Instruction and Practice in Learning to Use a Device

PETER A. BIBBY
University of Nottingham

STEPHEN J. PAYNE
University of Wales, Cardiff

We explore the extent to which Anderson's (1987) theory of knowledge compilation can account for the relationship between instructions and practice in learning to use a simple device. Bibby and Payne (1993) reported experimental support for knowledge compilation in this domain. This article replicates the finding of a performance cross-over between instruction type and task type that disappears with practice on the tasks. The research is extended by using verbal protocols to model the strategies of novice and more experienced individuals. Production system models of these strategies suggest that knowledge compilation only provides an adequate account of practice for one of two instruction groups. To model the strategy shifts for the second group, we employ a "procedure modification heuristic" (after Neches, 1987), which relies on access to a declarative model of the configuration of the device. This suggests that instructionally derived declarative knowledge may have a subtle ongoing effect on the changes in procedural knowledge with practice.

1. LEARNING TO USE DEVICES FROM INSTRUCTIONS

The acquisition of cognitive skill in most meaningful domains involves both learning from instructions and learning from practice. Consider learning to use a device, like a computer system. Typically the device will support a number of different tasks. With a few exceptions (so called "walk-up" systems), getting started on these tasks can only be achieved with the help of instruction, usually some mixture of general information about the device and specific information about the tasks. A large research literature has grown around the problem of how best to design such instructions (Carroll, 1990; Wright, 1988). Once the tasks can be attempted, learning can occur to
Anderson's (1982, 1983, 1987) ACT* theory of skill acquisition offers a clear, general account of the relation between instructions and practice in the acquisition of procedural skills. A central claim of the ACT* theory is that the two processes are quite separate at the cognitive level: Learning from instruction implicates the encoding of new declarative knowledge, from which task strategies (procedural knowledge) can be inferred by general problem-solving heuristics. Learning by practice is achieved through mechanisms that tune this task-specific procedural knowledge. Even expressed at this level of generality, this "knowledge compilation" model makes quite strong empirical predictions. It predicts that instructionally derived declarative knowledge will remain relatively unaffected by practice on tasks, and it predicts that procedural knowledge will become tuned for specific tasks, limiting its transferability to new tasks (the so-called "use specificity" effect; Anderson, 1987).

Bibby and Payne (1993) reported a series of experiments testing the adequacy of the knowledge compilation model in the context of learning to use a simple device. A priori, one might expect the enduring effect of instructionally derived declarative knowledge about a device to be compromised by interaction with that device, as we know that users of a device will themselves spontaneously generate accounts of the way it works (Payne, 1991; Shrager & Klahr, 1986). The device's behaviour during interactions offers a new source of information that may wash out any differences in initial instructions. Bibby and Payne manipulated the computational properties of informationally equivalent instructions, and devised device-based tasks that were relatively easier with different instructions. (The device and some of the instructions and tasks are reused in the current study, and thus are described later.) Bibby and Payne reported the findings:

1. Initial mental encodings of simple instructions share the computational properties of the external forms (They both facilitate the same inferences. Bibby and Payne, 1993, dubbed this the internalisation hypothesis.). Task A is easier than Task B with Instruction 1, but Task B is easier than Task A with Instruction 2, when instructions are being read, and when instructions have been remembered.
2. Procedural skills derived from these encodings continue to show effects of computational differences after considerable exposure to the device. The task by instruction interaction is unaffected by general use of the device.
3. These effects disappear with extended practice on the particular tasks through which they are revealed.
4. When practiced participants are introduced to novel tasks, the interaction between instruction and task reappears, suggesting that the initial declarative knowledge is unchanged, and still available as a basis for the compilation of new procedures.
These findings generally provide a high degree of support for the knowledge compilation model, in a potentially unfriendly domain. They are commensurate with the idea that device-using skills develop through the proceduralisation of instructionally provided declarative knowledge, and that subsequent learning results from the "tuning" of this use-specific procedural knowledge, leaving the initial declarative encodings of instructions unchanged.

However, the third finding we have noted is potentially a little troubling. If speed-up on tasks is due entirely to tuning of a procedural encoding, with no role for elaborated declarative knowledge, we would not expect the relative ease of different tasks to vary over time. The disappearance of a statistical interaction between tasks and instructions can only readily be explained, in the ACT* framework, as a statistical artefact, due to a floor-effect in task times.

Even if we are satisfied with this "explaining away" of one aspect of the Bibby and Payne (1993) data, clearly more work needs to be done before we can claim a full understanding of the role of instructions in the acquisition of device-use skills. We have exploited the general ideas in the ACT* theory, but we have not produced an explicit account of participants' mental representations, or of the dynamics of strategy shifts. These ambitions have been thwarted because the performance data that were collected do not sufficiently constrain the construction of explicit simulation models.

In this article, we report an attempt to replicate the first and third findings noted earlier, and to supplement the performance data by collecting offline verbal protocols. These enable us to question the constancy of participants' declarative device knowledge and to develop production system models of performance on one of the tasks, at two levels of experience.

The main aim of this research is to use the protocols and derived cognitive models to test the hypothesised use-specificity of participants' task strategies, and the extent to which changes in these strategies can be explained by procedural composition. A secondary aim is to use the same protocols to provide information about the internalisation hypothesis—to what extent does participants' knowledge about the device derive directly from the instructional materials?

2. THE EXPERIMENT

We conducted an experiment based on Bibby and Payne's (1993) Experiment 3. In this experiment, participants learn and practise three separate tasks on a simple device, having first learned about the device from one of two types of instructional material.

We planned to collect online think-aloud protocols from participants as they performed the tasks. However, in pilot studies the participants found it
impossible to verbalise informatively. As a compromise we collected detailed post-hoc protocols, by asking participants, immediately after they had finished the final task, to write a detailed instruction manual for novice users. The manual was to provide novice users with all the information they would need to perform the tasks, including "hints" for rapid performance.

Despite our focus on strategy change, experience was manipulated as a between-participants variable, with separate groups of novices and experts, defined by the amount of practice prior to writing the instruction manual. This design was used to avoid the possibility of learning being influenced by the writing of manuals.

2.1 The Device

The device was a simulated control panel with four switches and seven indicator lights (running on a BBC B microcomputer with a Z80-second processor, keyboard, and monochrome monitor). Pressing keys on the keyboard changed the position of the switches on the display, and the indicator lights "came on" (i.e., were filled with white pixels) if their associated components were working.

Figure 1 shows a picture of the device's display. This picture was given to participants as part of the instructional materials.

2.2 The Instructional Materials

The instructional materials were divided into three parts: a picture of the control panel (Figure 1), a general introduction to the device (Figure 2); a description of the device (a different description for each of two experimental groups).

The alternative descriptions of the device were as follows:

**Procedures:** A list of three procedures for setting switches to operate the device and fire the laser (cf. the procedural instructions of Kieras & Bovair [1984]; see Figure 3).
Imagine that you are on board a Klingon warship under attack from the Starship Enterprise for attempting to smuggle arms to the planet Orion III, your new allies. Your job is to operate the new laser weaponry developed using designs based on the phasers on board the Enterprise. The laser system has been designed so that it can be made to work when some of the components are broken.

The laser system comprises a power source (PS), two energy boosters (EB1 and EB2), accumulators (MA, SA1 and SA2) and the laser bank (LB).

Power is routed through the system by changing the position of switches directing the power from the power source on to one of the energy boosters then to one of the accumulators and finally an accumulator is selected to send power to the laser bank.

If a component is in working order then its indicator light will come on when it is receiving power.

**Figure 2. General introduction to the device.**

The sequences of switch positions enabling the laser to be fired are as follows:-

1. switch to on, switch to eb1, switch to ma
2. switch to on, switch to eb2, switch to sa1, switch to sa
3. switch to on, switch to eb2, switch to sa2, switch to sa

**Figure 3. Procedures.**

*Table:* A table listing the conditions under which each indicator light would be illuminated (see Figure 4).

The procedures were taken to be self-explanatory, but it was necessary to explain how to read the table. Participants were given two examples. First, starting on the top line of the table, it was explained that the indicator light PS comes on when the switch is changed to the on position. Second, it was
The following table shows the conditions under which each light will be illuminated if its associated component is in working order.

<table>
<thead>
<tr>
<th>LIGHT</th>
<th>SWITCH POSITION</th>
<th>LIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS</td>
<td>on</td>
<td>-</td>
</tr>
<tr>
<td>EB1</td>
<td>eb1</td>
<td>PS</td>
</tr>
<tr>
<td>EB2</td>
<td>eb2</td>
<td>on</td>
</tr>
<tr>
<td>MA</td>
<td>-</td>
<td>EB1</td>
</tr>
<tr>
<td>SA1</td>
<td>sa1</td>
<td>on</td>
</tr>
<tr>
<td>SA2</td>
<td>sa2</td>
<td>EB2</td>
</tr>
<tr>
<td>LB</td>
<td>ma</td>
<td>either</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or MA</td>
</tr>
<tr>
<td></td>
<td>sa</td>
<td>on SA1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>sa</td>
<td>on SA2</td>
</tr>
</tbody>
</table>

Figure 4. Table.

explained that the indicator light EB1 comes on when the switch is pointing to eb1 and the indicator light PS is on. Participants were asked to learn the table in such a way that they knew the conditions under which each indicator light could be expected to work if its associated component was not broken.

A diagrammatic representation of the internal topology of the device is given in Figure 5. This representation was not shown to the participants in the experiment and is here to clarify the reader's understanding of the device.

2.3 The Problem-Solving Tasks
The following three tasks were used in the experiment:

1. Fault finding (F). One of the components was broken and the participant's task was to work out which component was broken and type in the (abbreviated) name of that component. Only one of the components was broken at any time. Figure 6 shows an example of this task.

2. Switch setting (S). One of the switches would be in the wrong position for sending power through to the laser bank. The participant had to type in the name of the switch position that the misplaced switch needed to be pointing to so that power could be delivered to the laser bank. Only one of the switches needed to be changed. Figure 7 shows an example.
The correct answer is SA2.

3. Operating the device (O). The control panel started off with all the switches in the off position and the participants’ task was to try to get the laser to fire. They were told that they could change the position of the switches by pressing keys on the keyboard (they were given a chart showing which keys corresponded to which switch position that they were allowed to keep for the duration of the experiment). Sometimes it would be possible to fire the laser (success being indicated by the laser
Type in the change in switch position needed to make the laser fire

The correct answer is eb1

Figure 7. Example of the switch task.

bank indicator being "illuminated") and sometimes it would be impossible because too many components were broken or the laser bank itself was broken. Participants were told that they had to decide when to press "FIRE" or when to press "SHIELD." "FIRE" was pressed if the laser was working and "SHIELD" was pressed when the laser was not working, and this was described as a defence against enemy attack.

A message was displayed on the screen that indicated the correctness of a decision when firing the laser. They could change the position of as many switches as they liked in any order. There were an equal number of times when the laser could be fired and when it could not be fired. Figure 8 shows an example.

The tasks were designed to produce a performance cross-over with the instructions: We expected participants using the procedure instructions to find the switch task easier than the fault task and we expected participants using the table instructions to find the fault task easier than the switch task.

2.4 Procedure

The experiment began with a period of instruction. All participants were given a picture of the control panel (Figure 1—an exact copy of the device's display screen) that they could consult throughout this period. Each participant was introduced to the device by reading a general introduction (Figure 2), which explained a game scenario (cf. Kieras & Bovair’s fantasy condition) and gave general guidance about the configuration of the device.

Once participants had a general idea about the device, they progressed to one of two descriptions of the device. (Allocation of participants to experimental groups was done according to their order of arrival at the experimental laboratory.) They were asked to learn the description of the device
so that they could write it down without making mistakes. Participants were given a period of 5 minutes to learn the description of the device and were then given a paper-and-pen test to see if they could reproduce it accurately. If, when tested, they made any errors they were given an additional 5-min period in which to continue learning the materials. This learning and test cycle was repeated three times. If at the end of 15 minutes of learning time a participant was still unable to reproduce the device description, he or she was dismissed from the experiment.

Once participants had learned the materials, they were told about the tasks they were going to solve on the device. Each task was described in detail and a diagrammatic example of the task was given. Once the participants understood the nature of the tasks, they were introduced to the device and shown which keys to press on the keyboard to operate the device. The experiments differed in the number of examples of the tasks that the participants were expected to solve and which tasks were given.

In all these tasks, participants were asked to respond as quickly and accurately as possible. For the first block of each task, the participant was told by the experimenter whether he or she had solved the problem correctly. If the participant made a mistake he or she was asked to try and work out what the correct solution was. The experimenter did not help the participants other than to say when they had achieved the correct solution. After the first block they did not receive any information about the correctness of their solutions. Participants were not allowed to keep the instructional materials while solving the problems (in this aspect the procedure differed from that used by Bibby & Payne [1993], in which participants consulted the constructions during the first five trials of each task).

Throughout the experiments, key presses and timings were recorded automatically. Before analysis, timings for incorrect answers were discarded. Additionally, the first problem in each block was ignored to avoid the effects of undue interference from the previous problems. For the Operating task
the number of key presses was analysed alongside time and errors. Statistical analyses treated the Switch and Fault tasks, which required single responses, separately from the Operating task, which allowed interactive "playing" with the device.

As soon as each participant had completed his or her allotted quota of tasks, he or she was presented with pen and paper, and asked to write "an instruction manual for novice users of the system." Participants were asked to provide all the information needed to perform all the tasks, including any hints about strategies to make each task easier.

2.5 Participants
Twenty individuals, all of whom were first-year undergraduates at the University of Lancaster, were recruited from the psychology department's participant pool (average age of 18 years, 11 months). The participants were paid £2 per hour for their participation in the experiment. They were randomly allocated to the different conditions on their arrival at the experiment.

2.6 Design
The design comprised four groups of participants: two table and two procedure groups. Of the two groups that received the same instructions, one group had 20 examples of each of the tasks, and the other group had 80 examples of each of the tasks. The tasks were presented in blocks of five problems. The sequence of presentation of the tasks was FSOSOF (where F is the Fault-finding task, S is the switch-changing task, and O is the operating task).

2.7 Experimental Hypotheses
Regarding performance measures, we hypothesised replication of the results reported by Bibby and Payne (1993). Overall, participants will make very few errors on any of the tasks. Participants who received the table instructions will find the fault task easier than the switch task, but those who received the procedure instructions will find the switch task easier than the fault task. For those who are stopped after four blocks of each task (the novice users), this interaction effect will persist throughout the experiment. However, as participants practice the tasks through 16 blocks (the experienced users), a trial × task × instruction interaction will emerge, so that by the end of the experiment the effect of the different instructions on the relative ease of tasks will have disappeared.

If this pattern of results is obtained, the instruction manuals generated by participants (hereafter, the scripts) should provide clues about the different knowledge acquired and used in the different instructional conditions, how that knowledge affects strategies for the different tasks, and how these strategies change with experience.
What story do we expect to be able to tell? First, concerning participants' acquired declarative knowledge, we predict a simple effect of instructions on prime conceptual entities. The procedure instructions stress switches, whereas the table stresses components. We expect these biases between object types to be apparent in participants' descriptions of the device, according to which instructions they received. Furthermore, to the extent that declarative knowledge remains stable during the experiment, we expect any bias to be present in experienced users' as well as novice users' descriptions.

Second, with regard to task strategies, we expect the different instructional groups to begin with distinct strategies, explaining the observed performance cross-over. With experience, both groups' strategies will shift. The central experimental question is the extent to which these shifts can be modeled by application of procedural composition to the models of novice strategies.

3. RESULTS

3.1 Performance Data

We deal with the performance results in summary fashion because they reproduce results that were previously observed by Bibby and Payne (1993). We should note that, due to our focus on participants' verbal scripts, the number of individuals in each group is very small. Performance on the operating task will not be considered, as it plays no role in any of the experimental hypotheses, being included only as a part of the learning condition (general exposure to the device). Participants made too few errors overall on any task for any meaningful analysis.

All analyses of variance (ANOVAs) were conducted with the raw data and then repeated using a natural logarithm transformation to increase homogeneity of variance and to reduce skew. We report the analyses on transformed data, but all the significant effects in these analyses were also significant on the raw data.

We consider task performance times separately for novice and experienced users. For novice users, there was a significant main effect of instruction, $F(1, 8) = 6.21, MSE = 0.488, p < .0374$, a main effect of practice, $F(3, 24) = 46.32, MSE = 0.073, p < .0001$, and an interaction between task and instruction, $F(1, 8) = 57.103, MSE = 0.044, p < .0001$. Overall, the table group were significantly slower than the procedure group (table = 14 s, procedure = 9.49 s). With practice the time to solve the problems reduced (Block1 = 20.05 s, Block2 = 12.43 s, Block3 = 8.55 s, Block4 = 7.29 s). The interaction between task and instruction (see Figure 9) replicated the results found by Bibby and Payne (1993) with the table group taking longer to solve the switch-changing task than the fault-finding task (fault = 11.85 s, switch = 16.54 s) and the procedure group taking longer to solve the fault-finding task than the switch-changing task (fault = 11.46 s, switch = 7.86 s).
For experienced users there was a three-way interaction between task, instruction, and practice, $F(15, 120)=4.275$, $MSE=0.66$, $p<.0001$ (see Figure 10). Because this interaction captures all the other significant effects, it was decided to divide the data to look at the first four blocks of practice and perform separate analyses on each set of four blocks. For the first four blocks of practice, there was a significant main effect of practice, $F(3, 24)=19.165$, $MSE=0.102$, $p<.0001$, and an interaction between task and instruction, $F(1, 8)=58.103$, $MSE=0.044$, $p<.0001$. With practice, the time to solve the problems reduced (Block1 = 18.46 s, Block2 = 14.15 s, Block3 = 10.44 s, Block4 = 9.20 s). The interaction between task and instruction replicated the novice results with the table group taking longer to solve the switch-changing task than the fault-finding task (fault = 11.69 s, switch = 18.69 s) and the procedure group taking longer to solve the fault-finding task than the switch-changing task (fault = 11.78 s, switch = 9.75 s). For the last four blocks of practice there was only one effect, a main effect of practice, $F(3, 24)=4.759$, $MSE=0.061$, $p<.0096$ (Block1 = 5.76 s, Block2 = 5.39 s, Block3 = 4.76 s, Block4 = 4.41 s).
In summary, the pattern of results is as we hypothesised. We have replicated the findings of Bibby and Payne (1993, Experiment 3) despite the small sample size. The results of concern are the performance cross-overs between instructions and tasks. Such a cross-over was present for the novice users and for the first four blocks of problems for the experienced users. However, the interaction had disappeared by the final four blocks of the longer practice schedule. Now we can turn to participants' post-hoc scripts, to see what light they throw on this pattern of findings.

3.2 Analysis of the Scripts Produced After Interacting With the Device

All the general information about the device was collated and a number of different categories of information were identified:

1 The participants are coded according to which instruction group they belonged to and how much experience they had interacting with the device. For example, S1_Fn is the first participant in the procedure novice group.
1. Information referring to the original instructional materials. If, at any point in the instructions generated by the participants, a reference was made to the instructions the participants learned earlier in the experiment, either by mentioning them (e.g., suggesting that it is a good idea to know them) or by replicating them (e.g., drawing the power flow diagram, listing the procedures, or drawing the table), they were considered to view these instructions as useful (13 participants referred to the original instructional materials in this way).

S1_{TH}: "Having learned the table you will know that if the component and switch are working, the indicator lights up"

S1_{PN}: "The three routes available are:
1. switch ps on; switch to eb1; switch to ma.
2. switch ps on; switch to eb2; switch to sa1; switch to sa.
3. switch ps on; switch to eb2; switch to sa2; switch to sa.

2. A list of either all the components or all the switches, or both. Some participants mentioned just a few switches or components. This was usually done in the context of a particular problem, and was considered to be a separate type of information (two participants listed the components, two listed the switches, and three listed the components and the switches).

Two examples of this are:

S2_{TH}: "The laser device has several components:
- a power source
- 2 energy boosters (supplying energy to other components)
- 2 types of accumulator
  1. a main
  2. 2 secondary ones
and finally a laser beam operated by the two types of accumulator."

S1_{PE}: "The components are Switch for these are labeled

<table>
<thead>
<tr>
<th>Power source</th>
<th>PS</th>
<th>on/off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy Boosters</td>
<td>EB1</td>
<td>eb1</td>
</tr>
<tr>
<td>EB2</td>
<td></td>
<td>eb2</td>
</tr>
<tr>
<td>Main accumulator</td>
<td>MA</td>
<td></td>
</tr>
<tr>
<td>Secondary accumulators</td>
<td>SA1</td>
<td>sa1</td>
</tr>
<tr>
<td>SA2</td>
<td></td>
<td>sa2</td>
</tr>
<tr>
<td>Laser Bank</td>
<td>LB</td>
<td>sa/ma</td>
</tr>
</tbody>
</table>

3. Route information, that is, clearly stating what the three routes through the device were, either in terms of the components or the switches or both (six participants defined the routes in terms of the components, two defined the routes in terms of the switches, and two defined the routes in terms of both the components and the switches).
Two examples of this are:

\text{S4}_\text{Te}: \text{"Figure 1 shows the routes involved}

\text{\textbf{POWER SOURCE}}

\text{\textbf{BOOSTER 1}} \rightarrow \text{MA} \rightarrow \text{LB}

\text{\textbf{BOOSTER 2}} \rightarrow \text{SA1} \rightarrow \text{LB}

\text{\textbf{SA2}}

\text{S3}_\text{Te}: \text{"The two ways that it (the laser) can be made to work are as follows}

\text{a) PS} \rightarrow \text{EB1} \rightarrow \text{MA} \rightarrow \text{LB}

\text{b) PS} \rightarrow \text{EB2} \rightarrow \text{SA1/2} \rightarrow \text{LB"}

4. Information about the functions of the specific components (i.e., what the components did, what the indicator lights did, and what the switches did). Functional information specified what purpose each of the classes of objects in the device served, either by reference to the function of particular entities (e.g., by stating that the power source must be on for the device to work at all), or by describing the general purpose of a class of objects (four participants explained the function of the indicator lights, three explained the function of the switches, five explained the functions of the components and the indicator lights, seven explained the functions of the switches and the indicator lights, one explained the functions of the components, indicator lights, and the switches).

Three examples of this are:

\text{S3}_\text{Pe}: \text{"The way the power is switched from one component to another is by a system of switches, using these systems you can get the different pathways"}

\text{S4}_\text{Th}: \text{"A component is functioning properly if its light is on"}

\text{S5}_\text{Te}: \text{"......nothing will work if PS is off.}

\text{The EBs can only work if PS is giving them energy.}

\text{None of the accumulators can work without energy from the energy boosters.}

\text{The laser bank cannot work if it isn’t receiving energy from the accumulators."}

5. General operational principles (e.g., the principle of power flow through the system). Reference to general operational principles occurred
when participants stated that the goal of the operating task was to send
topower through the system, or to make reference to the need for power to be
sent through the system in order to test the systems functioning (one partici-
panant generated an analogy and seven used the idea of power flow).

Two examples of this are:

S5Pn: "I just imagine an electric current analogous to water flowing through a
series of taps (some of which get blocked) and pipes."

S3Pn: "The idea is for power to work through the system of switches until the
laser bank (LB) is on."

6. Reference to the interface, that is, which keys were pressed and what
the screen display looked like (five participants provided this kind of
information).

Two examples of this are:

S3Pn: "On the computer programme you can see which way the power is sup-
posed to be directed by the switches at the bottom of the screen."

S4Tn: "On the screen there will appear boxes representing components..."

None of the participants referred to all of these different aspects of the
device. In order to get some measure of the distribution of general device in-
formation contained in the scripts generated by the participants, a score was
produced for each participant that was a count of the number of different
categories of information that the participant referred to. These scores were
then subjected to an ANOVA.

The design for this ANOVA was 2 (amount of practice) × 2 (instructional
materials), where both amount of practice prior to writing instructions and
initial instructions were between-participant factors. One significant result
was obtained: a main effect for the amount of practice using the device
prior to writing instructions, $F(1, 16) = 4.777, p < .05$. The group who had
16 blocks of practice interacting with the device provided more varieties of
information about the device ($M = 2.73$) than the group who only had four
blocks of practice ($M = 1.60$).

### 3.6 Device Information:

**Distribution of References to Components and Switches**

For the ontology, route, and function information, a count was made of the
number of references made by the procedure and table groups that discussed
either the switches or the components. For example, if a participant
specified the ontology information in terms of the components, he or she
would score one point in the components categories. If he or she also
specified the ontology in terms of switches he or she would get one point in
the switches category. If he or she specified the route information only in
terms of the components, this would score one more point in the component category. Referring to the function of the switches only, and not the components, would receive an additional point in the switches category. This hypothetical participant would thus receive a score of two points in the components category and two points in the switches category. The scripts of the procedure and table groups were subjected to this scoring procedure.

The procedure and table groups were selected on the basis of the two strategies they might utilise and the consequent prejudices for particular classes of objects in the device ontology. With the procedure instructions, the switch task can be performed by checking the arrangement of switches against the three procedures in turn. One procedure will coincide with all but one of the switch settings; this is the one to change. For the Fault task, however, the list of procedures is not a great deal of help. Instead, participants in this group must rely on the additional information presented in the general instructions. Because there is no way a faulty component can be identified by directly reading from the list of procedures, this group has to work out how the components are connected together. Next, by checking each sequence consecutively, these participants can work out which component must be broken. These strategies rely on using the states of the switches as the primary source of information.

Participants with the table instructions will find it easier to solve the Fault task than the switch task. The table allows ready diagnosis of the non-working components. All that is required is that each component be considered one at a time and a check made on whether the conditions for it to be working are met. Once a component is found that meets the relevant conditions, but for which the indicator light is not illuminated, that is the component that is broken. On the other hand, the switch task is likely to provide far more problems. It is not clear from the table what sequences of switch positions deliver power to the laser bank, but by inspecting the alternative conditions in which LB is illuminated, and tracing back through the conditions for the antecedent components it is possible to work out sequences of switches for operating the device. Both these strategies use components as the primary source of information.

As we have suggested that participants may use these strategies, it seems likely that any instructions these participants generate will reflect the general biases for primarily considering the switches in the procedure group and the components in the table group. The ontology, route, and function information were the three categories of information that were thought most likely to demonstrate a difference between using switches or components as explanatory concepts. The original materials category was discarded on the basis of the inherent bias in the original instructional materials. The general operating principles were considered irrelevant because they are concepts that require a higher level understanding of this kind of device, separate
Figure 11. Interaction between instructional materials and component or switch references for the mean number of references made to switches and components.

from the particular device itself. Finally, the information about the interface was discarded because these tended to simply describe the display, which inevitably led to mentioning both the components and the switches.

The counts of references thus obtained were subjected to an ANOVA with a 2 (4 blocks practice/16 blocks practice) × 2 (instructional materials) × 2 (components/switches) design, where both amount of practice and instructional materials were between participant factors and components/switches a within-participant factor. Two significant results were obtained. Firstly, there was a main effect for the amount of practice prior to writing the instructions, $F(1, 16) = 4.654, p < .05$. The group with a large amount of practice ($M = 1.00$) produced more information than the group who had much less practice interacting with the device ($M = 0.45$). Second, there was an interaction between the initial instructional materials and the information generated by the participants referring to the switches or components, $F(1, 16) = 5.625, p < .005$ (see Figure 11).

An analysis of the simple effects showed that the procedure and table groups referred to the components as often as each other, but the procedure group referred to the switches more often than the table group. At the same time, the procedure group referred to the switches significantly more often than the components, whereas the table groups referred to the components significantly more often than the switches.
3.7 Task Information: General Information

About the Strategies for Solving the Fault-Finding Task

Further examination of the scripts showed that the participants described four different general strategies used to solve the Fault-finding task. (With regard to the Switch-changing task, it was not possible to generate any general strategies from the descriptions provided in the scripts. The information was either so vague as to be unusable, or so specific to particular situations it was impossible to generalise.) The strategies are given in the following.

1. Look at each component indicator light in turn, working from the left of the screen to the right, and check to see if the appropriate conditions for illumination exist. If those conditions are met and the indicator is not lit, then that is the faulty component.

Two examples of this strategy are:

S1_Tn: "Take methodical steps and go through each component one by one in the order they appear on the screen, checking to see if it should be lit."

S2_Tn: "Look at the components from left to right and work out which light should be on and isn’t."

2. Look at each switch in turn, working from the left to the right, and check to see that the component that is receiving power is working. It is working if the associated indicator light is illuminated.

Two examples of this strategy are:

S1_Pn: "Determine which component is broken by looking at the switches and appropriate lights (only one component is broken)."

S5_Pn: "Look to see which component is switched on and check to see if the light is lit up; if not this is the selection you make. Go through one by one checking to see if lit."

3. Use perceptual cues, available on the interface display, to reduce the amount of search needed to establish which component is broken. First, look to see which is the last indicator on the right that is illuminated, and then check against the following list:

(a) If no lights are lit then the power source is faulty.
(b) If the power source is lit then check to see which of the energy boosters has been selected. The energy booster that has been selected is the faulty component.
(c) If energy booster one is lit then the main accumulator is faulty.
(d) If energy booster two is lit then one of the secondary accumulators is faulty. Check the secondary accumulator switch to decide which has been selected and is consequently broken.
(e) If either the main accumulator or one of the secondary accumulators is lit, then the laser bank is faulty.
Two examples of this strategy are

S2Te: "For the fault-finding task, select the component after the last one that is lit (i.e., indicated working) and is selected on the switch and that is the broken component."

S5Te: "Looking at the last one lit

If Power switch set to on and PS not on the fault = PS
If PS on alone then fault = EB1 or EB2 depending on whether switch is set to eb1 or eb2.
If EB1 on but not MA then fault = MA.
If EB2 on but not SA1 or SA2 then fault = SA1 or SA2 depending on whether switch set to sal or sa2.
If MA on the fault = LB.
If SA1 or SA2 on then fault = LB."

4. Use information about different routes through which the power must flow to reduce the amount of search when deciding which of the components is faulty. Check which sequence of switches are selected and only consider those components that are part of that sequence of switch positions.

Two examples of this strategy follow:

S3Pe: "If power is flowing into a component and its light is unlit then it is broken. If the light is lit move on to the next component through which the current flows and test it until one is broken."

S2Pe: "Go through the switches in the order in which the power flows through them and the first one which should be alight and is not is the faulty component (e.g., PS -> EB1 -> MA -> LB) is one way the power should flow."

Although these examples are quite clear, five of the instructions generated by the participants were ambiguous about which strategy was being used. However, it is worth noting that this ambiguity only applied with regard to strategies 1 and 2 or strategies 3 and 4. There was no case in which strategy 1 could be confused with either strategy 3 or 4 and vice versa. Similarly, strategy 2 could not be confused with either strategy 3 or 4 and vice versa. Participants either used a strategy that reduced the amount of search (i.e., 3 or 4) or they used a more mechanical strategy (i.e., 1 or 2).

Using the Fisher Exact probability test, we tested two hypotheses on the basis of the data abstracted from the descriptions of the Fault-finding task. First, we looked at whether or not the strategies that were generated differed according to the level of practice that participants had received. It seems that there are two basic categories that can be applied to these strategies. One kind of strategy is quite mechanical in its application, whereas the other more sophisticated strategy reduces the amount of search required to
solve the problems. The number of participants who used a mechanical strategy in the two practice conditions were counted \((S_4 \text{ BLOCKS} = 7, S_{16} \text{ BLOCKS} = 0)\); also, the number of participants who used the strategies that reduced the amount of search were counted \((S_4 \text{ BLOCKS} = 1, S_{16} \text{ BLOCKS} = 9)\). All these categories of participants include some strategies that were ambiguous. However, because this ambiguity did not cross the boundary drawn by the distinction between mechanical and search-reducing strategies, this seemed a reasonable manipulation. A \(2 \times 2\) matrix was generated and the Fisher Exact test applied. The null hypothesis that there would be no difference between the cells in the matrix was rejected because under that assumption the probability of obtaining the particular distribution obtained here was \(p < .005\). Thus, we have clear evidence that with the increase in practice, there is a shift away from the mechanical strategies to the more sophisticated search-reducing strategies.

A second hypothesis based on the different strategies that different instructional materials were likely to generate was also tested. We argued earlier that the procedure group was likely to use a strategy that was based on the positioning of the switches. Both strategies 2 and 4 share this property. On the other hand, the table group was likely to use a strategy that relied more on the components or the indicators lights. Both strategies 1 and 3 share this property. Therefore, it was decided to look at the pairings of strategies 2 and 4 and 1 and 3. Strategies 2 and 4 are known as switch strategies and 1 and 3 are known as component strategies. The number of participants in the procedure group who used either a switch strategy or a component strategy were counted \((S_{\text{switch}} = 4, S_{\text{component}} = 0)\). Similarly, the number of participants in the table group who used either a switch strategy or a component strategy were counted \((S_{\text{switch}} = 2, S_{\text{component}} = 5)\). The Fisher Exact test was applied to a \(2 \times 2\) matrix thus constructed. The null hypothesis that there would be no difference in the distribution of the switch and component strategies between the procedure and the table groups was rejected because the probability of observing the present distribution was \(p < .005\). Hence, the table group is more likely to use a strategy based on the components, whereas the procedure group is more likely to use a strategy based on the switches.

4. DISCUSSION

Because the analysis of the timing data from the experiment concurred with the results obtained by Bibby and Payne (1993), we raise only two aspects of the data. First, consider the disappearance of the Task by Instruction interaction after substantial practice. As noted in the introduction, we could dismiss this as due to a floor effect. However, we are persuaded to take the phenomenon more seriously. There is no direct evidence of a "floor" having
been reached. In many studies of complex skill acquisition, performance speeds continue to increase over much larger numbers of trials than those studied in this experiment (Newell & Rosenbloom, 1981). Furthermore, informally, the first author (after extensive practice) can perform the same tasks very much quicker than any of the participants in this study. Taking the disappearance of the interaction seriously suggests the possibility that, with practice, participants’ strategies change, rather than merely becoming faster. The data from the scripts allow us to examine this possibility in detail.

Second, note that time and error performance on the first four blocks of trials show the same pattern for both novice and expert groups of participants. This supports our use of between-group comparisons to index changes due to learning. We assume that scripts generated by novices are reliable indicators of knowledge of experts at that same stage of learning.

We now turn to the results obtained from the analysis of the scripts generated by the participants. The quantitative patterns in the analysis of the scripts clearly fall into two categories: the effect of the increase in practice on the strategies that participants report for interacting with the device, and the effects of the different instructions on the users’ behaviour.

The quantity and variety of general device information that users generated depended on the amount of practice the users had interacting with the device. Users with extended practice gave more types of general information than users with only a small amount of practice. This tendency to provide more information extended to the quantity of specific strategic information about the tasks that users generated.

The strategies that are reported for solving the problems also differ according to the amount of practice the user has interacting with the device. When users have a small amount of practice solving the fault-finding task, they describe strategies that are mechanical in their application. There is a tendency to check all the possibilities in a sequential manner until a broken component is identified. The users who had more practice on this task described strategies that required far less work. They developed ways of restricting the amount of search that was necessary to isolate the broken component. Thus, there is a shift away from strategies that require a lot of work toward more efficient strategies. This shift mirrors the well-documented shift from weak to heuristic search (Anzai & Simon, 1979; Langley, 1987; Larkin, 1981; Newell & Simon, 1972; and many others), and thus gives some intuitive support to the post-hoc protocol methodology used in this study.

However, when we look at the results concerning the specific effects of the instructional materials, the picture becomes more complex. For the general information about the device, there is an interaction between the instructional materials and references to components or switches. The procedure group was more likely to refer to the switches than the components
when introducing the device, whereas the table group was more likely to refer to the components than the switches.

The descriptions of the strategies followed a similar pattern. The procedure group with a small amount of practice described a strategy that relied on looking at the position of the switches and checking to see if the associated component was working. The table group in this practice condition described a strategy that relied on looking at the components one at a time and checking to see if that component’s indicator light should be illuminated. The users with an increased amount of practice could also be separated according to which instruction group they belonged. The procedure group described a strategy that reduced the amount of search by using route information. It was no longer necessary to check all possibilities, rather only those that followed a particular sequence needed to be checked. The table group, on the other hand, used information about which indicator lights should be lit given particular antecedent conditions. This strategy also reduced the amount of search necessary to identify a faulty component, but did so in a different way.

It seems that the change from mechanical, inefficient strategies to search-reducing strategies does not solely depend on simply having more practice on this task, but it also depends on what the initial strategy is. This in turn depends on what instructions are made available to the user prior to interacting with the device.

It is interesting to note the lack of information generated about the Switch-changing task. There were no general strategies described for solving this task, and the specific information about the tasks abstracted from the scripts showed that all the users were more likely to give information about the Fault-finding task than the Switch-changing task. It is difficult to see why this should be the case, unless users do not develop strategies but rather learn particular configurations of switch positions, which would inevitably be more difficult to describe.

These results can be divided according to their implications for declarative knowledge about the device or for task-specific strategies. The scripts provided two kinds of information that support the suggestion that the table and procedure instruction groups have different device ontologies. The first piece of evidence is that the two groups show a preference for describing the device either in terms of the switches that direct power through the system, or the components through which the power flows. The second comes from the information they provide about the strategies used to solve the fault-finding task, with the procedure group depending primarily on the position of the switches and the table group depending on the component indicator lights. It seems likely that the different instructions have led the groups to consider the device from different viewpoints. In other words, the conceptual entities that form their representation of the device
are different. The table group seems to view the components as the dominant objects in the device ontology, with switch positions as attributes of components. On the other hand, the procedure group views the switches as the dominant objects, and the components are thus considered attributes of a particular switch position.

The findings we have just summarized may challenge the knowledge compilation account of skill acquisition. According to ACT*, participants' scripts must reflect their declarative knowledge—those aspects of the scripts that describe strategies reflect declarative knowledge of problem-solving procedures. The fact that participants report different strategies at different levels of experience indicates that their learning could not just be knowledge compilation as knowledge compilation on its own leaves declarative knowledge unchanged.

We can see three kinds of explanation that might address the observed shift in the scripts. The least interesting and least plausible, it seems to us, is to regard the reported declarative encodings of strategies as essentially independent of procedural encodings, and thus changes in those encodings as being outside the scope of the knowledge compilation theory. From this standpoint, some learning mechanisms may be affecting participants' scripts, but those mechanisms have nothing to do with the acquisition of skill in manipulating the device. This is implausible because it seems unlikely that as participants practice problem solving they could learn to report more efficient strategies without being able to use those strategies in their problem solving. Consequently, we now raise further possibilities that depend on the assumption that the strategies described in participants' scripts do indeed reflect the strategies embodied in their procedural knowledge.

The second possibility (K. Van Lehn, 1984, personal communication) is that participants acquire new declarative knowledge by deliberate reflection on their interactions with the device, and then proceduralise this new declarative knowledge by standard knowledge compilation processes. This possibility contradicts the ACT* account in that declarative learning mechanisms directly affect the ongoing dynamics of procedural skill acquisition.

The third possibility is that ACT* is basically correct. Knowledge compilation produces new procedural encodings, and then individuals use these to generate their scripts in the post-hoc protocols by mentally simulating problem-solving behaviour and reporting the working memory transactions just as they would in concurrent verbalisation. For this explanation to be accepted, we would need to demonstrate that the specific mechanisms of knowledge compilation, acting on the observed novice procedures, can generate the observed expert procedures. In fact, however, there is a simple argument that they cannot, given the following three steps:

1 For a clear formulation of this argument we are indebted to Kurt Van Lehn.
1. Compilation preserves all motor and perceptual acts—it reduces only the internal manipulations of working memory.

2. Novice table participants' strategy (see Number 1) looks at both lights and switches, starting from the left, whereas expert table participants' strategy (see Number 3) looks only at the rightmost illuminated light and a single switch. There are fewer perceptual acts in the expert strategy.

3. Therefore knowledge compilation cannot generate the expert strategy.

This argument is extended by detailed production system modeling shown later. The production system modeling also addresses the obvious challenge: If knowledge compilation alone cannot explain the observed strategy changes, what additional mechanisms are required?

5. MODELING STRATEGY CHANGE

Two sets of productions have been developed that model the behaviour of the novice procedure and table users' strategies. The knowledge compilation processes of composition and proceduralization have been applied to these productions and the outcomes have been compared to the strategies suggested by the more experienced users.

Deriving production system models from verbal protocols always requires an inductive leap. The problem is particularly acute in this case, as the protocols are post-hoc, and therefore rather incomplete, and as we are trying to model strategies common to a group of five participants, rather than a single person's behaviour. Our approach was the usual one: We cannot pretend that the production systems are determined by the protocols, but we can endeavour to ensure that the production systems are sufficient to explain the protocols. We do not expect participants to have conscious access to procedural knowledge, so the important criterion is that the productions must generate sequences of steps that concur with the participants' reported steps. An additional constraint on the novice production system models is that they must only require access to the knowledge that is encoded in the appropriate set of instructions.

Members of the novice procedure (N-PROC) group considered each of the switches in turn. Each switch position has at least one associated component (e.g., "on" directs power to the Power Source indicator light; "sa1" directs power to Secondary Accumulator 1). In the case of the "eb1" switch position, both Energy Booster 1 and the Main Accumulator take power when that switch position is selected. N-PROC participants checked to see what position a switch was pointing to and then checked if the associated component was working. If the component was not working, it was necessary to check that it was receiving power. This could be achieved by checking if the
preceding component in the sequence is working. If the associated component is receiving power and it is not working, that is the faulty component. If the component is not receiving power, or it is working, then it cannot be broken. The following productions solve the fault-finding task in a way that is consistent with strategies described by N-PROC participants.

PN-PROC 1: IF the goal is to find a fault THEN set a subgoal to check first switch.
PN-PROC 2: IF the goal is to find a fault AND a switch is checked THEN set a subgoal to check next switch.
PN-PROC 3: IF the goal is to check a switch THEN determine the switch position AND retrieve the associated component(s) AND set a subgoal to check the associated component(s).
PN-PROC 4: IF the goal is to check a switch AND associated component is working THEN POP the goal AND tag switch as checked.
PN-PROC 5: IF the goal is to check a switch AND associated component is not receiving power THEN POP the goal AND tag switch as checked.
PN-PROC 6: IF the goal is to check associated component THEN set a subgoal to check associated component is receiving power AND set a subgoal to determine the state of associated component indicator light.
PN-PROC 7: IF the goal is to check associated component AND associated component is not working AND associated component is receiving power THEN give the name of associated component as the fault AND STOP.
PN-PROC 8: IF the goal is to check associated component AND there is no preceding component AND associated component is not working THEN give the name of associated component as the fault AND STOP.
PN-PROC 9: IF the goal is to check associated component is receiving power THEN set a subgoal to check preceding component is working.
PN-PROC 10: IF the goal is to check associated component is receiving power AND there is no preceding component THEN POP the goal.
PN-PROC 11: IF the goal is to check associated component is receiving power AND preceding component is working THEN POP the goal AND tag associated component as receiving power.
PN-PROC 12: IF the goal is to check associated component is receiving power AND the preceding component is not working THEN POP two goals AND tag the associated component as not receiving power.
PN-PROC 13: IF the goal is to check if preceding component is working THEN retrieve preceding component.
PN-PROC 14: IF the goal is to check the preceding component is working AND the preceding component is retrieved THEN POP the goal AND set a subgoal to determine the state of preceding component indicator light.
PN-PROC 15: IF the goal is to check the preceding component is working AND there is no preceding component retrieved THEN POP the goal AND tag as no preceding component.

PN-PROC 16: IF the goal is to determine the state of associated component indicator light AND associated component indicator light is lit THEN POP the goal AND tag associated component as working.

PN-PROC 17: IF the goal is to determine the state of associated component indicator light AND associated component indicator light is not lit THEN POP the goal AND tag associated component as not working.

PN-PROC 18: IF the goal is to determine the state of the preceding component indicator light AND the preceding component indicator light is lit THEN POP the goal and tag the preceding component as working.

PN-PROC 19: IF the goal is to determine the state of the preceding component indicator light AND the preceding component indicator light is not lit THEN POP the goal AND tag the preceding component as not working.

The initial goal is to find a fault. The initial action taken is to check the first switch in the sequence (PN-PROC 1). In order to use this information, participants must retrieve further information (i.e., the identity of the associated component), and once this is done they can then proceed to check that component (PN-PROC 3). Checking the associated component entails that the indicator light is checked and that a check is made to see if the associated component is receiving power (PN-PROC 6). If the associated component’s indicator light is lit, the component is tagged as working and there is no further need to check whether that component is receiving power, or to further check the component (PN-PROC 16). Consequently, the switch can be tagged as checked (PN-PROC 4) and the next switch can be examined (PN-PROC 2). If the associated component’s indicator light is not lit, the component is tagged as not working (PN-PROC 17). In order to check if a component is receiving power, the preceding component in the procedure needs to be working (PN-PROC 9). This requires that the identity of the preceding component is retrieved from long-term memory (PN-PROC 13), and its indicator light is checked to see if it is lit (PN-PROC 14). Sometimes there is no preceding component retrieved, and it is tagged as such (PN-PROC 15). If the preceding component’s indicator light is lit, it is tagged as working (PN-PROC 18). If it is not lit, it is tagged as not working (PN-PROC 19). If there is no preceding component, the associated component cannot be receiving power, so there is no further need to consider this point (PN-PROC 10). If the preceding component is working, the associated component must be receiving power (PN-PROC 11), but if the preceding component is not working, the associated component will not receive power and there is no need to con-
Consider this component any longer (PN-PROC 12). In the case where a component is not receiving power, the switch can be tagged as checked (PN-PROC 5). Two productions can now be applied that will identify the fault: If there is no preceding component and the associated component is not working, the associated component must be at fault (PN-PROC 8). On the other hand, if the associated component is not working, but it is receiving power then it must be broken (PN-PROC 7).

Participants in the novice table group (N-TAB) looked at each component indicator light in turn and checked to see if it was working. If the component was working, they moved on to the next component until they reached a component that was not working. Not working, however, does not always mean that the component was broken, so they had to check the conditions of illumination. If either the switch condition or the light condition was not satisfied, they moved on to the next component. If both the switch condition and the light condition were satisfied when the component was not working, this component must be faulty. The following productions solve the fault-finding task in the manner described by N-TAB participants.

PN-TAB 1: IF the goal is to find fault THEN set a subgoal to check first component.
PN-TAB 2: IF the goal is to find fault AND a component has been checked THEN set a subgoal to check next component.
PN-TAB 3: IF the goal is to check component THEN set a subgoal to determine the state of the component indicator light.
PN-TAB 4: IF the goal is to check component AND component is working THEN POP goal AND tag component as checked.
PN-TAB 5: IF the goal is to check component AND component is not working THEN set a subgoal to check conditions of illumination.
PN-TAB 6: IF the goal is to check component AND component is not working AND conditions of illumination are satisfied THEN give the name of component as the fault AND STOP.
PN-TAB 7: IF the goal is to check component AND conditions of illumination are not satisfied THEN POP the goal AND tag component as checked.
PN-TAB 8: IF the goal is to determine the state of component indicator light AND component indicator light is lit THEN POP the goal AND tag component as working.
PN-TAB 9: IF the goal is to determine the state of component indicator light AND component indicator light is not lit THEN POP the goal AND tag component as not working.
PN-TAB 10: IF the goal is to check conditions of illumination THEN retrieve light condition AND retrieve switch condition AND tag the conditions of illumination as retrieved.
PN-TAB 11: IF the goal is to check the conditions of illumination AND the conditions of illumination are retrieved THEN set a subgoal to determine the state of retrieved light condition AND set a subgoal to determine the state of retrieved switch condition.
INSTRUCTION AND PRACTICE

PN-TAB 12: IF the goal is to check conditions of illumination AND retrieved switch condition is satisfied AND retrieved light condition is satisfied THEN POP the goal AND tag conditions of illumination as satisfied.

PN-TAB 13: IF the goal is to check conditions of illumination AND there is no retrieved switch condition THEN set a goal to determine the state of retrieved light condition AND tag retrieved switch condition as satisfied.

PN-TAB 14: IF the goal is to check conditions of illumination AND there is no retrieved light condition THEN set a goal to determine the state of retrieved switch condition AND tag retrieved light condition as satisfied.

PN-TAB 15: IF the goal is to check conditions of illumination AND retrieved switch condition is not satisfied THEN POP goal AND tag retrieved conditions of illumination as not satisfied.

PN-TAB 16: IF the goal is to check conditions of illumination AND retrieved light condition is not satisfied THEN POP goal AND tag retrieved conditions of illumination as not satisfied.

PN-TAB 17: IF the goal is to determine the state of retrieved switch condition AND retrieved switch condition is selected THEN POP the goal AND tag retrieved switch condition as satisfied.

PN-TAB 18: IF the goal is to determine the state of retrieved switch condition AND retrieved switch condition is not selected THEN POP the goal AND tag retrieved switch condition as not satisfied.

PN-TAB 19: IF the goal is to determine the state of light condition AND retrieved light condition is lit THEN POP the goal AND tag retrieved light condition as satisfied.

PN-TAB 20: IF the goal is to determine the state of retrieved light condition AND retrieved light condition is not lit THEN POP the goal AND tag retrieved light condition as not satisfied.

The initial goal is to find a fault. In order to find a fault, the system starts by checking the first component (PN-TAB 1). Checking the first component requires that the state of the indicator light is determined (PN-TAB 3). If the indicator light is lit, the component is tagged as working (PN-TAB 8). If it is not lit, the component is tagged as not working (PN-TAB 9). When the component is working, it is tagged as checked (PN-TAB 4), and the next production selected will cause the system to look at the next component on the screen (PN-TAB 2). If the component is not working, the conditions of illumination need to be checked (PN-TAB 5). First the conditions of illumination have to be retrieved (PN-TAB 10), and then the switch condition and the light condition have to be examined (PN-TAB 11). When the conditions of illumination need checking, it is possible that only one condition exists, either a switch condition or a light condition. If there is no switch condition retrieved, the switch condition is tagged as satisfied, and a goal is set to check the light condition (PN-TAB 13). If no light condition is retrieved, the light condition is tagged as satisfied, and a goal is set to check the switch condition (PN-TAB 16).
14). If the switch condition specified under the conditions of illumination is selected then the switch condition is tagged as satisfied (PN-TAB 17). On the other hand, if the switch condition is not selected, there is no need to check the light condition; consequently, that goal is removed from the stack and the switch condition is tagged as not satisfied (PN-TAB 18). If the component indicator light specified by the conditions of illumination is lit, the light condition is tagged as satisfied (PN-TAB 19). If it is not lit, the light condition is tagged as not satisfied (PN-TAB 20). If the switch condition is not satisfied, the conditions of illumination are tagged as not satisfied (PN-TAB 15). Similarly, if the light condition is not satisfied, the conditions of illumination are tagged as not satisfied (PN-TAB 16). If both the light condition and the switch condition are satisfied, the conditions of illumination are tagged as satisfied (PN-TAB 12). When the conditions of illumination are satisfied, and a component is not working, that component can be identified as the fault (PN-TAB 7). If the conditions of illumination are not satisfied, the component is tagged as checked, and once again the system will move onto checking the next component on the screen (PN-TAB 2).

With these models of novice strategies made explicit, we can apply the precise learning mechanisms of knowledge compilation, and then compare the compiled productions with the strategies reported by experienced participants. Proceduralization embeds the information retrieved from long-term memory or the external environment in the productions. Composition takes sequences of productions and compacts them into single productions that have the same effect as the sequence.

For the procedure group, the following productions are the result of knowledge compilation applied to the initial productions (CPN-PROC stands for compiled production for novice procedure group).

\[
\text{CPN-PROC-PS: } \text{IF on is selected AND PS indicator light in not lit THEN give PS as the fault.}
\]

\[
\text{CPN-PROC-EB1: } \text{IF on is selected AND PS indicator light is lit AND eb1 is selected AND EB1 indicator light is not lit THEN give EB1 as the fault.}
\]

---

1 The productions were implemented in OPS5 and traces were produced of the production firing for both procedure and the table production for all the examples of the fault-finding task. Both proceduralization and composition were mimicked by hand (see Neves & Anderson, 1981). In the case of proceduralization, whenever a variable was replaced by a value a new production was created that included that variable instantiation. For composition, the production traces were examined to find pairs of productions that systematically followed from each other. Once these pairs were identified, new composed productions were created. These new sets of productions were applied to the fault-finding tasks and proceduralization and composition were once again mimicked by hand. Thus there was an iterative change in the set of productions until single productions were created that could solve the tasks.
CPN-PROC-EB2: IF on is selected AND PS indicator light is lit AND eb2 is selected AND EB2 indicator light is not lit THEN give EB2 as the fault.

CPN-PROC-MA: IF on is selected AND PS indicator light is lit AND eb1 is selected AND EB1 indicator light is lit AND MA indicator light is not lit THEN give MA as the fault.

CPN-PROC-SA1: IF on is selected AND PS indicator light is lit AND eb2 is selected AND EB2 indicator light is lit AND sa1 is selected AND SA1 indicator light is not lit THEN give SA1 as the fault.

CPN-PROC-SA2: IF on is selected AND PS indicator light is lit AND eb2 is selected AND EB2 indicator light is lit AND sa2 is selected AND SA2 indicator light is not lit THEN give SA2 as the fault.

CPN-PROC-LBm: IF on is selected AND PS indicator light is lit AND eb1 is selected AND EB1 indicator light is lit AND ma is selected AND LB indicator light is not lit THEN give LB as the fault.

CPN-PROC-LBsl: IF on is selected AND PS indicator light is lit AND eb2 is selected AND EB2 indicator light is lit AND sal is selected AND SAl indicator light is lit AND sa is selected AND LB indicator light is not lit THEN give LB as the fault.

Let us now turn to the productions that model the novice table users. Applying composition and proceduralization generates the following productions:

CPN-TAB-PS: IF PS indicator light is not lit AND on is selected THEN PS is the fault.

CPN-TAB-EB1: IF PS indicator light is lit AND EB1 indicator light is not lit AND eb1 is selected THEN give EB1 as the fault.

CPN-TAB-EB2: IF PS indicator light is lit AND EB1 indicator light is not lit AND eb1 is not selected AND EB2 indicator light is not lit AND eb2 is selected THEN give EB2 as the fault.

CPN-TAB-MA: IF PS indicator light is lit AND EB1 indicator light is lit AND EB2 indicator light is not lit AND eb2 is not selected AND MA indicator light is not lit THEN MA is the fault.

CPN-TAB-SA1: IF PS indicator light is lit AND EB1 indicator light is not lit AND eb1 is not selected AND EB2 indicator light is not lit AND ma is selected THEN SA1 is the fault.

CPN-TAB-SA2: IF PS indicator light is lit AND EB1 indicator light is not lit AND eb1 is not selected and EB2 indicator light is lit and
MA indicator light is not lit AND SA1 indicator light is not lit AND sa1 is not selected and SA2 indicator light is not lit AND sa2 is selected THEN SA2 is the fault.

CPN-TAB-LBm: IF PS indicator light is lit AND EB1 indicator light is lit AND EB2 indicator light is not lit AND eb2 is not selected AND MA indicator light is lit AND SA1 indicator light is not lit AND sa1 is not selected AND SA2 indicator light is not lit AND sa2 is not selected AND LB indicator light is not lit AND ma is selected THEN LB is the fault.

CPN-TAB-LBs1: IF PS indicator light is lit AND EB1 indicator light is not lit AND eb1 is not selected AND MA indicator light is not lit AND SA1 indicator light is lit AND SA2 indicator light is not lit AND sa2 is not selected AND LB indicator light is not lit AND sa is selected THEN LB is the fault.

CPN-TAB-LBs2: IF PS indicator light is lit AND EB1 indicator light is not lit AND eb1 is not selected AND EB2 indicator light is lit AND MA indicator light is not lit AND SA1 indicator light is not lit AND sa1 is not selected AND SA2 indicator light is not lit AND LB indicator light is not lit AND sa is selected THEN LB is the fault.

In order to check if knowledge compilation is an appropriate means of modelling the kind of learning that has taken place during the development of task-specific strategies for solving the fault-finding task, the productions that have been generated through the compilation of the productions that modelled the novice table and procedure users should now be compared with the strategy descriptions provided by the experienced table and procedure users. The experienced procedure users described a strategy that required them to consider only those components that were on one of the three sequences of switches that constituted the procedures. The productions generated from the novice procedure users when knowledge compilation has been applied share this property. For example:

CPN-PROC-SA1: IF on is selected AND PS indicator light is lit AND eb2 is selected AND EB2 indicator light is lit AND sa1 is selected AND SA1 indicator light is not lit THEN give SA1 as the fault.

This generates the correct solution by considering only those lights that are relevant to the switch positions that correspond to one of the three operating procedures. No reference is made to the components that are not on this particular sequence of switches. In this case, it seems apparent that knowledge compilation provides an adequate model of the process of strategy change. Compiling the productions that model the novice procedure users' reported strategies generates productions that mimic the experienced pro-
procedure users' reported strategies. This provides support for the knowledge compilation model of learning that goes beyond the performance data.

Unfortunately, compilation of the productions that model the novice table users does not generate a satisfactory set of productions that describe the behaviour of the experienced table users. To see why, we must first develop an adequate production system model of the strategy reported by experienced table users. Consider the following set of productions that embody the strategy employed by the experienced table users:

Pet-PS: IF PS indicator light is not lit THEN give PS as the fault.
Pet-EB1: IF PS is the rightmost indicator light lit AND eb1 is selected THEN give EB1 as the fault.
Pet-EB2: IF PS is the rightmost indicator light lit AND eb2 is selected THEN give EB2 as the fault.
Pet-MA: IF EB1 is the rightmost indicator light lit THEN give MA as the fault.
Pet-SA1: IF EB2 is the rightmost indicator light lit AND sa1 is selected THEN give SA1 as the fault.
Pet-SA2: IF EB2 is the rightmost indicator light lit AND sa2 is selected THEN give SA2 as the fault.
Pet-LBm: IF MA is the rightmost indicator light lit THEN give LB as the fault.
Pet-LBs1: IF SA1 is the rightmost indicator light lit THEN give LB as the fault.
Pet-LBs2: IF SA2 is the rightmost indicator light lit THEN give LB as the fault.

In this set of productions the requirement to look at the rightmost indicator light that is lit refers to the observation that the experienced table users looked to see which was the last light to the right that was lit.

Clearly, these productions are not the same as the productions generated by compiling the productions that model the novice table users. For example:

CPN-TAB-SA1: IF PS indicator light is lit AND EB1 indicator light is not lit AND eb1 is not selected AND EB2 indicator light is lit AND MA indicator light is not lit AND SA1 is not lit AND sa1 is selected THEN SA1 is the fault.

requires a lot more work to be done than,

Pet-SA1: IF EB2 is the rightmost indicator light lit AND sa1 is selected THEN give SA1 as fault.

Indeed, the productions that model the experienced table users have removed the condition redundancies that exist in the compiled novice table productions. Condition redundancy in the compiled productions becomes obvious if we consider the different routes that the power can take to flow through the device:


The power must always flow through the system according to one of these routes. This means that if power is moving through a particular route, checking the states of any components that are not on that route is an unnecessary action. Thus, in CPN-TAB-SA1, checking the state of Energy Booster One is unnecessary as Energy Booster Two is already on. This applies equally to the requirement in CPN-TAB-SA1 that "eb1 is not selected." The switch position "eb1" will never be selected if Energy Booster Two is working.

If the condition redundancies are removed from CPN-TAB-SA1, the following production is obtained:

CPN-TAB-SA1(ed): If PS indicator light is lit AND EB2 indicator light is lit AND SA1 is not lit AND sa1 is selected THEN SA1 is the fault.

This production is far more efficient and could be considered as a variant of Pet-SA1, as it establishes the identity of the last indicator light to the right that is lit. The difference is due to the fact that the productions that mimic the experienced table users only encode the conditions specified by those users in their descriptions. The following productions are derived by editing out the condition redundancies from the compiled table productions:

CPN-TAB-LBS1(ed): IF PS indicator light is not lit THEN give PS as the fault.
CPN-TAB-EB1(ed): IF PS indicator light is lit AND EB1 indicator light is not lit AND eb1 is selected THEN EB1 is the fault.
CPN-TAB-EB2(ed): IF PS indicator light is lit AND EB2 indicator light is not lit AND eb2 is selected THEN EB2 is the fault.
CPN-TAB-MA(ed): IF PS indicator light is lit AND EB1 indicator light is lit AND MA indicator light is not lit THEN MA is the fault.
CPN-TAB-SA1(ed): IF PS indicator light is lit AND EB2 indicator light is lit AND SA1 is not lit AND sa1 is selected THEN SA1 is the fault.
CPN-TAB-SA2(ed): IF PS indicator light is lit AND EB2 indicator light is lit AND SA1 indicator light is not lit AND sa2 is selected THEN SA2 is the fault.
CPN-TAB-LBm(ed): IF PS indicator light is lit AND EB1 indicator light is lit AND MA indicator light is lit AND LB indicator light is not lit AND ma is selected THEN LB is the fault.
CPN-TAB-LBS1(ed): IF PS indicator light is lit AND EB2 indicator light is lit AND SA1 indicator light is lit AND LB indicator light is not lit AND sa is selected THEN LB is the fault.
CPN-TAB-LBS2(ed): IF PS indicator light is lit AND EB2 indicator light is lit AND SA2 indicator light is lit AND LB indicator light is not lit AND sa is selected THEN LB is the fault.
These edited productions now give an adequate account of the experienced table group, as they are in each case simple restatements of counterpart Pet productions.

What mechanism might explain the production editing that we have done ad hoc? As mentioned earlier, the compiled table productions contain condition redundancies. In order to generate more satisfactory productions these redundancies need to be removed. Neches (1981, 1987) described a set of heuristics to eliminate various redundancies in the operation of production systems, one of which targets redundant conditions:

Over-determined Tests: IF a procedure contains two tests as part of a decision, call them T1 and T2, and it is observed that the tests agree (i.e., T1 is observed to succeed on several occasions with T2 also succeeding on all of those occasions, and is observed to fail on several other occasions, with T2 also failing on all of those occasions), THEN try deleting one of the tests. (Neches, 1987, p. 176)

This heuristic cannot actually do the job of removing all redundant tests that we need to remove, as it will only delete an arbitrary one of two tests if there is a two-way implication between the pair (i.e., if T1 implies T2 AND T2 implies T1). In several of the examples noted earlier, the implication is in one direction only, so the redundancy is asymmetric and one of the tests cannot be deleted.

Even if this problem is overcome, the overdetermined tests heuristic, like all Neches' procedure modification heuristics, works by inspecting a stored goal structure trace of problem-solving episodes, and thus requires a memory that is not available in the ACT* architecture, or in many cognitive model production systems (but see Langley, 1983, 1987). Such a detailed memory for problem-solving steps seems a rather implausible basis for general psychological learning mechanisms.

We therefore propose replacing the overdetermined tests heuristic with a redundant implication heuristic that requires no such memory, but rather exploits declarative knowledge about causal relations between tests. In the current situation, such knowledge can be provided by a model of the device, just as the authors used to edit the composed table group productions by hand. Such knowledge can be represented in an ACT* model.

Redundant Implication: If a procedure contains two tests as a part of a decision, call them T1 and T2, and it is observed that T1 implies T2 THEN try deleting the second test, T2.

Let us see how this heuristic could be applied. Consider the following trace of productions:

P1: IF the goal is to find a fault AND PS indicator light is lit THEN set a subgoal to check EB1.

P2: IF the goal is to check EB1 AND EB1 indicator light is not lit AND eb1 is not selected THEN POP the goal AND tag EB1 as checked.
P3: IF the goal is to find a fault AND EB1 has been checked THEN set a subgoal to check EB2.
P4: IF the goal is to check EB2 AND EB2 indicator light is lit THEN POP the goal AND tag EB2 as checked.
P5: IF the goal is find a fault AND EB2 has been checked THEN set a subgoal to check MA.
P6: IF the goal is to check MA AND MA indicator light is not lit AND EB1 indicator light is not lit THEN POP the goal AND tag MA as checked.
P7: IF the goal is to find a fault AND MA has been checked THEN set a subgoal to check SA1.
P8: IF the goal is to check SA1 AND SA1 indicator light is not lit AND SA1 is selected AND EB2 is lit THEN give SA1 as the fault AND STOP.

These productions have been generated through composing and proceduralizing sub-sequences of productions that occur when the table users solve the fault-finding task when SA1 is broken. The redundant implication heuristic can be applied to P2 and P6. In P2, it is always the case that the EB1 indicator light will not be lit if eb1 is not selected. Similarly in P6, the MA indicator light will never be lit if the EB1 indicator light is not already lit. In both cases, only the first of the two conditions is required to maintain the validity of the production. If the “EB1 is not lit” condition and the “MA is not lit” condition are deleted from P2 and P6, respectively, we get the following productions:

P2ri: IF the goal is to check EB1 AND eb1 is not selected THEN POP the goal AND tag EB1 as checked.
P6ri: IF the goal is to check MA AND EB1 indicator light is not lit THEN POP the goal AND tag MA as checked.

Reapplying composition at this point leads to the following productions:

P9: IF the goal is to find a fault AND PS indicator light is lit AND eb1 is not selected AND EB2 indicator light is lit AND EB1 indicator light is not lit THEN set a subgoal to check SA1.
P10: IF the goal is to check SA1 AND SA1 indicator light is not lit AND SA1 is selected AND EB2 is lit THEN give SA1 as the fault AND STOP.

Once again, P9 requires that eb1 is not selected AND EB1 indicator light is not lit. As stated earlier, there is no need to have both these conditions. Removing the same condition as before, a new production is created.

P11: IF the goal is to find a fault AND PS indicator light is lit AND eb1 is not selected AND EB2 indicator light is lit THEN set a subgoal to check SA1.

However, this new production also has two tests that have an implication relationship: “eb1 is not selected” and “EB2 indicator light is lit.” Whenever the EB2 indicator light is lit, there is no way that eb1 can be selected. Thus, applying the redundant implication heuristic generates the following production:
P12: IF the goal is to find a fault AND PS indicator light is lit AND EB2 indicator light is lit THEN set a subgoal to check SA1.

Reapplying composition produces:

P13: IF the goal is to find a fault AND PS indicator light is lit AND EB2 indicator light is lit AND SA1 indicator light is not lit AND SA1 is selected THEN give SA1 as the fault AND STOP.

This production solves the fault-finding task in the manner described by the experienced table users.

In summary, the application of the redundant implication heuristic has generated a production rule that can successfully mimic the experienced user's fault-finding strategy. Indeed, if this heuristic is used during the compilation of the production traces for all the different fault-finding tasks, it leads to productions that are able to solve the different examples of the task in the manner described by the experienced users.

6. CONCLUSIONS

Two sets of production rules that model the fault-finding strategies described by the novice procedure and table groups have been offered and then subjected to Anderson's compilation mechanisms (Anderson, 1987; Neves & Anderson, 1981). Comparing the results with the descriptions of the fault-finding strategies provided by the experienced users, it was found that knowledge compilation successfully modeled the procedure group's change in strategy, but was unsuccessful with respect to the table group. The application of a heuristic, suggested by Neches' (1987) work on the modification of procedures, was found to fill the gap left by knowledge compilation.

There is one more puzzle that requires further consideration: Why does knowledge compilation work in some circumstances and not in others? Knowledge compilation provides an accurate account of the change in strategy for the procedure users but not the table users. Condition redundancies remain in the procedure group's productions. For instance:

CPn-PROC-SA1: IF on is selected AND PS indicator light is lit AND eb2 is selected AND EB2 indicator light is lit AND SA1 is selected AND SA1 indicator light is not lit THEN give SA1 as the fault.

requires that both "on is selected" and "PS indicator light is lit," when only the latter is required, because for the latter to be true the former must also be true. Therefore, it may be possible to remove the "on is selected" condition. The question arises, why does the table group remove the condition redundancies and not the procedure group?

One possible explanation relies on our earlier finding that different instructions lead to mental representations of the device in which different ob-
jects are the primary conceptual entities. For the table group, the change from productions with many redundant conditions to productions with no redundant conditions may be triggered by the lack of consonance with the table users’ knowledge. The primary conceptual entities represented by the table users are the components and the attributes of these entities are the switch positions. The productions generated through knowledge compilation, however, have test conditions that rely disproportionately on checking the positions of the switches. This imbalance may become obvious to the table users because they view the device primarily in terms of components. Such an account can also explain why the procedure group does not notice the condition redundancies. Their primary conceptual entities are the switches, and there is no mismatch between the way they are solving the task and their representation of the device. If such a mismatch is not apparent to the procedure users, then they may not notice any redundant conditions or actions.

In conclusion, we have found that knowledge compilation provides a good, but not quite adequate explanation of the interplay between instructions and practice in learning to use a device. An additional mechanism proved necessary to model the efficient strategies developed by one group of participants on one of the two main tasks. We have argued that a kind of heuristic procedure modification, like those proposed by Neches (1987), can fill this role. The particular mechanism we propose does not require expensive statistical computations over a trace of problem-solving activity, but it does require ongoing access to declarative knowledge.

This proposed expanded role for declarative knowledge in procedural learning is in broad agreement with several observations in the recent cognitive science literature. Ohlsson and Rees (1991) presented a new model of learning within the production system framework. This model, HS, revises production rules to take account of general domain knowledge expressed as state constraints. These revisions work on the actions of production rules to ensure that inconsistent states are never generated. Clearly such a mechanism cannot perform redundant-condition eliminations without some extensions; nevertheless, the central idea that knowledge-based constraints on problem states can be brought to bear on the encoding of procedures is shared by the two proposals.

In Ohlsson’s system, the application of domain knowledge is automatic. Our proposal is less well developed, and therefore open to the alternative interpretation, that application of domain knowledge to procedural learning requires conscious strategic deliberation. Recent work on metacognition offers some support to this conjecture. In experiments on the mechanics of problem solving, Chi, Bassok, Lewis, Reimann, and Glaser (1989) discovered that successful students were those that explained their own problem-solving activities to themselves. Chi et al. argued that reflecting on self-generated
solutions allows a learning-with-understanding, in which the conditions for
taking certain actions are refined. Van Lehn (1991), in a detailed reanalysis
of the Anzai and Simon (1979) Tower of Hanoi protocol, found evidence
that the learner breaks off from his or her routine problem-solving activity
to explicitly consider the strategies that he or she is applying, and to im-
prove these where inefficiencies are observed.

There is emerging empirical evidence, then, that explicit knowledge-driven
consideration of current strategies can play a role in learning by doing. The
study we have reported provides new evidence for explicit use of declarative
knowledge in the refinement of procedural skill. This evidence suggests a
revision of the simplest model of the relationship between instruction and
practice. It seems that instructionally derived declarative knowledge does
not only provide the raw material for general problem-solving procedures
and learning mechanisms; rather it remains available for exploitation in
refinement of task strategies.

REFERENCES

Press.
124–140.
How students study and use examples in learning to solve problems. *Cognitive Science*,
5, 121–152.
R. Neches (Eds.), *Production system models of learning and development*. Cambridge,
MA: MIT Press.
tation, Carnegie Mellon University, Pittsburgh.
Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.


