation is reviewed, its activation increases steadily, facilitating retrieval on the next attempt. Thus more reviews make it more likely that the model can recall the equation and move on.

The third stage allows the models to check the quantity equation created in the study stage. Since NC subjects can use only the S2 equation to solve the problem correctly, only PC subjects can actually check their equations. As before, the model contains two competing productions, one for checking and one for skipping the check. To check, the PC model goes through the same steps as the NC model does for studying, bouncing between S2 and SP. By definition in the PC case, the check will never fail, so after the check the model simply continues to the next stage. We will modify this check stage later when we merge the terminal models into a single learning model.

The fourth stage, where the terminal models solve the current test problem, is relatively straightforward. The model traverses the mental representation of the quantity equation, beginning with the first symbol. If the symbol is a quantity, the model searches TP for that quantity, clicks on its corresponding value, and types the value. If the symbol is an operator, the model simply types it in. Finally, when the model has traversed the entire equation, it clicks on the lower-right button to indicate that it has finished with the problem. For subsequent problems, the model enters a review stage in which the quantity equation is reviewed until it can be readily retrieved (just as in the study stage). When this review finishes, the model shifts back to the solve stage to enter the solution.

The stage descriptions above refer to many subprocesses, such as scanning the sample problem for a particular value or searching the test problem for a quantity. Each of these processes is implemented by a separate subgoal and a set of production rules. In this sense, the models are rather low-level, since they describe what the model is "looking at" during all stages of the analogy process. The leaf subprocesses are primarily visual- or motor-related, such as clicking on a block or typing in a number. The models do not actually perform these actions, but rather output a trace item indicating the time and location of each action. The trace created by the interface productions can be compared with experimental results, or even run through the experiment program in simulation mode.

At this point we have provided a broad overview of the models at the symbolic level. However, we still need to define the subsymbolic, real-valued parameters associated with chunks and productions. For chunks, base-level learning handles the setting of all base-level activations, so we need not concern ourselves with them. For productions, we used the default settings[7] for strength, $\underline{b}$, and $\underline{q}$ for all productions. We used fixed strengths in the terminal models because we assumed that productions were already at peak strength; these strengths will be learned in the learning model. This leaves only the settings for the $\underline{a}$ and $\underline{r}$ parameters.

The $\underline{a}$ parameter controls the latency for each production. We set all productions to have the standard ACT-R default latency of 50 ms, except the rules which implement the visual and motor actions. In setting the latencies for the visual and motor rules, we followed the general guideline that

---

[7] The default settings for these parameters are strength=0, $\underline{b}$=1, and $\underline{q}$=1.

"short" actions take approximately 100 ms while "long" actions take approximately 1 s. Reading a single digit or symbol, or noticing an empty space, has a latency of 100 ms. For partial sentences (i.e., the SP and TP quantities) reading takes 1 s. Long-distance mouse movements (e.g., from SP to S1 or S2) have a latency of 1 s. Typing after moving the mouse, or vice-versa, takes an additional 1 s, modeling the latency for shifting the hand between mouse and keyboard. Clicking on the "Done" button also takes 1 s. To fit our data more accurately, we stretched our guidelines in two specific cases. Short-distance mouse movements have a latency of 300 ms, and typing a character has a latency of 650 ms.[8]

The r parameter allows for adjustment of the conflict resolution between productions in cases where we design the model to have competing productions. For most productions, setting the r parameter is straightforward, merely reflecting which rule we prefer over others. However, the r parameters for certain crucial decision points need to be estimated using empirical data. We would like the parameters to be estimated such that the model makes each decision with approximately the same probability as our subjects. Given the above parameter settings, we proceeded to estimate the r parameters for the decision points shown in Figure 6. Fitting the model to the data was essentially a hill-climbing search manipulating these crucial r parameters. We performed this search manually; that is, we ran several simulations, examined the output, and adjusted the parameters accordingly. We repeated this process until the parameters produced satisfactory results.

Stating the actual values of the estimated r parameters would not be highly illustrative, since it is not obvious how they correspond to the probabilities of firing the respective productions. However, we can actually derive the expected probabilities using the r parameters and the amount of PG – C noise. Let $E_i$ be the evaluation, or PG – C value, of production i. The probability of choosing production i among all competing productions j is

$$\Pr(i) = \frac{e^{E_i/s}}{\sum_j e^{E_j/s}} \tag{4}$$

The parameter s relates to the standard deviation $\sigma$ of the PG – C noise as $s = \sigma\sqrt{6}/\pi$. This formula allows us to compute the probability of choosing each path in the general control flow graph. There is one caveat, however: The likelihood of firing a production that retrieves the studied (quantity) equation depends on the current activation of the equation, thus we cannot statically compute the probabilities for these paths. The control flow graph in Figure 6 includes probabilities for all statically computable paths.

From our presentation, one might get the impression that so many parameters could easily fit almost any data set. This is far from true. Although the two terminal models are indeed distinct models, they actually differ in only a few productions. Specifically, we vary only the specific subgoals pushed for the "read-top-down," "study-mapping," "check-mapping," and "review-mapping" productions in Table 5; all other productions, and all continuous parameters, are kept the same across models. This high degree of overlap allows us to combine the models into one learning model, as we shall see later. More importantly, though, the overlap manifests the real predictive power behind these models. It highly constrains the predictions of one terminal model given the parameters of the other. Thus, if both models provide good fits to their respective data sets, we have strong evidence to support our underlying theory.

## 5.2. Terminal Model Results

Let us now examine the predictions and behavior of the terminal models in comparison with the final set data. Given the two models, we can run repeated simulations that model individual subjects

---

[8] This typing speed is slower than continuous typing speed, but justifiably, since subjects are entering numbers and operators rather than words in text.

performing the final set of five problems. To compare the models' results with the experimental data, we ran 20 simulations for each model and examined the resulting traces.[9] Overall, the models produced good fits to many aspects of the data.

We first compare the total latencies to complete each problem, for both the models and subjects. Figure 5 includes the models' predictions (dashed lines) with the experimental data (solid lines) for the PC and NC conditions. Both models' latency curves fit their respective data nicely. Most of the analogizing is completed in the first problem, with the latencies flattening out quickly in later problems. The fit is especially good considering our rigid demand that every production's latency be set at 50 ms (with the exception of the visual and motor productions).

Regarding the visual data, Table 2 shows the predicted (parenthesized) and observed (unparenthesized) number of references per item. Again, the models capture many of the nuances of the data. For example, recall that NC subjects scanned S1 almost half as much as S2, even though the S1 variables were misleading. The NC model reproduces this phenomenon in the read top-down stage, predicting that many subjects read S1 on the scan from the top of the screen down to the relevant equation for analogizing (S2 for the NC condition). Also, the PC model explains why PC subjects referred to the S2 equation, namely, to check that their inferences from the S1 equation are correct.

Table 3 shows the empirical data and model predictions for reference times across the two experimental conditions. The high latency for reading SP quantities stems from a higher production latency for reading sentences rather than single words or symbols. The models also predict that reference latencies for S1 variables and S2 values are significantly longer than those for S1 or S2 operators. The extra time spent arises from the visual scans necessary for variables and values; for instance, after reading a value, the model must scan SP for that value. The only real discrepancy between the model predictions and the experimental data occurs in the reference latency for S1 variables in the PC condition. The data showed no significant difference between latencies for S1 variables and S2 values in either condition; however, the PC model predicts some additional processing for S1 variables, namely, storing or strengthening the mental representation of its corresponding quantity. In the NC condition, the extra latency needed for the quantity representation appears in the higher SP latency. The effect arises for SP, rather than for S2, because subjects must click on SP to find the correct quantity to store. On the whole, the models do account for much of the time spent looking at and processing the items in the various visual fields.

The reference paths for the experimental data revealed a great deal of variability in the strategies of analogy during the task. Though the models do not exhibit as much variability as the subjects, they are able to produce some of the important facets of these differences. Tables 6a and 6b illustrate the reference paths for the 20 simulation runs for each model. In both cases, we see varying amounts of studying and reviewing across simulations. Also, many of the simulations begin by reading SP, just as many subjects did. We can observe several reviews of the equation in later problems, though the reviews are less frequent than in the empirical data. Interestingly, only two simulations (MNC6 and MNC10) executed the most efficient strategy; recall that of our 38 subjects, exactly two (PC1 and NC16) did the same. Thus the models not only reproduce the latency and visual data present in the experiment, but also some of the crucial differences exhibited by subjects.

---

[9] Terminal model simulations were run with base-level learning enabled (decay rate of 0.3), strength learning disabled, an expected gain (<u>PG</u> - <u>C</u>) noise of 0.2, and an activation noise of 0.1.

Table 6a. PC terminal model reference paths.

| | Problem 1 | Problem 2 | Problem 3 | Problem 4 | Problem 5 |
|---|---|---|---|---|---|
| MPC1 | SP S1 S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC2 | S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC3 | S1 S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC4 | SP S1 S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC5 | SP S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC6 | SP S1 S1 S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC7 | S1 S1 (S2 SP) (TP TS) | S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC8 | SP S1 S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC9 | SP S1 S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC10 | SP S1 S1 (S2 SP) (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC11 | SP S1 S1 S1 S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC12 | SP S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC13 | SP S1 S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC14 | SP S1 S1 S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | S1 (TP TS) |
| MPC15 | SP S1 S1 S1 S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC16 | SP S1 S1 S1 S1 (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC17 | S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC18 | SP S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC19 | S1 S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MPC20 | SP S1 S1 S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |

Table 6b. NC terminal model reference paths.

| | Problem 1 | Problem 2 | Problem 3 | Problem 4 | Problem 5 |
|---|---|---|---|---|---|
| MNC1 | SP S1 (S2 SP) (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC2 | SP SP S1 (S2 SP) (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC3 | SP S1 S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC4 | SP S1 (S2 SP) (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC5 | (S2 SP) (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC6 | (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC7 | SP S1 S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC8 | SP S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC9 | SP S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (S2 SP) (TP TS) | (TP TS) | (S2 SP) (TP TS) |
| MNC10 | (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC11 | SP S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC12 | SP S1 (S2 SP) (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC13 | (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC14 | SP S1 (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC15 | SP S1 S1 S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC16 | SP S1 S1 S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC17 | SP S1 (S2 SP) (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC18 | SP S1 S1 (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC19 | (S2 SP) (S2 SP) (TP TS) | (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) |
| MNC20 | (S2 SP) (S2 SP) (S2 SP) (TP TS) | (TP TS) | (TP TS) | (TP TS) | (TP TS) |

## 5.3. Learning Model

We have seen how the PC and NC terminal models operate to fit the final set data. However, there is still the question of how these terminal models might evolve. We would like for the models to arise from a single initial model which would itself evolve into one of the terminal models. That is, we desire a learning model that runs on the entire task (i.e., all eight sets), and depending on the problems presented to the model, would shift to either the solve-by-variable strategy (the PC terminal model) or the solve-by-value strategy (the NC terminal model).

Since the terminal models were designed with the learning model in mind, merging them is fairly straightforward. The learning model retains a single chunk that describes the current strategy to utilize, either to solve by variable or to solve by value. The few productions which differ between the terminal models are modified to retrieve this chunk and determine the current strategy. If the strategy dictates to solve by variable, the production pushes the subgoal(s) corresponding to the PC terminal model; otherwise, it pushes the subgoal(s) corresponding to the NC terminal model. Initially, the strategy chunk is set to solve by variable. As the model simulates the task, it eventually checks the S2 values to ensure that they correspond to the S1 variables. In the PC condition, the variables and values always correspond, so the learning model continues to use the solve-by-variable strategy. In the NC condition, the check fails, and the model decides whether or not to consider the check; this choice models how some subjects perform the sequence of clicks for a check but do not notice the discrepancy in variable namings. When the check is both performed and considered, the model switches to the solve-by-value strategy. The learning model thus adjusts to the condition in which the simulation is run, shifting to either the solve-by-variable or solve-by-value strategy as appropriate.

Because the terminal models assume that the relevant task skills are already present and highly tuned, they ignore the strengthening of productions during simulation. The learning model, on the other hand, should account for the tuning of appropriate productions. We thus activate strength learning in the learning model simulations, as defined by Equation 3. This learning increases the production strength for frequently-fired productions, resulting in lower firing latencies. The model also incorporates the assumption that all subjects read top-down for the first problem in the first set of the task.

## 5.4. Learning Model Results

We now present the results of 20 learning model simulations of the entire task in each condition.[10] The learning model correctness results are included in Figure 3. Comparing the model's predictions (dashed lines) with the empirical data (solid lines), we see that the model captures several aspects of subjects' behavior. The PC model, like PC subjects, performs at ceiling. This behavior arises from the fact that the learning model always tries initially to solve by variable, which is the correct approach in the PC condition. In the NC condition, the model requires some number of trials before it notices the discrepancy between the sample solution variables and values; recall that subjects exhibited similar epiphanies. The NC error curve then essentially maps out when the model simulations reached their epiphanies.

Figure 4 shows the average set latencies for the model (dashed lines) and the experiment (solid lines). We see that the learning model nicely reproduces the shape of our empirical data; namely, the sets involving more complex problems require more time to complete. In the three initial sets (which have identical complexities), we observe a steady decrease in latency, due primarily to the strengthening of relevant productions. Also, the model requires slightly more time in the NC condition

---

[10] Learning model simulations were run with base-level learning enabled (decay rate of 0.3), strength learning enabled (decay rate of 0.5), an expected gain (PG - C) noise of 0.2, and an activation noise of 0.1.

than in the PC condition, since the NC times include an overhead for epiphanies that occur during the sets.

Generally the learning model captures the important aspects of the data. Nonetheless, we can point to two incompletenesses of the model. First, the model does not account for the acquisition of productions specific to the task. Presumably, many of the rules (for scanning, typing, etc.) are in place at the onset of the task. However, we could argue that the task invokes the creation of certain specialized productions that act as schemata for problem solving (see Anderson & Thompson, 1989). This crucial issue is beyond the scope of this model; much further analysis of the task would be required to determine how and when such rules are created. Second, there may be other strategies which subjects utilize when first attempting the task. Our model assumes that subjects begin with the solve-by-variable strategy, and sometimes (i.e., in the NC condition) shift to the solve-by-value strategy. It is possible, however, that subjects initially use a different strategy, perhaps a hybrid of solve-by-variable and solve-by-value, perhaps an altogether different approach. Again, further work is needed to look for systematic strategies in the early stages of the task that evolve to the terminal strategies.

## 6. Generalizing the Simple Physics Model: The People-Piece Model

We have seen how a production rule model can capture many aspects of the simple physics task. Nevertheless, we might reasonably question how useful the model is when considering the broader picture of problem solving by analogy. Can we generalize the simple physics model to make predictions about other analogical tasks? More generally, can production rule models for specific analogical tasks assist in modeling similar tasks? The answer is yes, and centers on our earlier claim that models for similar analogical tasks share common production rules that implement analogy. We now provide evidence for this claim by presenting a rule model for a similar analogical task, where the model uses specific productions taken from the simple physics model.

Our comparison task is the Sternberg (1977) "people-piece" analogy task. Sternberg presented subjects with picture analogies of the form A:B::C:D. The elements of the analogy were drawings of people varying four binary attributes: sex (male-female), color (blue-red), height (short-tall), and girth (fat-thin). Sternberg asked subjects to respond whether the analogy was true or false. His data[11] show that subject response latencies increased as elements A and B differ by more attributes; the data show a similar effect for differences between A and C. The solid lines in Figure 7 graph Sternberg's results for differences of one and two attributes for both the A-B and A-C pairs. The data thus suggest that subjects consider both A-B and A-C mappings, and take more time as these mappings become more complex.

We can further characterize subject behavior in the people-piece task by considering the process models of Sternberg (1977) and Grudin (1980). Sternberg provided four distinct process models for solving people-piece analogies. Though the models differ in specifics, each requires encoding of the elements, mapping between source and target elements, and application of the mapping to derive a solution. Sternberg proposed that subjects map both A to B and A to C during the process. Grudin suggested a change to Sternberg's model: Subjects map <u>either</u> A to B <u>or</u> A to C, but do not infer both mappings. We combine the ideas of both researchers into our process model for the people-piece task. Our process model first decides to execute one of two strategies, one using the A-B mapping, the other using the A-C mapping. If the model chooses to use the A-B strategy, it runs through the following sequence of steps: encode A and B, infer the A-B mapping, encode C and D, apply the mapping to C to create D′, compare D and D′ for equality, and respond. If the model chooses the A-C strategy, it follows the same steps with B and C switched.

---

[11] Sternberg (1977) also presented data for verbal and geometric analogies, including effects of cueing and degenerate analogies. We do not address these data here.

We can frame this process model within the general strategy for the simple physics task (Figure 6). The study-mapping stage of the simple physics model handles the encoding of the sample and the formation of the mapping. Similarly for the people-piece model, we bundle the encoding and mapping of A and B (or C) into a study-mapping stage. The solve-problem stage of the simple physics model corresponds to the application of the mapping. For the people-piece model, the solve-problem stage similarly involves the application of the A-B or A-C mapping. The read-top-down and check-equation stages in the simple physics model have no analog in the people-piece model, so they are omitted. Thus, the two models share the study-mapping and solve-problem stages.

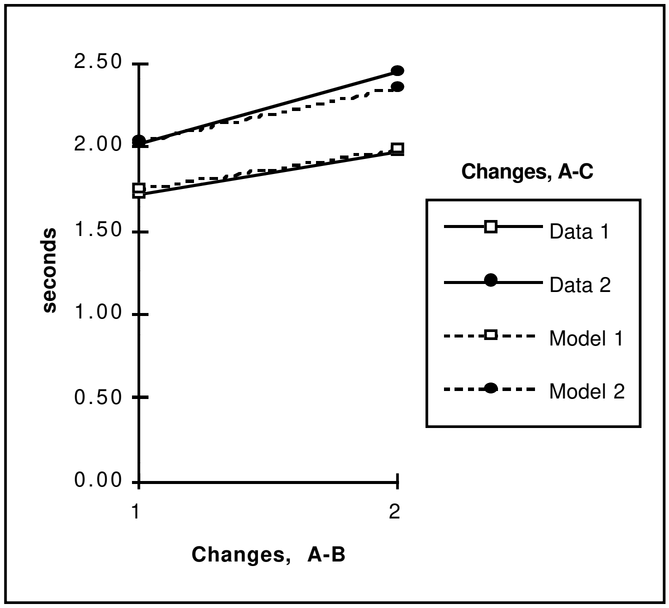


Figure 7. People-piece data and model predictions. The solid lines show the empirical data from Sternberg (1977) for one and two attribute changes from A to B and A to C. The dashed lines show the people-piece model predictions.

The production rule specification for the people-piece process model utilizes this overlap between models. The common stages allow us to borrow productions from the simple physics model and insert them directly into the people-piece model. Specifically, the people-piece model uses the productions in the simple physics model that implement the study-mapping and solve-problem stages; these productions are the "study-mapping," "retrieve-mapping," and "solve-problem" rules in Table 5. We also insert productions that skip over the read-top-down and check-equation stages, since they are not applicable here. Other productions are then added to implement subprocesses specific to the people-piece task. Before analogizing, the rule model decides using competing productions whether to use the A-B or A-C strategy. It encodes people-piece drawings as chunks with four slots for each of the binary attributes. The model handles mapping by examining the four attributes sequentially and creating a linked-list of attribute mapping chunks. Each mapping chunk indicates a single attribute which changes from source to target. To apply the mapping, the model first copies the source person chunk into a target person chunk. It then runs through the list of mapping chunks and changes each mapped attribute. The model finally compares the target chunk to the solution chunk and responds true or false.

We set production parameters for the people-piece model by following the basic assumptions of the simple physics model. Productions which were copied from the simple physics model maintained the same parameter settings. For other productions, latencies (a parameters) were set to 50 ms. The only crucial r parameters are those for the competing productions that choose the A-B or A-C strategy; in that case, the r parameters were given identical values so that the model would choose each with equal likelihood. We also increased the productions' strength parameters to model repeated trials, since subjects solved over 1000 people-piece analogies in the original Sternberg task; we estimated these strengths to have a value of 1.5.

Figure 7 includes the predictions of the model (dashed lines) along with Sternberg's empirical data (solid lines). The model provides an excellent fit to the data. The model predicts increasing latencies as the number of differing attributes between A and B (or C) increases. This behavior arises from both the mapping and application stages. In the mapping stage, different attributes call for several extra productions that handle the creation of a new mapping chunk. In the application stage, each attribute that must be mapped requires several additional production firings to change values in the target person chunk. Because the model chooses the A-B and A-C strategies with equal likelihood, the effects are the same for attribute changes in the A-B and A-C pairs.

We have thus shown that it is possible to transfer the implementations of analogical skills between models for similar analogical tasks. The simple physics and people-piece models share several analogy productions, and also incorporate domain-specific knowledge relevant to the particular task. Both models also illustrate how production rule models can account for multiple strategies of analogy: The simple physics model can use either a solve-by-variable or solve-by-value strategy, and the people-piece model can choose between the A-B and A-C strategies. Presumably, if either the A-B or A-C strategy proved more effective (which is not the case in Sternberg's experiment), the people-piece model could incorporate learning of the appropriate strategy similar to the learning in the simple physics model.

# 7. General Discussion

## 7.1. Related Theories of Analogy

Our approach to analogy relates to a number of theories in the literature. The work of Holyoak and Thagard (1989b) perhaps most resembles ours, in that they consider analogy within the broader context of general problem solving. In fact, their PI (processes of induction) system and ACT-R share several important traits. PI uses condition-action rules to act upon a store of memory elements, called "concepts." Like ACT-R, the system incorporates spreading activation from the current context, but it also spreads activation to rules related to the context. The rule activation reduces the number of possible productions that can fire, since the system only considers active rules in the matching stage. Also, in contrast to ACT-R's sequential operation, PI rules can fire in parallel.

PI models analogical mapping and schema formation in seven distinct stages. First, the system modifies the sample problem and sample solution concepts to include a relation between them. Second, the system attempts to solve the test problem using rules. Third, given that the current rule set cannot solve the test problem, the relevant concepts in the current context must provide sufficient activation to trigger analogical mapping. Fourth, the system determines a partial mapping which relates concepts to other concepts to which they have spread activation. Fifth, the system completes the mapping using the constraints imposed by the relations in the partial mapping. Sixth, PI uses this mapping to perform analogous actions on the test problem. Finally, assuming the analogy has led to a successful solution, the system builds and stores a schema in which appropriate values are variabilized.

In these stages PI makes a clear distinction between rule matching and analogical mapping. When solving a problem, the system first attempts to use existing rules. If the current rule set cannot solve the test problem, the system begins the process of analogical mapping. Regarding this distinction, Holyoak and Thagard (1989b) claimed that "analogical mapping is more cognitively demanding than rule matching." In the models we have presented, analogical mapping involves the matching and firing of many production rules to execute the mapping. Thus mapping actually comprises a number of rule firings. This illustrates one way in which analogy may be "more cognitively demanding" than rule matching.

Gentner's (1983) original structure-mapping theory seems very amenable to a production system framework. In fact, it seems likely that one could actually implement the theory using only basic production rules. The theory represents facts in a simple predicate language comprising relations and objects. Gentner proposed three principles for inferring mappings between a source and target analog: discard attributes of objects, preserve relations between objects, and prefer systems of relations (i.e., systematicity; see also Gentner & Toupin, 1986). To build a production rule model of the theory, we would need to represent the relevant concepts as hierarchical structures of declarative knowledge units. The model would contain production rules that can traverse such a structure and build up a representation of a mapping. These rules would implement Gentner's three principles and would produce a structure mapping from source to target. Such a production rule model would decompose analogy into a number of discrete steps rather than treating it as a single act. As such it would be possible to predict observables like the sequence of visual scanning actions that subjects would make in performing the analogy. Decomposing the overall process into a number of rules would also enable us to make predictions about how the analogy process would improve with practice. Such predictions could be made by ACT-R's strength learning mechanism, as we have seen with respect to the simple physics learning model in Figure 4.

Another analogy mechanism, the Incremental Analogy Machine (IAM), centers on an incremental mapping strategy from source to target (Keane et al., 1994). IAM addresses working memory constraints by incrementally mapping parts of the source to parts of the target. The algorithm operates in six steps. First, a seed group, or group of predicates with the most higher-order connectivity, is selected from the base domain. Second, IAM chooses a seed match for an element of this group, based on various theoretical constraints. Third, the mechanism finds a one-to-one set of matches between the source and target of the seed match. Fourth, IAM uses these matches to infer transfer relations between unmapped relations, forming a complete mapping. Fifth, if the mapping is suboptimal (by some predefined metric), the system attempts to map an alternative seed match. Sixth, IAM moves on to map other groups.

Keane et al. (1994) discussed several experiments that provide convincing empirical support for their theory. We believe that such support could be strengthened in two ways. The addition of a dedicated limited-capacity working memory would help in solidifying the argument that IAM addresses working memory constraints; Keane et al. mentioned this idea as a viable option. Also, they based their empirical support on comparisons between subject latencies and the number of alternative mappings computed by the theory. Though the comparison works well in this context, this metric may not generalize well to other theories; in fact, even within the paper, the authors noted a "caveat" when applying the metric to the Structure Mapping Engine (Falkenhainer et al., 1989). Production system architectures can help address such problems. Since most architectures have a limited-capacity memory built into the system, any analogy model developed in the architecture must necessarily deal with working memory constraints. Also, rule models produce traces that can be directly compared with empirical evidence, avoiding the problem of determining suitable metrics for comparison.

Like the above mechanisms, Holyoak and Thagard's (1989a) Analogical Constraint Mapping Engine (ACME) considers analogy as the creation of mappings between structures, but takes a slightly different approach. ACME builds up a network of mapping propositions with excitatory and inhibitory connections, and uses constraint-satisfaction methods to arrive at a solution. Although some work has linked network-based systems to rule-based systems (e.g., Cho, Rosenbloom, & Dolan, 1991; Lebière & Anderson, 1993), fitting ACME into a production rule framework seems awkward and certainly non-trivial. We can discuss, however, how our framework addresses some of the same issues as ACME. Holyoak and Thagard emphasized three constraints that affect analogical behavior: structural, semantic, and pragmatic. Structural constraints state that the system can form correspondences between the objects and relations of the source and target analogs; the mappings are usually one-to-one. Semantic constraints add a similarity metric between objects and predicates that allow the system to prefer certain mappings over others. Finally, pragmatic constraints guarantee that analogy is relevant to the current context, that is, that the current goal directs the mapping process.

Production rule models of analogy can implement each of these constraints. Structural constraints arise from the particular rules that implement the mapping process. Modelers can design these productions so that relations map to relations and objects to objects; for instance, the simple physics model contains rules that find one-to-one correspondences between the symbols in the solution equation and the schematic quantity equation. Semantic constraints fall out of the spreading activation process for facts in declarative memory. During rule matching, the source facts being considered help to activate similar target facts, thus facilitating retrieval of similar target objects and relations. Pragmatic constraints arise from the fact that we handle analogy within a general problem-solving framework. Within such a framework, all actions are purpose-directed, taking into account the current goal and context. Thus production rule systems are well-suited to address all three constraints.

It is important to note that all of the above theories use a single, deterministic mechanism to implement analogy. Therefore, as discussed earlier, these theories cannot predict variability in analogical strategies that does not arise from representational differences, nor can they predict adaptation of these strategies during learning. In addition, the theories as presented (with the exception of PI) are not incorporated into a more general processing system, though the current trend seems to indicate a move toward this incorporation (e.g., Gentner, 1989). Production system models of analogy do not have these limitations, as evidenced in the simple physics and people-piece models.

## 7.2. Other Related Work

Ross's work (1989) discusses the effects of superficial similarities when analogizing. Ross showed that superficial similarities between the sample and test problems can affect the quality of solution by analogy. In several studies, he manipulated the similarity between corresponding objects in probability problems presented to subjects. To summarize his results, performance on the problems increased with high similarity between corresponding objects, and decreased with high similarity between non-corresponding objects. Our simple physics task reflects similar subject behavior. The variable namings used in each condition hint at superficial similarities between variables and quantities; for instance, subjects are more likely to associate the variable m with the quantity mass rather than area, though there is no structural justification for such reasoning. In the PC condition, we see high initial performance, since the similarity between corresponding objects is high. In the NC condition, we see poor initial performance because of superficial similarities between non-corresponding objects.

The results of the simple physics study, then, corroborate the results of Ross (1989). If we look more closely at the cause of high versus poor performance, though, we can see subtle differences in the origin of subjects' behavior. In the simple physics task, the misleading variable names almost certainly lead to a mistaken schematic structure, in the form of an erroneous quantity equation. Subjects seem to map variables to quantities incorrectly, thus remembering the schematic equation with the quantities in the wrong places. Errors in Ross's probability problems arose from superficial similarities between the sample and test problems, rather than between the sample problem and sample solution. Thus superficial similarities can affect both the mapping process that produces a schema (our research) and the use of the schema (Ross's research).

Anderson (1993) has discussed the creation of schematic ACT-R productions as a result of analogy (see also Anderson & Thompson, 1989). ACT-R's so-called "analogy mechanism" builds new productions that map a subgoal to another subgoal using an example from declarative memory. The analogy mechanism and our analogical framework both involve what might be called analogy, but they operate on very different levels. The analogy mechanism in ACT-R acts at a lower level of cognition, creating single rules based on individual facts stored in declarative memory. Our framework describes how production systems can account for high-level analogical reasoning, mapping between arbitrarily complex cognitive structures and possibly involving many other skills (e.g., visual processing). In fact, the framework says nothing about the specific mechanisms for production compilation, so the ACT-R analogy mechanism may very well fit into many models of high-level analogy. We avoid using the analogy mechanism in the simple physics and people-piece models primarily so that the models can more easily generalize to other production systems. However, it may be the case that the analogy mechanism could be incorporated into these and other models.

## 7.3. Conclusions

We believe our results have shown that production system architectures are well-suited for modeling analogical behavior. Production systems can account for the variability and adaptation of analogical strategies. The systems also allow direct comparison of model predictions to both high- and low-level empirical data. Furthermore, production systems offer a unified approach to cognition and can thus incorporate analogy within the broader scope of problem solving. We have also demonstrated that comparison of model predictions to high- and low-level observable data is an effective method of evaluating theories of analogy. Past empirical support for many theories of analogy has concentrated on relating high-level data to some aspect of the analogy mechanism using a predefined metric. We have shown that it is also possible to make predictions about the time and identity of observable low-level events (visual scans, key presses, etc.).

## References


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

Anderson, J. R., Corbett, A. T., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. The Journal of the Learning Sciences, 4, 167-207.

Anderson, J. R., & Matessa, M. P. (in press). A production system theory of serial memory. Psychological Review.

Anderson, J. R., Matessa, M., & Douglass, S. (1995). The ACT-R theory and visual attention. In Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Reder, L. M., & Lebière, C. (in press). Working memory: Activation limitations on retrieval. Cognitive Psychology.

Anderson, J. R., & Thompson, R. (1989). Use of analogy in a production system architecture. In S. Vosniadou & A. Ortony (Eds.), Similarity and analogical reasoning. Cambridge, England: Cambridge University Press.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. Cognitive Science, 13, 145-182.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. Cognitive Science, 5, 121-152.

Cho, B., Rosenbloom, P. S., & Dolan, C. P. (1991). Neuro-Soar: A neural-network architecture for goal-oriented behavior. In Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum Associates.

Duncker, K. (1945). On problem solving. Psychological Monographs, 58, 270.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). Structure-mapping engine. Artificial Intelligence, 41, 1-63.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. Cognitive Science, 7, 155-170.

Gentner, D. (1989). The Mechanisms of Analogical Learning. In S. Vosniadou & A. Ortony (Eds.), Similarity and analogical reasoning. Cambridge, England: Cambridge University Press.

Gentner, D., & Toupin, C. (1986). Systematicity and surface similarity in the development of analogy. Cognitive Science, 10, 277-300.

Gick, M. L., & Holyoak, K. J. (1980). Analogical problem solving. Cognitive Psychology, 12, 306-355.

Grudin, J. (1980). Processes of verbal analogy solution. Journal of Experimental Psychology: Human Perception and Performance, 6, 67-74.

Halliday, D., & Resnick, R. (1988). Fundamentals of physics. New York, NY: John Wiley & Sons.

Hinsley, D. A., Hayes, J. R., & Simon, H. A. (1977) From words to equations: Meaning and representation in algebra word problems. In M. A. Just & P. A. Carpenter (Eds.), Cognitive processes in comprehension. Hillsdale, NJ: Erlbaum.

Holyoak, K. J., & Thagard, P. R. (1989a). Analogical mapping by constraint satisfaction. Cognitive Science, 13, 295-355.

Holyoak, K. J., & Thagard, P. R. (1989b). A computational model of analogical problem solving. In S. Vosniadou & A. Ortony (Eds.), Similarity and analogical reasoning. Cambridge, England: Cambridge University Press.

John, B. E., Vera, A. H., & Newell, A. (1991). Toward real-time GOMS (Tech. Rep. No. CMU-CS-90-195). Pittsburgh, PA: Carnegie Mellon University, Department of Computer Science.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. Psychological Review, 99, 122-149.

Keane, M. T., Ledgeway, T., & Duff, S. (1994). Constraints on analogical mapping: A comparison of three models. Cognitive Science, 18, 387-438.

Kieras, D. E., & Meyer, D. E. (1995a). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction (EPIC Tech. Rep. No. 5, TR-95/ONR-EPIC-5). Ann Arbor: University of Michigan, Department of Electrical Engineering and Computer Science.

Kieras, D. E., & Wood, S. D., & Meyer, D. E. (1995b). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task (EPIC Tech. Rep. No. 4, TR-95/ONR-EPIC-4). Ann Arbor: University of Michigan, Department of Electrical Engineering and Computer Science.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. Artificial Intelligence, 33, 1-64.

Laird, J. E., & Rosenbloom, P. S. (1990). Integrating execution, planning, and learning in Soar for external environments. In Proceedings of the Eighth National Conference on Artificial Intelligence. Menlo Park, CA: AAAI Press.

Lebière, C., & Anderson, J. R. (1993). A connectionist implementation of the ACT-R production system. In Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Erlbaum.

Lebière, C., Anderson, J. R., & Reder, L. M. (1994). Error modeling in the ACT-R production system. In Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Erlbaum.

Lehman, J. F., Lewis, R. L., & Newell, A. (1991). Integrating knowledge sources in language comprehension. In Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Erlbaum.

Lovett, M. C., & Anderson, J. R. (1995). Making heads or tails out of selecting problem-solving strategies. In Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A. (1990). Unified theories of Cognition. Cambridge, MA: Harvard University Press.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), Cognitive skills and their acquisition. Hillsdale, NJ: Lawrence Erlbaum Associates.

Novick, L. R. (1988). Analogical transfer, problem similarity, and expertise. Journal of Experimental Psychology: Learning, Memory, and Cognition, 14, 510-520.

Novick, L. R., & Holyoak, K.J. (1991). Mathematical problem solving by analogy. Journal of Experimental Psychology: Learning, Memory, and Cognition, 17, 398-415.

Payne, J. W. (1976). Task complexity and contingent processing in decision making: An information search and protocol analysis. Organizational Behavior and Human Performance, 16, 366-387.

Polk, T. A., & Newell, A. (1995). Deduction as verbal reasoning. Psychological Review, 102, 533-566.

Reeves, L. M., & Weisberg, R. W. (1994). The role of content and abstract information in analogical transfer. Psychological Bulletin, 115, 381-400.

Ross, B. H. (1989). Distinguishing types of superficial similarities: Different effects on the access and use of earlier problems. Journal of Experimental Psychology: Learning, Memory, and Cognition, 15, 456-468.

Ross, B. H., & Kennedy, P. T. (1990). Generalizing from the use of earlier examples in problem solving. Journal of Experimental Psychology: Learning, Memory, and Cognition, 16, 42-55.

Schunn, C. D., & Klahr, D. (in press). Stances: Production systems. In W. Bechtel & G. Graham (Eds.), A companion to cognitive science. Oxford: Basil Blackwell.

Singley, M. K., & Anderson, J. R. (1989). The transfer of cognitive skill. Cambridge, MA: Harvard University Press.

Spellman, B. A., & Holyoak, K. J. (1993). An inhibitory mechanism for goal-directed analogical mapping. In Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum Associates.

Sternberg, R. J. (1977). Component processes in analogical reasoning. Psychological Review, 84, 353-378.

Sternberg, R. J., & Gardner, M. K. (1983). Unities in inductive reasoning. Journal of Experimental Psychology: General, 112, 80-116.

VanLehn, K., & Jones, R. M. (1993) Better learners use analogical problem solving sparingly. In Machine Learning: Proceedings of the Tenth International Conference. San Mateo, CA: Morgan Kaufman.

Whitely, S. E., & Barnes, G. M. (1979). The implications of processing event sequences for theories of analogical reasoning. Memory & Cognition, 7, 323-331.

Zhu, X., & Simon, H. A. (1987). Learning mathematics from examples and by doing. Cognition and Instruction, 4, 137-166.

## Appendix: Simple Physics Sample Problems

This appendix lists the sample problems and solutions for each of the eight topics presented in the simple physics PC condition. The solutions in the NC condition were identical except for the variable names in the first equation; refer to Table 1 for the NC equations used. The test problems used were similar to these sample problems, but the values were changed and the ordering of the quantity lines varied.

### Set 1

Sample Problem:
Two spaced metal plates form a capacitor.
The area of each plate is 9.
The spacing between the plates is 7.
What is the capacitance C?

Solution:
$C = A / s$
$= 9 / 7$

### Set 2

Sample Problem:
A sheet of glass is put between a hot stone and a cooler one.
The thickness of the glass is 8.
The conductivity of the glass is 3.
What is the thermal resistance R of the glass?

Solution:
$R = T / c$
$= 8 / 3$

### Set 3

Sample Problem:
A rod traveling near the speed of light contracts.
The rod's proper length is 5.
The Lorentz factor is 9.
What is the rod's length L' at that speed?

Solution:
$L' = L / f$
$= 5 / 9$

### Set 4

Sample Problem:
A block is fastened to a spring attached to a wall.
The spring constant is 4.
The distance from equilibrium is 7.
What is the force F exerted by the spring on the block?

Solution:
$F = - c * d$
$= - 4 * 7$

### Set 5

Sample Problem:
A beam of light is traveling through some medium.
The index of refraction is 5.
What is the velocity of light v in the medium?

Solution:
v = c / i
= c / 5

## Set 6

Sample Problem:
Light shines through a narrow slit.
Consider minimum number 2,
which falls at angle 5.
The wavelength of the light is 7.
What is width s of the slit?

Solution:
s = n * w / S a
= 2 * 7 / S 5

## Set 7

Sample Problem:
A cylinder contains 1 mole of oxygen under pressure.
The pressure of the gas is 4.
The temperature of the gas is 7.
What is the volume V of the cylinder?

Solution:
V = R * T / p
= R * 7 / 4

## Set 8

Sample Problem:
A coiled wire forms an inductor.
The flux through the inductor is 4.
The number of windings is 9.
The current through the wire is 2.
What is the inductance L of the inductor?

Solution:
L = N * f / c
= 9 * 4 / 2