CHAPTER 7:

MULTIPLE SPACE SEARCH IN SCIENTIFIC DISCOVERY

Christian D. Schunn & David Klahr

In D. Klahr (2000) (Ed.), *Exploring science: The cognition and development of discovery processes*.

Cambridge, MA: MIT Press

> I argued that it was important not to place too much reliance on any single piece of experimental evidence. It might turn out to be misleading .... . Jim (Watson) was a little more brash, stating that no good model ever accounted for all the facts, since some data was bound to be misleading if not plain wrong. A theory that did fit all the data would have been "carpentered" to do this and would thus be open to suspicion. (Crick, 1988. pp. 59-60, emphasis in original.)

Most practicing scientists would agree with Jim Watson's claim and acknowledge that, for any theory, there always remain the annoying anomalies, the data that would require undesirable carpentering of the theory in order to be brought into its scope. Indeed, both scientists and nonscientists have a wide variety of potential responses to apparent anomalies. These range from outright rejection of the data, to explaining it away, to fully accepting it and revising their theory (Chinn & Brewer, 1992, 1998). Although our studies -- based upon BigTrak and its isomorphs - - suggest that sticking to hypotheses in the face of inconsistencies is not unusual, we wanted to explore further the effects of repeated cycles of consistent and inconsistent data. Our challenge was to craft a discovery context that causes people to tentatively abandon lines of inquiry and that forces them to reconsider the very features of the data that they are attending to.

This was one of the motivations for the study reported in this chapter. Although the BT microworlds are much more complex than the types of problems typically used by cognitive psychologists interested in problem-solving and reasoning, they represent only a small step on the continuum of complexity from "typical" cognitive studies to the real-world complexity of scientific discovery. Recall that both adults and children tended to spend no more than about 30 minutes on the task whether or not they discovered the correct rule. In contrast, real scientists work on their scientific discoveries for years — and with much lower success rates. What — in addition to the acquisition of vast amounts of domain-specific knowledge — accounts for this difference in difficulty? Is it simply a matter of more of the same activities or are qualitatively different strategies and heuristics involved?

To investigate these questions, we designed a new discovery microworld that was considerably more complex than BigTrak. We "scaled up" the discovery task from one that required, in effect, a single insight (i.e., the shift from a Counter frame to a Selector frame) during about 30 minutes of problem solving to one that required a series of interrelated discoveries over more than an hour or so. We were particularly interested in the extent to which analysis of participants' behavior in a more complex microworld would provide evidence of different strategies and processes than we had already identified. That is, the new task had the potential for revealing new heuristics and strategies that people use to deal with increased task complexity.

Several specific objectives motivated the design of this new discovery microworld. First, we wanted to make it sufficiently interesting to sustain participants' interest for more than an hour of focused problem solving. Second, in order to make the task more realistic, we wanted to provide an external memory for participants’ experiments and their outcomes. Just as scientists can analyze and re-analyze outcomes from multiple experiments, we wanted our participants to be able to do so without having to rely exclusively on their own memories, as they did with BigTrak. Third, as in the BT microworld, we wanted prior knowledge to render some hypotheses more plausible than others. Fourth, we wanted continue to use a task requiring a complex mapping between the experiment space and the hypothesis space.

THE MILK TRUCK MICROWORLD

The task we designed to meet these constraints is called MilkTruck[1] . As with BT, participants conducted experiments by using a computer microworld in which they controlled the actions of an animated device. Here too, their goal was to discover the effect of a complex mystery function, but the MilkTruck cover story is different and the interface has many new features. Because it is difficult to understand our analysis of participants' behavior on the task without fully understanding the MilkTruck microworld, we will describe it in detail.

An overview of the MilkTruck environment

Participants were introduced to the MilkTruck task with a cover story. They were told that they were milk truck drivers and that their company had received a fleet of computerized milk trucks. Unfortunately, the instructions accompanying the new trucks are all in a foreign language unknown to the participants. Thus, they must figure out on their own how to control the milk trucks. However, all but one of the commands is very straightforward, and the participants were given step-by-step instructions on how to use those commands.

Figure 7.1 depicts the full display that participants encountered as they worked with MilkTruck. The milk truck is programmed using a keypad (upper left of Figure 7.1). Each program step is entered by clicking on an "action" key (located at the top of the keypad) followed by a number key (located in the middle of the keypad). The action keys control the five different actions that the milk truck "driver" can perform. (It turns out that the sequence in which these action keys are arranged on the keypad is important, although this point was not emphasized during the familiarization phase.) As indicated by their icons on the keypad, these actions are: beep the horn, deliver milk, deliver eggs, receive money, and receive empties (garbage). The number keys correspond to the six houses on the delivery route (depicted in the upper right of Figure 7.1).[2] The CLEAR key is used to delete all the steps from a prior program. Any action can be performed at any house, and both actions and locations can be repeated in a program. A program consists of a sequence of up to 14 action-location pairs.

To help participants remember the programs that they entered, their current program was displayed in the program window. For example, the leftmost program shown at the bottom of Figure 7.1 is the following sequence: beep the horn at house one, deliver milk to house two, and then eggs to house three. Once participants had entered a program, they would click on the RUN key, and the program would be executed. During program execution, an animated milk truck traced out the programmed route in the run window (upper right of Figure 7.1). The truck first exited the garage in the center of the screen, and then moved to each location in the program route in the order dictated by the program. Animated icons were used to indicate what transpired at each location. In Figure 7.1, we see the milk truck picking up money from house 5. At the end of the route, the driver would exit the truck and jump up and down several times. This signaled the end of the route (and helped to keep the participants interested). While programming the route, the participants could use the CLEAR button to clear out the current program to start over from scratch. However, once the RUN button was hit, the program would run to completion.

Immediately to the right of each program window is the trace window. Before the program is executed, the trace window is empty. As each step in the program is executed, the trace window displays the executed step. During the training phase—that is before the introduction of the mystery key—the milk truck executes the steps in the program exactly in the order than they were programmed, and by the end of the program, the content and order of the items in the trace window is identical to the listing in the program window (cf., program 1 at the bottom left of Figure 7.1). However, with the addition of the mystery command, the trace could be different from the program.

Below the number keys on the keypad are the δ keys and its associated parameters. The δ ("delta") command is the mystery command whose function must be discovered. Recall that in the BigTrak task the mystery key (RPT) is a function with only one parameter (a number). In MilkTruck, things are much more complicated, because δ is a function with three parameters. The first parameter following δ must be one of six numbers (1-6). The second parameter is one of two triangles (either ⊿ or ◣). The third parameter is one of two Greek letters (either α or β). When the **δ** command is used in a program, it must always be followed by its three parameters (see programs 2, 3, and 4 in the lower panels of Figure 7.1). For example, Program 3 shows the legal sequence δ, 5, ◣ ,β. This sequence of four special keys must be at the end of the program (i.e., once δ is selected, no other commands may be added).

To appreciate the difficulty of this task, try to figure out the function of the δ command from the example programs and their outcomes displayed in Figure 7.1. Spend a few minutes to come up with at least one hypothesis for the function of each the parameters. What does the number signify? What do the triangles signify? What do α and β signify?

Consider first the second program and its trace shown in Figure 7.1. Early in the task, participants typically describe this kind of outcome as "it switched the second and third steps" or "the delivery to house 2 was put last". Thus, based on program outcomes such as this one, participants typically propose that the **δ** switches steps, with N being the house number of the item that is moved (house 2 in this case), and ◣ signifying that deliveries to that house number should be put last. The implication is that with ⊿ , the action would be put first in the list. However, as can be seen from the traces of programs 3 and 4, the actual function of **δ** is much more complex. At this point, try to propose a hypothesis about **δ** that could account for the three examples in Figure 7.1.

How does **δ** work?

The global function of **δ** N is to reorder—or sort—the last N steps in the program. The particular direction of the sort depends on the triangles: ⊿ sorts in ascending order and ◣ sorts in descending order. The "sort key" (i.e., the thing that determines the sort) is controlled by the selection of either **α** or **β**. For **α** it is the order of the actions on the keypad, and for **β** it is the house number of the command. The two triangles and the two Greek letters can be combined into four different specific sorting functions, as shown in Table 7.1

For example, in program 4 of Figure 7.1, δ5◣ α caused the last 5 steps in the initial program to be executed in decreasing action order. Note that when a step is already in the correct order, its position is left unchanged. For example, in program 3, only three of last five steps are moved—the last two steps were already in the correct order (increasing house number) and thus are unaffected.

Like many scientific discoveries, the function of δ is easy to describe and very difficult to induce from evidence. Successful discovery requires at least two insights. The first insight is that only part of the program is affected (i.e., the last N steps) by the δ command. In other words, like the original BigTrak RPT function, δ functions as a Selector, not a Counter. Without this insight, it is nearly impossible to discover a pattern to the ordering. Moreover, as in the BT microworld, for programs where $\lambda \le N$, this selection effect is undetectable. The second insight is that the particular items and houses that are in the program determine the specific effects of the δ command. Participants typically initially assume that the particular values do not matter. For example, they might believe that δ simply reverses the last N items, regardless of which commands or house numbers are included in that set of commands. Several features of the to-be-discovered rule add to this difficulty. First, the rule itself is a compound conditional. That is, it is of the form “if (*specific triangle*) and (*specific Greek letter*) then (*specific sort function)* over the last N items”. Second, the sort key for β is “natural” in that it uses the number attached to the command, but the sort key for α is arbitrary and local to the context of the MilkTruck keypad. Discovery of these features requires a very high degree of attention to experimental outcomes.

To aid the participants in this complex task, the interface contained several supportive features. Not only was the content of the current program and its outcome displayed in the program and trace windows, but also the programs and outcomes of all previous programs were available for inspection. The seven most recent program and trace listings were displayed concurrently below the run window, and participants could scroll this "history window" to view any and all previous programs (see the bottom of Figure 7.1). When a new program was being entered, the program and outcome of the previous program was added automatically to the history window. Another supportive feature of the interface involved the syntax of program entry: the keys that could be legally selected at a given point in time were highlighted. Moreover, illegal selections were ignored.

STUDY 9: MILKTRUCK

Participants and Procedure

Twenty-two university undergraduates attempted to solve the task in two sessions on two consecutive days. The first day consisted of an introduction phase and a discovery phase. The introduction phase included detailed instruction about the basics of the MilkTruck domain, followed by a description of the syntax of the δ command, and the presentation of the goal of discovering the effect of δ . In the discovery phase, participants designed, conducted, and analyzed experiments in order to discover the role of δ and its arguments. On the first day, following the introduction phase, participants were given 30 minutes in the discovery phase to work at the task until they felt they had solved it, or they wished to give up. If they had not finished after 30 minutes on the first day (19 of 22 did not finish), then they returned the next day to continue.

All programs were automatically recorded on the computer, and all participants' verbal protocols were audio recorded. These 22 participants generated over 33 hours of discovery behavior data, including over 1000 programs and over 110,000 words of verbal protocols. The analyses presented in this chapter are based on several forms of aggregation of these data: 1) the content of the programs generated; 2) statements made in generating programs; 3) statements made in analyzing program outcomes; and 4) the participants' final solutions.

Results

Participants were grouped into three categories: those who discovered how the δ key and its parameters worked (Solvers); those who mistakenly thought they had made such a discovery

(False Solvers); and those who gave up without any solution (Non Solvers). Table 7.2 presents the mean number of experiments and time on task for each group. The task was difficult but not impossible: half of the participants were Solvers, and solution times ranged from 30 to 179 minutes. Note that the Non Solvers did not fail from simple lack of effort: they ran slightly more experiments over a longer period of time than the Solvers. As expected, the MilkTruck task was significantly more difficult than the BigTrak studies described earlier: MilkTruck participants ran three times as many programs over four times as much time.

A typical MilkTruck protocol.

Appendix 7.1 presents the full protocol of a solver (MA). MA is a third year physics major who solved the task in 41.5 minutes and 42 experiments. Each of MA's experiments and their executions are listed in Appendix 7.2. Recall that this history is available at all times to MA (Only 7 experiments at a time are visible, but MA can scroll through the entire history window whenever he wants to.) His experiments and hypotheses are representative of solvers, but he is atypical in one respect: he induces patterns quickly, and solves the task fairly quickly. In this section, we provide an overview of MA's protocol. Although the protocols from this task are quite long, they reveal an unusually rich set of strategies that accompany the discovery process. In subsequent sections, we will return to the protocol to discuss various aspects of the discovery process in further detail.

Like most participants, MA does not have a detailed initial hypothesis. This leads him, like many of the better participants, to begin with a fairly simple experiment. In his first several experiments, MA's primary goal is to find any effect of δ, and this goal is not reached until experiment 4. MA's first hypotheses are in terms of particular steps being "delayed", and these hypotheses are quickly disconfirmed.

In experiments 7, 8, and 9, MA tries again to produce any change with δ. Difficulties in reproducing an effect lead MA to examine how experiment 4 is different from the subsequent programs. This leads MA to propose that perhaps the items matter and that δ may place certain items first. Although the next several experiments produce no confirmation of this hypothesis, MA begins a careful search of the experiment space (experiments 12 - 17) with this general hypothesis in mind.

The outcome of experiment 17 leads to the hypothesis that N relates to the number of steps in the program, which is apparently confirmed in experiment 18. Note, however, that MA is still entertaining the hypothesis that **δ** switches a particular pair of steps rather than working on a larger group of steps (“see if it reverses the first two commands”).

After experiment 21, MA first proposes that the items may be placed in house number order, a very important step towards the final solution. However, this hypothesis is quickly disconfirmed, and does not return until much later.

Following experiment 22, MA reaches an impasse, and lapses into silence—a typical pattern in these participants. In response to this impasse, MA switches from investigating q to investigating π, and, with one exception, does not return to q until the following day.

Beginning with experiment 24, MA embarks on a more systematic experiment space search, contrasting α with β, and π with q. This sequence of experiments enables MA, for the first time (following experiment 26), to propose a complete hypothesis for all the **δ** parameters: that q affects the trash command and π affects the horn command; that β is picking that item second, and α for picking that item first; and that N refers to the number of steps (implicit from previous comments and experiment selections).

This complex hypothesis is immediately disconfirmed by experiment 27. However, in experiment 28, MA carefully varies only N with respect to experiment 27. His choice of program content is particularly serendipitous not only because $1<N<\lambda$, but also because all four of the selected items are rearranged with this π α combination, making the role of N quite salient. Thus, MA is now able to propose the correct role of N—that N refers to the number of items from the end that are changed.

In experiments 29 and 30, MA attempts to induce the function of α and β by varying only that aspect of the program. Since the outcomes are the same, these experiments completely disconfirm the notion that α and β simply refer to opposing orders.

Experiment 31 is conducted with the goal of determining the ordering among the items. The outcome leads to two alternative hypotheses: that π α sorts either in decreasing command order or in decreasing house number order. At this point, MA has all the important components of the correct solution. Experiment 32 then confirms that π α is decreasing command order. In the immediately following experiment, MA proposes that π β refers to decreasing house number order.

At this point, the first day session ends. MA's final guess for that day is completely correct, although not yet adequately tested. Interestingly, during the interval between sessions, MA appears to forget some aspects of this final hypothesis, as indicated by his comments -- on the second day -- during experiments 34 and 35. By experiment 36, MA re-establishes that N refers to the number of steps, and that q is increasing order—however, the role of α is not yet correct.

In experiment 37, MA re-establishes the fact that β places the steps in house number order and a fully correct hypothesis is re-proposed following experiment 38. The remaining experiments are then used to insure that the hypotheses for all α/β q /π combinations are correct, to rule out any alternative orderings.

As this protocol demonstrates, even with a fairly sophisticated experimenter, the path to discovery is complex and fraught with many blind-alleys. How representative is this protocol? Because the MilkTruck task is a complex, multiple-insight problem, the duration and order of occurrence of the various events are idiosyncratic within each participant. However, the set of processes and strategies found within this protocol are highly representative of those found in the other protocols. We shall now consider various process illustrated in this protocol in further detail, applying the SDDS framework.

Experiment space search

One of the most striking differences between these experiments and the earlier BigTrak experiments is the frequency with which MilkTruck experiments were conducted without hypotheses. Recall that in Study 1, more than 70% of BigTrak experiments were conducted while participants had a particular hypothesis in mind. In contrast, only 39% of MilkTruck experiments were conducted with an active hypothesis[3]. Table 7.3 presents the mean proportion of experiments conducted with explicit active hypotheses for each solution group. While Solvers have the highest proportion and Non Solvers the lowest, even the Solvers have hypotheses for only 51% of their experiments. The distinction between H-space search and E-space search corresponds to participants asking themselves "what's going on here?" versus "what can I do next?" It is clear that the complexity of the MilkTruck domain led to an emphasis on the latter.

What factors might contribute to this tendency to spend more time considering what experiment to run next than to thinking about specific hypotheses? We suggest several candidates. First, as we discovered in the mystery key study, when the label on the unknown key conveys no semantic hints about its function -- as in the case of δ -- then it is difficult to have much confidence in any hypothesis generated via pure hypothesis space search, in the absence of data. Indeed, there is an extremely wide range of initial hypotheses bout how δ works, with little overlap in these initial guesses. A second possible contributing factor is that the conditional sorting invoked by the δ key is much more complex than the RPT's single repetition of the final program segment. This makes it much more difficult to come up with even one hypothesis that can account for a current outcome. A third contributing factor is that program content plays a role in the effect of δ. Consequently, for many programs (such as those in which items are already sorted in the order mandated by the δ key and its parameters), there may be no discrepancy at all between program and execution trace. This may produce what appears, at first, to be quire erratic behavior. Finally, the external memory support provided in MilkTruck -- i.e., the display of a full set of previous experiments and their outcomes -- provides an unanticipated extra hurdle for hypotheses. Hypotheses must account not only for the current experimental outcome but also for the past experiment outcomes which are not easily ignored as they are displayed prominently in the history window[4]. Thus, MilkTruck participants frequently devote their experimentation efforts towards uncovering data which might suggest new hypotheses. This corresponds to the Induce Frame process within the SDDS model (Figure 2.9), the very process that was so difficult in the BigTrak context.

If we suppress the detail about the contents of specific commands, then the MilkTruck experiment space is about 50% larger than the 225 cells of the BT experiment space: there are 14 * 6 * 2 * 2 = 336 cells[5] and the two additional parameters in addition to λ and N (i.e., the choice of α or β and q or π) make it more structurally complex. We will begin with an expanded version of the λ - N analysis used previously with BigTrak and its isomorphs, and then we will introduce new aspects that result from the increased complexity.

For our initial analysis, we divided the experiment space into five mutually exclusive regions, as shown in Figure 7.2 Region 1: N = 1; Region 2: $\lambda = 1$, N > 1; Region 3: $\lambda > N > 1$; Region 4: $\lambda = N > 1$; Region 5: $1 < \lambda < N$. Then we looked at how participants from the three solution groups searched the Experiment space. All three groups showed a similar profile: a very large proportion of Region 3 experiments (which are the most informative because $\lambda > N$) very few Region 2 experiments (where, because $\lambda = 1$, outcomes are the least informative), and an intermediate proportion of experiments in the other regions. However, as shown in Figure 7.3, there were some differences across the solution groups. False Solvers had the largest proportion of experiments in Region 1, in which having N = 1 makes them the easiest to interpret but least informative. Solvers had the largest proportion of Region 4 experiments -- in which having $\lambda = N$ makes the sorting action more obvious because the entire program is affected -- and the smallest proportion of Region 5 experiments in which having $\lambda < N$ masks the function of N. Thus, there appears to be a similar relationship between the λ-N regions and solution rate as in the BigTrak tasks. We will return to this relationship subsequently, in our analysis of hypothesis space search.

In addition to analyzing λ-N regions, we also investigated search through the triangle-Greek letter regions. Over all programs, participants were more likely to use α than β (64% versus 46%) and equally likely to use either of the triangles. Temporally, participants typically began with α, and α was more difficult to understand and thus was used in more experiments. In contrast, although participants typically began with q, it usually did not produce any changes (because the steps were commonly already in order), and thus participants switched to investigating π. Interestingly, there were no relations between use of these parameters and solution group,[6] suggesting that this aspect of the experiment space search was not as important as correctly navigating the λ-N regions.

What strategies did participants use to search the complex MilkTruck experiment space? Some of these strategies related to aspects of λ-N regions as with BigTrak. However, other strategies related to the expanded MilkTruck regions, which included the Greek letters and the triangles. In terms of λ-N regions, there was one very popular strategy: A majority of the participants used a strategy of starting simple and building-up towards more complex experiments. Use of this strategy is indexed by analyzing changes in program length over the discovery session.. The mean length of first programs was 2.7 steps. By the end of the task, participants were using programs between 6 and 14 steps long. Figure 7.4 illustrates this gradual increase in mean program length over task quartiles. All three solution groups had this same gradual increase across quartiles, although Solvers began with slightly smaller programs and finished with slightly longer programs than the other two groups.

Another strategy for searching the MilkTruck Experiment Space is one we call the Put Upon Stack Heuristic (PUSH). We define PUSH as follows: 1) When confronted with an impasse while investigating a particular region of the experiment space (e.g., πα), explicitly switch to investigating a different region of the experiment space (e.g., πβ) with the goal of returning to the confusing region later. 2) Upon successfully completing the investigation of the new region, return directly to the old region, rather than a third region (e.g., qβ). In other words, when participants use PUSH, they do not entirely abandon the confusing experiment space region; instead they momentarily defer investigation of it. The time spent away from the problematic region proved useful as participants not only solve the problem in the new region, but also, upon returning, quickly resolve the confusion underlying the impasse in the old region. We observed the used of PUSH in several of the Solvers' protocols, but in none of the False Solvers' or Non Solvers' protocols.

PUSH can be useful in three different ways. First, by enabling participants to work on a different problem, PUSH allows new ideas to become activated and the activation of old ideas to decay, thereby reducing set effects and affecting the hypothesis space search in the old situation. Second, the investigation of a different problem can suggest new operators which may be applied to the old situation, thereby improving the experiment space search. MA’s protocol (Experiments 22-23) provides evidence for this use of PUSH:

> “Yeah. I’m just baffled right now. OK, let’s take a different tack. See if we can figure something out from a different angle. Let’s go at it, looking at these other, what this other triangle does.”

Then, for the first time, MA ran two programs which directly contrasted α with β, i.e., varied only α/β between two programs (Experiments 24-25). This approach produced useful information and lead MA to say (Experiment 26):

> “Let’s try the same combination with other triangle again. I have a feeling that this might be what I need to be doing all along. The white triangle. We’ll start with alpha.”

MA then went on using this type of newly discovered contrast, among others, to successfully solve the task.

The third way that PUSH is useful is that in inducing a complex concept involving interactions (such as those underlying the δ command), discoveries about one part of the concept facilitate discoveries about another part of the concept. Thus, as the predictive power of hypotheses improve, the easier it is to resolve the remaining ambiguities.

A related strategy identified in previous research is the Investigate Surprising Phenomena (ISP) strategy (Kulkarni & Simon, 1990), in which people focus their attention on surprising experimental results and set a subgoal to determine what caused the surprising results. On the surface, the two strategies would seem to be incompatible: in the face of difficulty, one strategy (PUSH) advises switching to a different experiment space region, whereas the other strategy (ISP) advises redoubling effort in this exact experiment space region. However, the two apply to slightly different situations. When the surprising phenomenon has some unique, salient characteristics or features that can be tested in follow-up experiments, then ISP applies. On the other hand, when all the salient possible reasons for the surprising phenomena have been exhausted, then PUSH applies. In this case, the strategy is to defer further investigation until new information has been gathered.

Two additional aspects of search in the MilkTruck experiment space deserve mention. The first is the role of confirmation in experimentation. The SDDS model suggests that individuals go through a natural cycle of using experiments to generate hypotheses followed by experiments to test the generated hypotheses. The structure of the MilkTruck task enables us to see whether this simple cycle generalizes to a situation in which there are multiple components to discover. That is, do people try to induce a complete hypothesis and then test it, or do they test the partial hypotheses as they are generated?

We found that MilkTruck participants tended to follow both approaches. In the example protocol, MA usually conducted a hypothesis-testing experiment to evaluate newly generated hypotheses. For example, based on the outcome of experiment 4, MA proposes that N refers to the step that is placed last, and then tests this hypothesis in experiment 5. However, at the end of the second session, MA conducts an additional series of complex experiments (experiments 38–42) all designed to confirm his current hypothesis. This was a common pattern among the MilkTruck participants.

The final aspect of search in the MilkTruck experiment space that we will discuss relates to our own operational definitions. Thus far, we have been defining an experiment as a single program and its outcome[7]. However, because participants have access to the results of all previous programs, an experiment could also be defined as an aggregate of more than one program. For example, some participants ran a series of programs in which they held the base program constant and varied the **δ** parameters systematically (e.g., 2qβ, 2πβ, 3qβ, 3πβ, etc.). These participants spent little time examining the intermediate outcomes. Instead, they waited until the end of the series before attempting to induce some regularities over the set of outcomes. A related strategy is to add or remove a feature with respect to the previous program and see how the outcome differs. In both of these approaches, each program is a "cell" in a larger factorial experiment.

In generating such experiments, one must decide how many features to vary from one program to the next. Figure 7.5 presents the mean number of features varied by the different solution groups across quartiles of the task. There were four features defined for this analysis—one for each of the **δ** parameters and one for remaining content of the program. Overall, participants tended to vary two or three features at a time between successive programs, and the overall effect of solution group was not significant. However, there was an interaction between quartile and solution group: Solvers varied the fewest features early in the process. In the later quartiles, Non Solvers and False Solvers also varied fewer features. False Solvers varied the fewest features, and this may have prevented them visiting E-space regions that would have disconfirmed their incorrect hypotheses.

Valid inferences from experimental contrasts demand that only one feature can be varied at a time. Under this strict view, few participants could be described as having created unconfounded experiments because most of them varied two or three features from one experiment to the next. However, there are two reasons why our index -- the number of feature changes in between successive programs may overestimate participants' penchant for confounded experimental comparisons. First, while our measure is based on comparisons between adjacent programs, participants may be basing their experimental contrast for experiment N+1 on one that occurred earlier than experiment N. Second, the structure of the changes within a single program provides a large portion of the information. For example, the exact coincidence of N=4 and the last four items of a program being changed suggests that N is the number of items from the end that are changed. In this second case, a single program can serve as a full experiment. In such experiments, which can be thought of as case studies, the participant looks for identity relationships between aspects of the program and the outcome (e.g., noticing that the δ number is 2 and the delivery to house 2 was moved last).

Hypothesis space search

We begin our discussion of search in the hypothesis space by focusing on aspects that are similar to BigTrak and then move on to novel features resulting from the increased complexity of MilkTruck. Developing the correct hypotheses in MilkTruck required three primary discoveries: 1) that N acts as a selector on the last N items in the program; 2) that -- in contrast to BigTrak -- the items in the program (i.e., (house numbers and actions) matter ; and 3) that action of the δ command is to sort the steps. These three insights were approximately equally difficult (achieved by 16, 18, and 17 of the 22 participants respectively). Moreover, none of them was more likely to occur before the others.

How independent were these three discoveries? While there was no relation between discovering the selector function of N and the other two insights, there was a significant relation between success and the other two discoveries. Of the 17 participants discovering that δ sorts the steps, 94% discovered that the items matter, whereas only 40% of the remaining 5 participants discovered that items matter. It is not entirely surprising that there is some relationship between the sort and item discoveries, since the sort discovery implies that something about the step matters. However, the two are separate discoveries, with neither insight being necessary nor sufficient to induce the other. Moreover, of the 16 participants who made both discoveries, only 3 both made both discoveries on the same experiment.

What was the impact of experiment space search activities on these insights? Discovering the selector role of N was associated with proportionally fewer N>λ experiments (.07 versus .24), more N<λ experiments (.55 versus .44), and more experiments involving β (.39 versus .22). Thus, the importance of N<λ experiments is the same as it was in BT. Discovering that items matter was associated with proportionally more λ = N experiments (.22 versus .08), fewer N<λ experiments (.48 versus .68), and more α experiments (.66 versus .53). The reason for the association with α experiments is obvious—items didn't matter for β experiments. The importance of λ = N experiments is also intuitive—having the full program sorted made the sort function more apparent. Discovering the sort function was unrelated to these gross-level λ-N and Greek letter-triangle E-space regions. However, it was related to the more specific conjunction of λ>4 and N>4 experiments (.26 vs. .15), whose results are sufficiently complex to push the participants away from simple item-swap hypotheses (e.g., "it swaps the last item with the Nth item from the end"). Thus, as with BigTrak, MilkTruck displays an important correspondence between E-space and H-space search—certain hypotheses are less likely to be entertained without visiting the appropriate E-space regions. In the next sections we turn to some of the novel features of hypothesis space search in the MilkTruck domain.

Piecemeal Induction

One of the most striking features of participants' search in the MilkTruck hypothesis space is the incremental fashion in which hypotheses were constructed. Rather than developing monolithic hypotheses about the function of δ, the participants developed smaller hypotheses about each of the parameters, typically one parameter at a time. We call this process piecemeal induction. Because it constructs only partial hypotheses, piecemeal induction tends to produce a large number of hypotheses. MilkTruck participants produced on average of about 32 hypotheses each,[8] and one participant produced 63 unique hypotheses.

In piecemeal induction, new hypotheses frequently add to, rather than replace, existing hypotheses. For example, in experiment 19, MA has the hypothesis that N represents the number of items in the program (from experiment 17). From the outcome of experiment 19, MA adds the hypothesis that qα places the horn command first. Here, the hypothesis for the function of N is retained, and the hypothesis regarding the qα combination is added.

Another form of hypothesis change involves the combination of component hypotheses to form more general hypotheses. For example, a participant might generalize the hypotheses that N=2 changes the last two steps and N=3 changes the last three steps to the more general hypothesis that N changes the last N steps. A variation of this generalization process involves segmenting hypotheses about parameter combinations into separate, more general hypotheses about each parameter. For example, from the hypotheses that πα sorts items in decreasing action order and πβ sorts them in decreasing house number order, participants frequently generalize to the hypothesis that π refers to decreasing order.

Hypothesis generality

Not only do MilkTruck hypotheses vary in terms of their referents (objects, boundary conditions, or both), but also they vary with respect to the generality versus specificity of those referents. For example, in the hypothesis, “with black triangle alpha, N refers to the number of steps from the end of the program that are reversed”, the object is N. The generality-specificity of boundary conditions refer to the limitations on applicability of the hypothesis. In this example, the boundary conditions are πα . That is, the hypothesis regarding the object N is true only when the boundary conditions π and α are met. Table 7.4 presents examples of hypotheses with general and specific objects and boundary conditions.

Boundary conditions occur frequently in both the physical and social sciences. For example, the laws of Newtonian physics are assumed to apply only at speeds significantly below the speed of light. In the social sciences, hypotheses regarding group structure are often specific to boundary conditions such as sex, age, ethnicity, etc. In this domain, the generality of both object and boundary conditions varies from combination hypotheses to specific parameter hypotheses to general parameter hypotheses. We have already seen several examples of combination hypotheses (e.g., "πα sorts decreasing function", "πβ reverses the steps"). Some combination hypotheses are very specific: "2πβ with an even number of items swaps the last two steps". Specific parameter hypotheses refer to specific values of a parameter (e.g., N=2, π , β). At the most general end of the generality-specificity continuum are general parameter hypotheses which refer one of the three general parameters—N, the triangles, or the Greek letters.

Table 7.5 presents the relative frequency of object hypotheses at various levels of generality. The most common hypotheses involve specific triangles, specific Greek letters, specific combinations and the general N. Interestingly, for the triangles and Greek letter parameters, specific value hypotheses were more common than general value hypotheses, whereas for the N parameter, the reverse was true. This difference may reflect participants' general belief that N, being an interval scale, should involve a general function. This presumption would preclude proposing hypotheses for which the value of N could not be generalized immediately. In support of this assumption, specific N hypotheses are considerably less frequent that specific triangle and specific Greek letter hypotheses. However, the difference could also reflect the ease with which general hypotheses for N are proposed—general hypotheses for N are three to six times as frequent as with the triangles and the Greek letters.

How did the generality of hypotheses vary over time? Typically, the participants began by trying to propose very general hypotheses. For example, MA's first hypothesis (developed following experiment 4) was that N refers to the house number that is placed at the end (in his terms, which action it "delays"). In this complex task, these first general hypotheses failed miserably. Following these early failures, participants attempted to build more specific hypotheses, accounting for regularities of more limited scope. For example, following experiment 5, MA proposes that the specific combination qα "delays" milk commands. Over the course of the discovery processes, participants attempted to generalize these specific hypotheses to more general hypotheses. Of course, this process was iterative, as the second, third, and fourth attempts at general hypotheses often failed. However, by the end of the task, all of the Solvers and False Solvers proposed general hypotheses. It seems that the participants considered a single general hypothesis as considerably more desirable than many specific hypotheses.

One final aspect of hypothesis space search deserves mention: when are new hypotheses proposed? The SDDS model (see Figure 2.9) assumes that new hypotheses are proposed either during the evidence evaluation phase (in Review Outcomes, where the outcomes of one or more experiments are examined) or during the Induce Frame process. For the MilkTruck task, hypotheses were proposed primarily (79% of the time) while the participants were encoding the outcome of the most recent experiment rather than while examining outcomes of previous experiments (5% of the time). It appears that new data are used to suggest new hypotheses and previous data are used to confirm or verify them. Surprisingly, , a substantial proportion (15%)of hypotheses were also proposed during the design of an experiment . However, it appears that for the majority of these, participants had actually developed the hypothesis earlier, while examining the outcome of an experiment, but did not mention the new hypothesis until in the middle of designing the next experiment.

Consider several examples from MA’s protocol. On five occasions, hypotheses were first mentioned during the design of an experiment. In four of these cases—experiments 5, 10, 18, and 41—the most plausible assumption is that the hypothesis motivated the design of the experiment rather than vice versa, since there are usually direct precursors to the hypothesis to be found in the outcome interpretation of the immediately preceding experiment. For example, the hypothesis “let's see if the delta changes the priority in which they are done” that occurs in the design of experiment 10 is preceded by the statement “I wonder if it is like a priority thing” in the interpretation of the outcome of experiment 9.

The fifth case, experiment 6, does not seem to follow this pattern. The quest for a new approach to designing experiments (varying the triangles and Greek letters) led MA to the examine how the previous experiments fit in this scheme, which in turn led to the proposal that q α delays the delivery of milk. This example shows how experiment space search can have a direct influence on hypothesis space search (by producing a new way to interpret previous outcomes) in addition to the more typical sequence in which experiments generate data which is then analyzed to produce hypotheses.

Data Representation space search

Our analysis thus far has been couched in the now-familiar terms of the dual space framework, that is, in terms of search in the experiment space and the hypothesis space. However, analysis of the MilkTruck protocols led us to formulate an entirely new space: the Data Representation space.

Consider the four excerpts from participant MW listed in Table 7.6. Of particular importance are the different the aspects of the MilkTruck’s behavior to which the MW attends. In the first excerpt, MW counts the number of times the milk truck driver jumps up and down at the end of the program. He makes no mention at all of the content of the program. In the second excerpt, MW attends to the path of the milk truck in the run window, in particular whether milk truck took the most efficient route from house 1 to house 6. In the third excerpt, MW continues to attend to the path of the milk truck, but now he characterizes it in terms of different features of the path: "top row," "pass at four," "path evidently was back." In the final excerpt, MW refers to the commands used and their ordering as a group. Note that only the commands (“Milk. Milk. Eggs. Eggs.”) and not the house numbers are mentioned. In these four excerpts, there is little overlap among the features that are considered as data from one excerpt to the next. During the course of his session, MW’s conception of what counts as data (i.e., his representation of the data) changes radically. That is, in addition to changing hypotheses, MW changes what aspects of the data are worth encoding and how to characterize the phenomenon that is to be explained.

How might one formally define the Data Representation space? Since data representations are usually not distinguished from hypotheses by researchers studying scientific discovery (cf. Schunn & Klahr, 1996), we will provide a formal definition of the Data Representation space in the context of a definition of the hypothesis space. The hypothesis space involves propositions about the world, potentially at varying levels of abstraction and of varying levels of generality. For example, one might have the following hypotheses—varying in scope and abstraction—about the results of an experiment on reading comprehension: "there was no effect of manipulating motivation in this experiment (on comprehension of some history text)", "there is generally no effect of monetary reward on comprehension of history texts", or "there is generally no effect of motivation on reading skills". In our earlier discussion of hypothesis space search, we provided several examples of MilkTruck hypotheses at these different levels of scope and specificity.

By contrast, the Data Representation space involves the objects and object features of the data. For example, one might graph different relationships between different variables, one might use different kinds of graphs, and one might re-code, collapse, or expand the same data. In the MilkTruck domain, objects include the jumps of the driver at the end of the route, the way in which the milk truck exits the garage at the beginning of the route, the path taken in the run window during the delivery route, the individual steps in the program and trace listings, groups of steps in program and trace listings, and the δ parameters. Each one of these objects has multiple features. For example, the individual steps in the program and trace listings have the following features (all of which were spontaneously mentioned by at least some of the participants): house color, house number, house number parity (even/odd), top row houses versus bottom row houses, Nth step (first, second, third,...), Nth from the end, action, and action type (deliveries versus pickups).

The two spaces also differ in terms of their goals. The goal in searching the hypothesis space is to produce parsimonious explanations and/or descriptions of objects and relations in the world. By contrast, the goal in searching the Data Representation space is to find regularities. A data representation is abandoned if it doesn't lead to regularities or interpretable patterns, whereas it is maintained when it does. In other words, people search in the Data Representation space in order to find regularities, and search in the hypothesis space in order to explain them.

We coded participants' data representations by using their descriptions of experiments as they designed them and interpreted their outcomes. Thus, data representations could be revealed by experiment design statements (e.g., "Let's try commands going to different blocks"), outcome prediction statements (e.g., "See if there is any, uh. If I can get any differences in the path."), data description statements (e.g., "So the path evidently was back."), and hypothesis statements (e.g., "White triangle and alpha groups deliveries by the commands.").[9]

For each experiment (design and outcome interpretation combined) we coded for the presence of 7 objects and 30 total object features. Although at least one feature was typically coded with the presence of each object, this was not necessarily true—it is possible the participant did not specify any particular feature of an object. We did not code the frequency of mention of a given feature within each experiment; but rather simply its presence or absence—since the verbal protocols are not always complete, it was assumed that a finer level of analysis would be unreliable. Initial and final hypotheses were also coded, but counted separately from the experiments.

How did the data representations typically change over the course of each participants' discovery process? At the gross level, there were shifts in the types of objects that the participants included in their data representations. Figure 7.6 presents the mean frequency of each object within each quartile of the task. While there was no overall increase in the number of objects across quartiles, the main effect of object type and the interaction of object frequency by quartile were significant. Man jumps, garage exits, and delivery path were objects that appeared primarily in the beginning of the task—the participants quickly learned to exclude those features from their data representations.

Both program step and δ objects remained quite frequent throughout the task. The program steps and δ objects were the level at which the programs were entered (i.e., one step at a time), and this aspect of the computer interface may explain the high frequency of use for steps and δ. Thus, if programs could have been entered in some other way (e.g., cutting and pasting larger program segments from previous programs) then the frequency of discussion of individuals program steps might have changed over the course of the task. The reason for the constant, frequent presence of the δ object is obvious—it is the target of the discovery task.

The use of program segment and whole program objects increased steadily over the course of the task, with the largest increases occurring in the use of house number order, command order, and # of steps in a segment—these features corresponded to the actual function of the δ command. Thus, the gross level changes in data representations reflected the influence of the target function of δ—those features which were not relevant were weeded out, and those features which were relevant were gradually added. Moreover, although participants changed their focus from one set of objects to another, they usually had several objects and features in each experiment. The mean number of objects (2.7) and features (5.4) did not change over the course of the task, nor were they related to success.

At the micro-level, the changes in data representation from one experiment to the next were quite striking—for all of the participants, frequent change was the norm. Over 97% of transitions from one experiment to the next involved at least one data representation change and over 71% of experiments involved a change in which objects were included. Adding features, deleting features, and replacing features (i.e., adding and deleting) were all common activities (see Table 7.7)

In sum, the search through the space of data representations in the MilkTruck task was a frequent activity. From one experiment to the next, the objects and object features used to the describe the data changed more often than not. And, by definition, only by converging on the correct data representation could participants discover how the δ key worked.

THE EFFECTS OF A MORE COMPLEX DISCOVERY MICROWORLD

In this chapter, we have addressed one overarching question: How do the discovery processes described in the preceding chapters change as the discovery task becomes more complex? The data from the MilkTruck task suggest that while some important new features of the discovery process are introduced, the earlier features remain. Most importantly, behavior in MilkTruck can be described using the constructs from dual space search: experiments constrain search in the hypothesis space, and hypotheses constrain search in the experiment space.

What features of the discovery process changed as the discovery task became more complex? First, we found evidence of several new experiment space search heuristics which are useful in managing the increased complexity of the task. For example, the participants started with relatively simple experiments and gradually switched to more complex experiments, and they also made use of the PUSH heuristic.

Second, we found more complex kinds of hypotheses and patterns of hypothesis change. Hypotheses involved many components which could vary in generality along several dimensions. Moreover, hypotheses were found to be built up in a piecemeal fashion rather than from whole cloth.

Third, behavior in the MilkTruck task lead us to propose the existence of a new space: the Data Representation space. We believe that this new space is not unique to the MilkTruck task. Whenever the discovery task is difficult and there are many features to which one could pay attention, we expect that people will explore the Data Representation space. As a common real-world example, scientists must frequently decide what dependent measures to gather in their studies or how to code and plot their data. These are examples of search in the Data Representation space. Yet, previous research on scientific reasoning did not propose a Data Representation space. Why is this? We believe that data representation change has frequently been assimilated into the hypothesis space. In other words, researchers have simply assumed that changes in descriptions of the data simply reflected changes in hypotheses. Also, in some very simple scientific reasoning tasks, it may have been that the important features were immediately identifiable.

The addition of such theoretical entities raises an important meta-theoretical question: will entirely new theoretical entities be needed to account for the much more complex discovery tasks found in real science? Moreover, the decision to add theoretical entities like new spaces must be carefully justified (cf. Schunn & Klahr, 1996). While our experience with increasing complexity in a small way suggests that new entities are likely to be needed, we expect that the features that we have described thus far will remain important in the more complex situations, just as the hypothesis space and experiment space used to describe behavior in the BigTrak task were useful in describing behavior in the MilkTruck task. In the next chapter, we will examine how these new theoretical entities might be included in the SDDS framework.

Footnotes

1. While the MilkTruck task has been used to address a variety of questions about scientific discovery (e.g., Schunn & Klahr, 1992, 1993, 1995, 1996), in this chapter, we describe only the first in our series of MilkTruck studies, and we focus on the basic impact of the more complex interface on discovery processes.

[2]. Note that this is unlike BT, where the numerical part of each command corresponded to the number of times that an action was executed (e.g., "Fire 5" led to the cannon being fired 5 times). In BT, this tended to reinforce subjects propensity to view the N associated with RPT as a counter rather than as a selector.

[3]. These numbers were derived from a detailed encoding of the verbal protocols from the first 15 participants.

[4]. Crick's opening caveat to the contrary notwithstanding.

[5]. There are from 1 to 14 program steps, 6 possible houses associated with each action, two values for the Greek letter, and two kinds of triangles.

[6]. However, there was a weak trend for Solvers to have the smallest proportion of α experiments (the more difficult Greek letter) and Not Solvers the highest proportion (.61 versus .70) .

[7]. Or, in a few cases in Study 1, as a pair of programs: a "control" experiment without a δ followed by the same program with a δ.

[8]. The data regarding the hypothesis space search is based on careful protocol analysis of the first 10 participants. Data from one of the participants was removed because that participant typically did not verbalize his current hypothesis. Thus, the N for these analyses is 9.

[9]. These examples are all taken from the four excerpts shown in Table 7.6.

Table 7.1.

Summary of the function of the δ command, in conjunction with parameters α, β, ◿ , and ◣ .

	δN re-orders the last N steps in the program according to:	
	◿ (increasing)	◣ (decreasing)
α (action)	... action in increasing keypad order: Beep, Milk, Eggs, Money, Garbage	... action in decreasing keypad order: Garbage, Money, Eggs, Milk, Beep
β (house)	... house in increasing number order.: 1, 2, 3, 4, 5, 6	... house in decreasing number order.: 6, 5, 4, 3, 2, 1

Table 7.2.

Mean number of experiments and time on task for each solution group (and standard deviation).

Group	n	Experiments (sd)	Time on task (sd)
Solver	11	49.5 (18.6)	78.6 (40.9)
False Solver	5	33.6 (19.6)	61.6 (42.8)
Non Solver	6	54.5 (17.6)	118.8 (17.8)
All Groups	22	47.2 (19.4)	85.7 (41.2)

Table 7.3.
Mean proportion of experiments (and standard deviation) for which participants had a current hypothesis, for each solution group.

Group	n	Proportion of experiments having explicit hypotheses
Solvers	7	.51 (.06)
False Solvers	5	.33 (.18)
Non Solvers	3	.20 (.14)
Total	15	.39 (.17)

Table 7.4.
Examples of general and specific **objects** and *boundary conditions*.

	Boundary condition	
Object	General	Specific
General	LS Exp 12: The **triangle** is just on and off.	MW Exp 44: The **number** is simply the number of deliveries that it will organize for alpha, er, for *white alpha*.
Specific	MA Exp 6: **White triangle alpha** was the milk got delayed.	MW Exp 26: Delta **five**, seems like it reverses them. *Except where they are the same command*.

Table 7.5.

The mean of participant (N=9) proportions (and SD) for each type of object hypotheses in descending order of frequency, with examples and, when possible, experiment number pointers to MA's protocol.

Object	Example	Frequency	MA Exp#
Specific triangle	White triangle	.21 (.06)	26
Specific letter	Beta	.21 (.09)	33
Specific combination	White alpha	.20 (.16)	19
General N	N	.17 (.09)	18
Specific N	N=3	.12 (.09)	-
General letter	Alpha/Beta	.05 (.06)	30
General triangle	White/Black	.03 (.05)	-
General combination	N and Triangles	.01 (.03)	-

Table 7.6. Excerpts from MW (Biomedical engineering sophomore)

Excerpt 1:	
Exp. 3 B1 δ3qα	OK. I'm going to try this one more time. I'm going to beep a house one. Use the delta command. This time I'm going to do three times with the white triangle and alpha. And run.
- B1	That blew that hypothesis. I thought after the delta that, uh, these numbers related something to the person jumping and the number of times. I guess it didn't work. He jumped like six times with output three. Next one. I'm going to change a little bit. By using delta again. I'll do delta three again. I'm using the black triangle this time with alpha.
Excerpt 2:	
Exp. 12 (11 42) B1 M6 δ6qα	Uh, well. Let's try commands going to different blocks. From one house one to house six. Delta. (BEEP) Whoops. Sorry. Um. Six. White triangle and alpha. It doesn't allow you to go back to commands does it? Hmm. Delta has to have something to do with the finishing of the route. Cause you can't put anymore commands after you put in a delta. ... But. Let's see what is going to happen.
- B1 M6	It took the long route for some reason. ... Hmm. Uh. I'm just going to write down something. Just what happened there. So it went to ... *Keep talking.* I'm just writing down the path, and what the last, the delta value that I put in. See if there is any, uh. If I can get any differences in the path.
Excerpt 3:	
Exp. 21 (21 47) B4 M5 E6 δ2πβ	Wait a second. Something different happened this time. Ahhh! For some reason, it did the commands in opposite. Let me test something. Uh. I'm going to do the same commands, except I'm going to use the bottom row. Just to see what it does. Hopefully, if I use a number up in the top row, it will reverse it. ...
- B4 E6 M5	Ah, let's see. It should pass at four. Maybe not. ... What. This is totally messed up now. Whoa. Wait a second. ... I'm going to write down what it did for these last two programs. Do anything different. So the path evidently was back. This was delta five black beta. And the last one went to 4 to 6 and back to 5. And I have no idea why. Um. ...

Excerpt 4:

Exp. 40 (20 55)
B1 M2 E3 E2 M1 B3 δ 6qα
-
B1 B3 M1 M2 E3 E2

It seems if I put a value of n for delta that's less than the number of commands, it doesn't follow that grouping deliveries by home. ... I won't worry about that. I'll just. I'll do something that I am sure of right now. And see what alpha does to the program. I'll come back to the numbers later.

... Milk. Milk. Eggs. Eggs. Well that's real cool. White triangle and alpha groups deliveries by the commands. I mean by what the deliveries are. It just groups them together. ...

Table 7.7.
The mean frequency of each data representation change type from one experiment to the next for objects and object features.

Entity	Addition	Only Addition	Deletion	Only Deletion	Addition and Deletion
Objects	.45	.27	.44	.26	.18
Features	.81	.21	.76	.16	.60

Figure Captions

Figure 7.1 Lay out of the MilkTruck environment, showing the keypad (upper left), current program and trace (upper middle), run window (upper right), and history window (bottom).

Figure 7.2 . Regions of MilkTruck experiment space, with examples.

Figure 7.3. The mean proportion of experiments in each λ-N E-space region for each solution group.

Figure 7.4 The mean length per program (and standard error) within each task quartile.

Figure 7.5. Mean program feature between successive programs for each solution group for each quartile of total experiments.

Figure 7.6. Mean frequency of data representation objects within each task quartile.

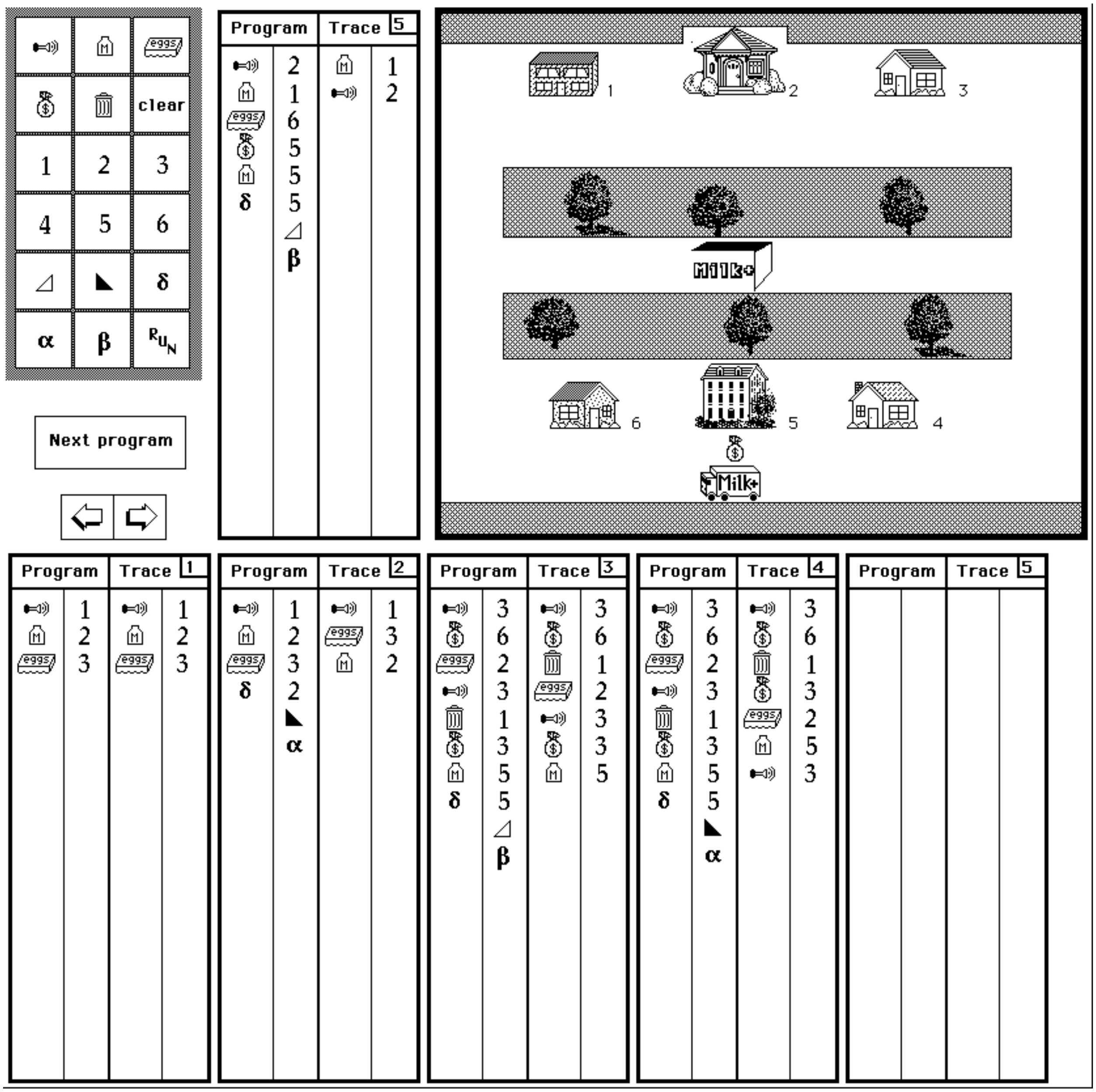
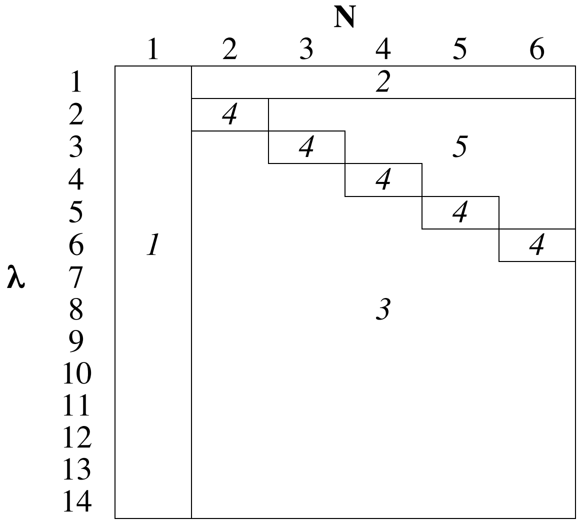


Figure 7.1 Lay out of the MilkTruck environment, showing the keypad (upper left), current program and trace (upper middle), run window (upper right), and history window (bottom).



Region	Boundaries	Example
1	N=1	B1 M2 δ1πβ
2	λ=1, N > 1	B1 δ3πβ
3	λ>N>1	B1 M2 E3 $4 δ2πβ
4	λ = N>1	B1 M2 E3 δ3πβ
5	1<λ<N	B1 M2 E3 δ5πβ

Figure 7.2. Regions of MilkTruck experiment space, with examples.

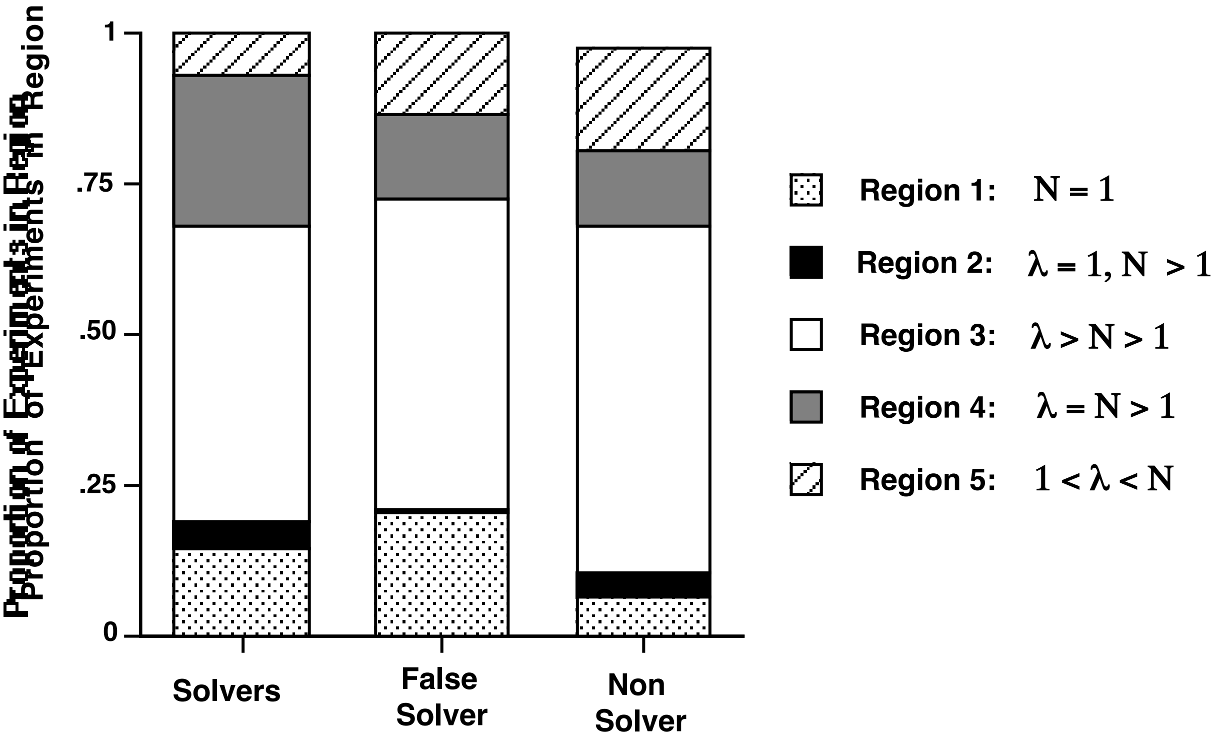


Figure 7.3. The mean proportion of experiments in each λ-N E-space region for each solution group.

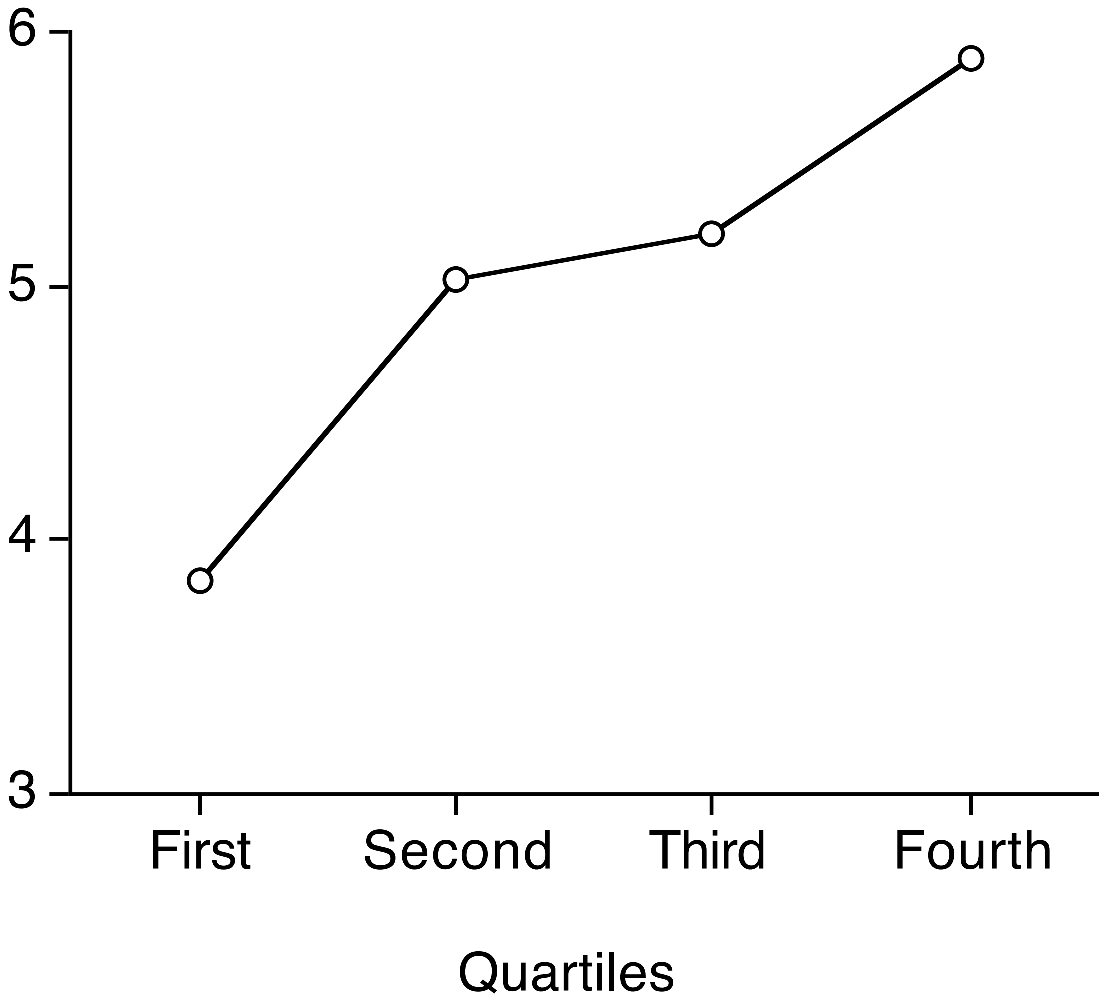


Figure 7.4.

The mean length per program (and standard error) within each task quartile.

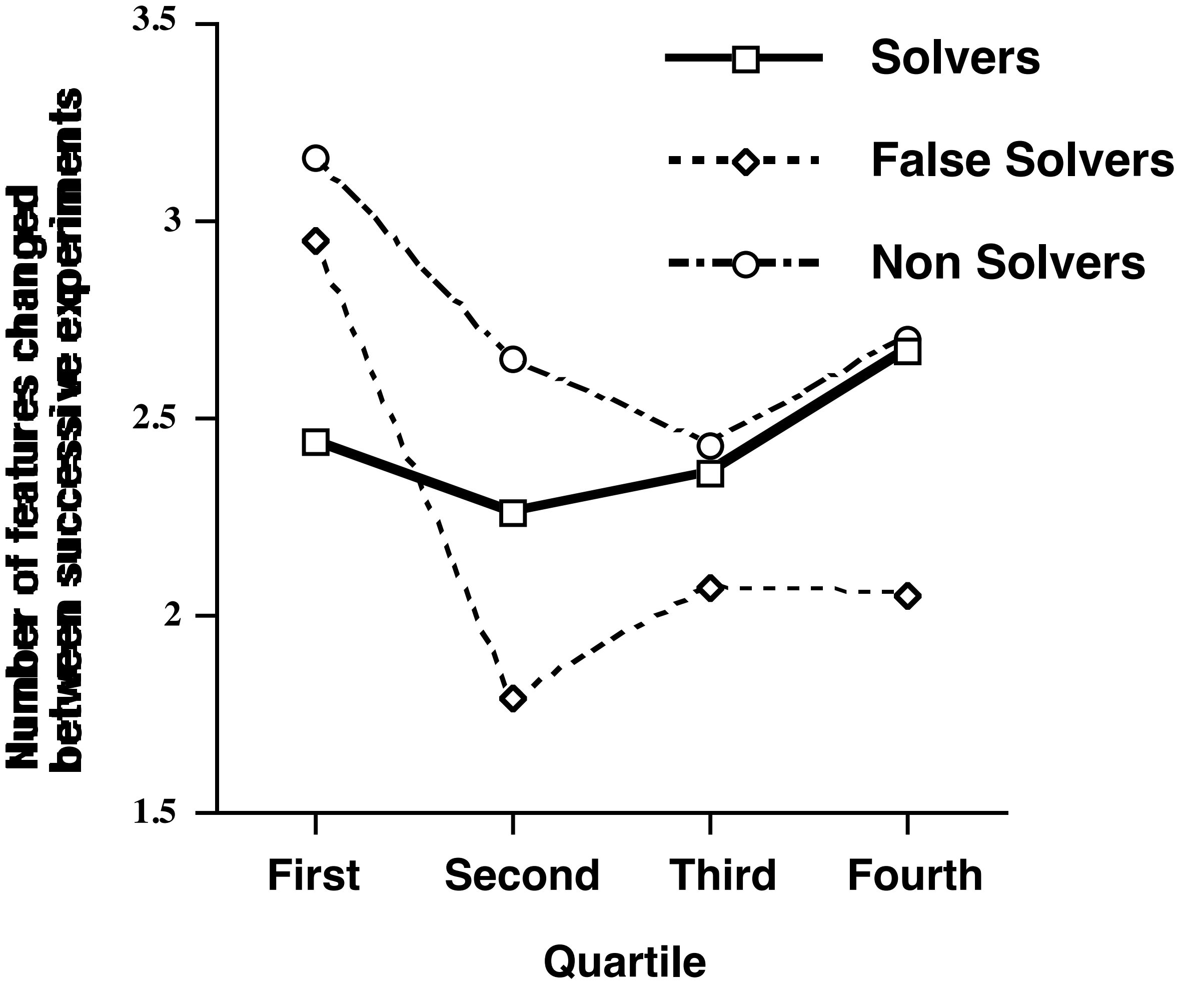


Figure 7.5.
Mean program feature between successive programs for each solution group for each quartile of total experiments.

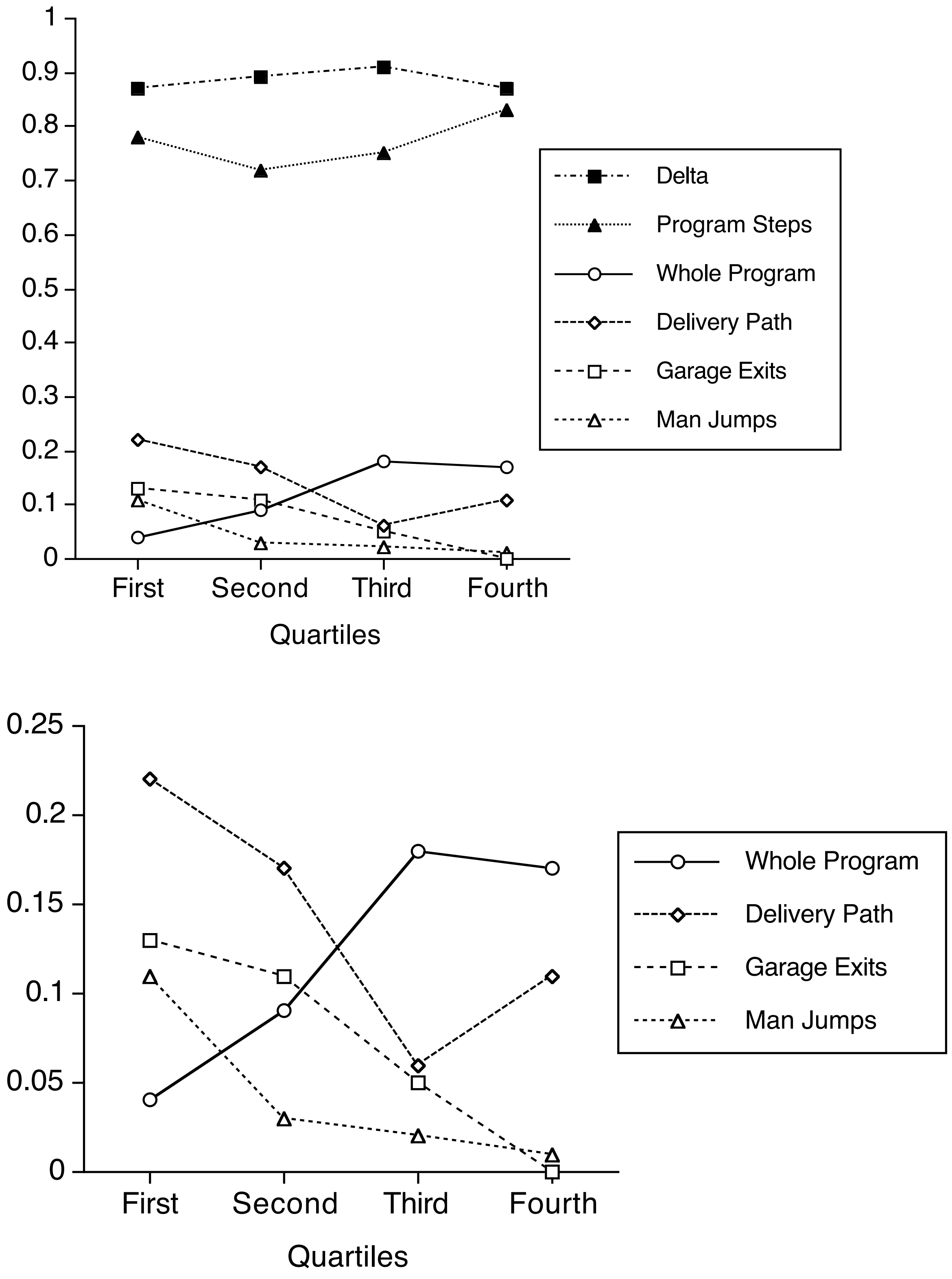


Figure 7.6. Mean frequency of data representation objects within each task quartile.