Int. J. Man-Machine Studies (1986) 25, 153–189

Instructionless learning about a complex device: the paradigm and observations

JEFF SHRAGER† AND DAVID KLAHR
Department of Psychology, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.

(Received 5 October 1985 and in revised form 4 April 1986)

In order to study the mechanisms that underlie "intuitive" scientific reasoning, verbal protocols were collected from seven computer-naive college students asked to "figure out" a *BigTrak* programmable toy, without a user's guide or other assistance. We call this paradigm *Instructionless Learning*. The present paper presents a detailed account of how people learn about a complex device in an instructionless-learning context. Subjects' behavior is divided into an *orientation phase* and a *systematic phase*. We attend most carefully to the systematic phase. Learners form *hypotheses* about various aspects of the BigTrak: the *syntax* of interaction, the *semantics* of operators, and the *device model*— which includes objects such as memories, switches, etc. Subjects attempt to confirm hypotheses from which predictions can be made, to refine hypotheses that do not immediately yield predictions, and to verify their total knowledge of the device. Hypotheses are formulated from observation. If an initial hypothesis is incorrect, it will yield incorrect predictions in interactions. When such failures occur, learners change their theory to account for the currently perceived behavior of the device. These changes are often based upon little evidence and may even be contradicted by available information. Thus, the new hypotheses may also be incorrect, and lead to further errors and changes.

1. Encountering complex devices

How do people learn to operate moderately complex devices—such as digital watches, videocassette recorders, copying machines, or video games— with little or no instruction? Such computer-controlled devices often contain internal mechanisms that obscure the relationship between the user's input and the device's behavior, and thereby make the devices difficult to learn about. In order to deal with such devices, users attempt to discover how external operations are mediated by the device's internal states. In other words, users try to construct and use a "conceptual model" of the device.

> There may be some situations in which we cannot control the user's training, such as the bank customer walking up to an automated teller for the first time. The user(s) in these situations must, if they are to acquire a conceptual model at all, induce it from interacting with the system. This is indeed difficult . . . (Moran, 1981, p. 43).

Little is known about how users cope with Moran's "difficult" situation. In this paper, we describe the results of an investigation in which we intentionally created such a situation in order to better understand how people formulate conceptual models

† Current address: Intelligent systems Laboratory, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, U.S.A.

0020-7373/86/080153+37$03.00/0

of complex devices. We presented adults with a computer-controlled toy (to be described below) and asked them to "figure it out" without the benefit of any written or verbal instructions. We call this an instance of *instructionless learning*. Our goal is to provide a rich and detailed description of subjects' behavior in an instructionless-learning context. Such a description is a necessary first step in two related enterprises: constructing more learnable devices, and formulating a psychological theory of the scientific-reasoning process that underlies instructionless learning.

In order to figure out how a device works our subjects must discover the possible actions of the device and the range of their own behaviors that is necessary to get the device to exhibit such actions. In the course of learning these things, they may formulate and use a theory about why their actions cause the device to do whatever it does—a conceptual model.

We distinguish between a subject's knowledge about the internal workings of the device—a "device model", and the more inclusive knowledge about what the device can do and how to get it to perform—the "device schema".† Young (1981) suggested that differences in users' models might lead to differences in the ease of learning to use the target systems. Support for this view comes from studies showing that a veridical device model helps the user infer specific operating procedures for the device (Kieras & Bovair, 1984) and to construct an appropriate problem space for use of the device (Halasz, 1984). Most of the emphasis in the rapidly growing literature on mental models has been on how people *use* mental models (e.g. Gentner & Gentner, 1983, Halasz, 1984, Halasz & Moran, 1983), rather than on how they form them in the first place. A few experimental studies do deal with learning (e.g. Bott, 1979; Kieras & Bovair, 1984). However, they focus on how an explicit, externally imposed model is assimilated and used, or on the role that a single mechanism (e.g. analogy) plays in isolated learning events, rather than on complete learning episodes.

The present study directly addresses the question of how people formulate device schemas and device models when they are not explicitly provided, starting with no domain-specific knowledge, and ending with a relatively complete schema. In our situation, the device model and the device schema are formulated simultaneously: that is, the user learns how the device works at the same time as he learns how to make it work. This leads to highly complex cognitive processing, involving concept formation, rule induction, learning via analogy, and the design of experiments. A complete account of how all these processes interact to produce a device schema would constitute a theory of instructionless learning.

In section 2 we describe the context for our investigation: the particular device, the subjects, and our procedure. The broad characteristics of subjects' behavior are described in section 3. The content of what is learned about the device is described in sections 3.3 and 4. In section 5, we take a preliminary step toward the construction of a theory of instructionless learning by presenting a brief sketch of some of the components of such a theory. The details of these mechanisms and of a computer implementation are discussed elsewhere (Shrager, 1985).

† The literature on mental models has not yet produced a consensus on terminology. Concepts similar to ours were proposed by Young (1981, 1983), who claimed that a useful model of a device must consist of both a "surrogate" model describing the device and its functionality and a "task/action" model describing how to use the device to obtain results. Young's surrogate and task/action models correspond roughly to our "device model" and "device schema" notions.

2. A context for studying instructionless learning

In order to elicit behavior sufficiently rich to enable us to carry out detailed analysis of subjects' thinking processes, we used a device that elicits easily observable behavior (such as control setting and button pushing), while also enabling subjects to make unambiguous reference to different aspects of its functioning (for the analysis of verbal protocols). The device is sufficiently novel, interesting, and challenging to keep subjects actively engaged for about 30 min, but sufficiently learnable that they are satisfied with their progress. A commercially available toy called BigTrak† meets all of these criteria.

2.1 BIGTRAK

The BigTrak is a six-wheeled vehicle, entirely self-contained and powered, approximately 30 cm long, 20 cm wide and 15 cm high. Interaction takes place via a keypad on the top of the toy, illustrated in Fig. 1. In order to get BigTrak to behave, the user

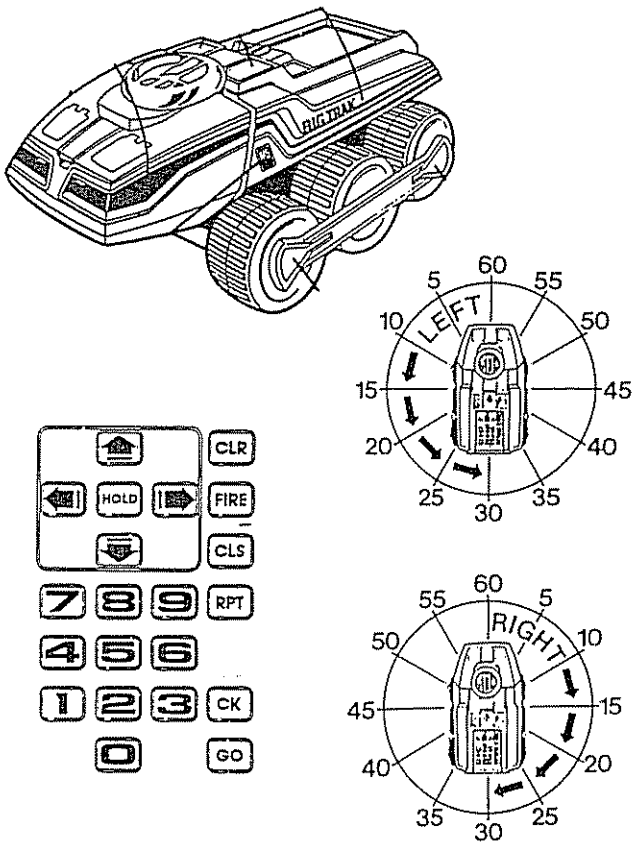

FIG 1. View of BigTrak, with close-up of its keypad, and an illustration of turn units

† BigTrak is a product of the Milton Bradley Corporation.

enters a series of up to 16 instructions, each consisting of a function key (the command) and a 1- or 2-digit number (the argument), terminated by the **GO** key. BigTrak then executes the program by moving around on the floor. The effect of the numerical argument depends on which command it follows. For forward (↑) and backward (↓) motion, each unit corresponds to approximately 1 ft. For left (←) and right (→) turns the unit represents 6° steps. The **HOLD** unit represents a delay (or pause) of 0·1 sec, and the **FIRE** unit represents one "event": the firing of the cannon (indicated by appropriate sound and light effects). For **RPT**, the numerical argument does not correspond directly to a physical action, but to the number of recent steps to be repeated at execution time. That is, at run time, **RPT** *n* causes the *n* steps preceding the **RPT** command to be repeated once more.

The other keys shown in Fig. 1 are **CLS** and **CK**. **CLS** clears the most recently entered instruction, and **CK** checks the most recently entered instruction by executing it in isolation. Using **CK** does not affect the contents of memory. The **GO, CLR, CLS,** and **CK** commands do not take an argument. The series of keystrokes shown in Table 1 would produce the following behavior by BigTrak: Move forward 5 ft, turn left 42°, move forward 3 ft, turn right 90°, pause for 5 sec, fire twice, then repeat the preceding three instructions: turn right 90°, pause for 5 sec, and finally fire twice.

TABLE 1
An example of programming BigTrak

CLR	Zero out the toy's memories.
↑	Indicate that the toy should move forward . . .
5	5 ft
←	Turn left . . .
7	seven units (of 6° each).
↑	
3	Forward 3 ft.
→	
1	Turn right 90° (or 15 "minutes" on a clock face)
5	
HOLD	
5	Pause for 50 tenths of second.
0	
FIRE	Fire the "laser cannon" . . .
2	. . . two times.
RPT	Repeat the preceding . . .
3	. . . three instructions (turn, hold, fire) once more.
GO	BigTrak executes the program.

Certain combinations of keystrokes (e.g. a third numerical digit, or two motion commands without an intervening numerical argument) are not permitted by the *syntax* of the programming language. With each syntactically legal key-stroke, BigTrak emits an immediate, confirmatory beep. Syntactically illegal key-strokes elicit no response, and they are not entered into program memory.

2.2. SUBJECTS

Seven CMU undergraduates (between the ages of 17 and 20) participated in the experiment in order to fulfill a class requirement. Because BigTrak is a programmable device, we selected subjects having at most a single course in computer programming.

The subjects are:

AM: a freshman design major with no computer experience (age 18, female).

CB: a sophomore biology major who had played video games and used a database retrieval program written by someone else (age 19, male).

EC: a freshman business major with no computer experience (age 18, female).

FC: a freshman mechanical engineer who had "messed with" BASIC in high school (age 18, male).

JS: a freshman of undeclared major. She received "A" in an introductory Pascal course, and had some programming experience with an HP41CV calculator (age 18, Female).

RD: a junior chemical engineer who had taken introductory Pascal and worked as a student consultant, but claimed to know very little about programming (age 20, male).

TE: a freshman business major with a half semester of Pascal (estimated grade: "B") (age 17, female).

2.3. METHOD

The subjects were run individually. They were given a few minutes of preliminary instruction and practice on how to give a good verbal protocol while solving a "fifteen" puzzle. They were then seated on the floor in the middle of a laboratory (5×3 m) with BigTrak within reach (at "base"), but covered with paper so that it was not visible until the start of the experiment. Subjects then read an instruction sheet containing the following key phrases:

> "The purpose of this experiment is to observe people learning about a new toy . . . called the "BigTrak" . . The general idea is for you to learn as much as you can about the BigTrak in the time allotted . . "

The experimenter then removed the covering paper, turned on BigTrak (in sight of the subject, but without explanation) and left the room. The experimenter interacted with the subject as little as possible when the subjects asked questions. The following normal intrusions occurred:

If the subject was pressing buttons without talking, or stopped talking for a few seconds, the experimenter would ask what the subject was thinking

If the subject forgot to return the tank to base, the experimenter would ask the subject to do so

If the subject entered a sequence of instructions that would have caused BigTrak to go on for a very long time (e.g. **FIRE 99**), the experimenter might show the subject that the toy could be stopped by turning it off.

Occasionally the BigTrak would do something abnormal (e.g. did not move when **GO** was pressed with a valid program). If the experimenter noticed this, he might ask the subject to try the operation again, perhaps after clearing it or turning it off and on.

In addition, the following discretionary intrusions occurred on several occasions:

If the subject had not obtained a movement from the device after 2–3 min, the experimenter could recommend pressing **GO** in a context that would obtain an action. No explanation was given for this.

If the experimenter felt that the subject had totally confused the BigTrak, confused him or herself, was doing the same operation over and over, or was stuck doing something in a particularly bad way, the

experimenter might ask a question of the subject, recommend pressing a particular button, or otherwise force a new goal on the subject in order to break out of the situation. This sort of prod was very rarely used.

Sessions ended when the subject claimed to be done, for whatever reason; or the experimenter would, after about 30 or 40 min, stop the subject at a convenient point.

All verbalizations and keypresses, as well as whatever part of BigTrak's behaviors took place within camera range were recorded on videotape. Although subjects were allowed to do anything that they wanted to do with the toy, they were reminded to return BigTrak to its base position before pressing buttons or otherwise interacting, in order to keep the keypad within camera range.

After the session, the experimenter asked the subject to explain the toy to him and collected a personal history including the subject's experience with computers and a self-assessment on various skills including maths, sciences, sports, cooking, and gadgetry. The present paper is not concerned with individual difference in behaviors so we will not explore the results of that survey.

The videotapes for each subject were transcribed into the format shown in Appendix I (the complete raw protocol for subject FC). The protocol includes all verbalizations and all of the subject's key presses as well as a description of BigTrak's behavior. In order to convey a sense of the complexity and richness of the data with which we are dealing, we urge the reader to take a careful look at the first few pages of the protocol in Appendix I. (A plus sign "+" to the left of a keypress indicates that the key beeped: that is, it was syntactically legal and was registered by BigTrak. This convention will be used throughout the paper).

We will proceed through several levels of aggregation and interpretation of the protocols. In the next section (section 3) we discuss the broad temporal properties and some of the gross features of subjects' performance. In section 3.3 we characterize the different types of knowledge that must be acquired about BigTrak. In section 4 we present a more rigorous basis for making inferences about what subjects know from what they say and do.

3. Overall description of learning

In this section, we provide a coarse-grained description of the learning process in order to familiarize the reader with the protocols, introduce terminology, and statistically summarize subjects' behavior. In section 4 we discuss what is actually learned by subjects in the task.

3.1 INITIAL ORIENTATION PHASE

We divided the protocols into two phases: *initial orientation* and *systematic investigation* according to the first time that a subject obtained a movement (or firing) action from BigTrak. In the initial orientation phase, subjects begin with direct attempts to get BigTrak to behave perceptibly. Most subjects expect the device to move immediately in response to presses of the directional arrow keys or the **GO** key. These initial attempts invariably fail, because subjects are ignorant of the rudiments of BigTrak's syntax. That is, they do not know that unless the **GO** or **CK** keys are pressed while there is at least one instruction in BigTrak's program memory, BigTrak will not behave. For example, as shown in Appendix I, FC initially expects **GO** to cause some action (FC001-002). However, because no other commands have yet been entered, BigTrak

does nothing. Then he tries various combinations of commands and numbers (FC002-044)—to no avail.

We define the end of the initial orientation phase as the point where BigTrak actually moves or fires. All subjects eventually learn that the **GO** key releases the action of the device. (Five of them discovered this on their own, and two needed a hint from the experimenter). For FC, this occurs after 3·5 min when he gets BigTrak to execute its first motion: a left turn of 264° (FC048-053). Over all subjects, time spent in this phase ranged from 30 sec to 7·5 min ($x = 3{\cdot}14$ min, S.D. = 2·4 min).

3.2. SYSTEMATIC INVESTIGATION PHASE

The remainder of the protocol constitutes the systematic investigation phase, in which the subjects' behavior can be segmented into a series of *episodes*. We separate one episode from the adjoining one at presses of either the **GO** key or the **CK** key. Several features of the protocols suggest that this is a meaningful segmentation: BigTrak (usually) does something after one of these keys is pressed; subjects tend to verbalize observations after pressing **GO** or **CK** regardless of whether BigTrak does anything or not; they tend to formulate hypotheses (for the next experiment) after **GO**; sometimes they predict the result of an experiment just before hitting **GO** or **CK**.

The systematic investigation phase is where we see evidence of the three processes that guide behavior: hypothesis formation, experiment construction, and analysis of results. Subjects generate hypotheses about the nature of the device and conduct experiments (i.e. sequences of key presses) to confirm these hypotheses. Experiment planning includes a prediction of results—what the toy will do in response to the experimental interaction. If BigTrak's performance is consistent with the predictions, then the only immediate learning that results is the confirmation of the hypothesis. If the performance does not match the prediction, then the subject can attempt to analyse the toy's behavior and form a new or modified hypothesis. Alternatively, the subject can attempt to replicate the experiment or simply abandon the hypothesis and construct another.

For example, in segment FC078-107,FC:
formulates the hypothesis that the numerical arguments for turns correspond to 1° units (FC078);
generates an experiment to assess that hypothesis (FC079-083);
notices that BigTrak's behavior is inconsistent with the hypothesis (FC084);
and abandons the degree hypothesis for another: that numerical units correspond to 1-ft units in a specific direction (FC094-107).

3.2.1. Features of behavior during the systematic investigation phase

The gross features of subjects' behavior during the systematic investigation phase are summarized in Table 2. Total time in this phase averages about 30 min, but not all of it is available for pressing buttons, because BigTrak is executing the programs during some of this time (e.g. the program in Table 1 takes about 30 sec to execute). We define the *duration* of the systematic investigation phase as total time minus the BigTrak run time. Durations range from 18·7 to 29·3 min, distributed as shown in the second column in Table 2. All subjects maintain a fairly high degree of activity during the systematic phase: they execute several hundred key presses, producing a mean rate of 18

TABLE 2
Gross performance during the systematic investigation phase

Subject	Phase Duration (Min)	Key presses†	Words per Min	Keystroke rate‡	Number of episodes	Episode duration§	Length¶
AM	29·3	423	150	14·5	68	25·9	6·2
CB	22·3	470	70	21·1	45	29·7	10·4
EC	25·7	362	71	14·1	46	33·5	7·9
FC‖	22·3	319	60	14·3	45	29·7	7·1
JS	23·3	543	125	23·3	62	22·5	8·8
RD	18·7	391	101	21·0	33	34·0	11·8
TE	19·8	345	92	17·4	64	18·6	5·4
Mean	23·0	408	96	18·0	52	27·7	8·2
S.D.	3·6	78	32	3·8	12·9	5·7	2·3

† Total number of key-presses during systematic investigation phase.
‡ Mean number of keys pressed per min.
§ Mean time (in sec) between successive episodes, less BT run-time.
¶ Mean number of key-strokes per episode (including **CLR**, **CK** and **GO**)
‖ Note that FC's full protocol is listed in Appendix I.

key-presses per min, and they talk at normal conversational rates (mean rate of approximately 100 words per min). These high activity rates suggest that our protocols are recording most of the reasoning that subjects do (e.g. experiment planning) during this phase. More importantly, they also support a key assumption we will make when analysing the protocols: that they are true concurrent verbalizations of what the subjects are thinking, rather than retrospective and potentially distorted reports. (We will have more to say about the validity of the protocol contents in later sections).

As Table 2 indicates, there are wide individual differences in the amount of activity subjects generate, as well as in how much they get BigTrak to do. At the level of simple motor activity, there is approximately a two-to-one difference between the maximum and minimum rates at which different subjects press keys (from 14 to 23 keys per min) and talk about what they are doing (from 60 to 150 words per min). There is also a two-to-one difference in the number of distinct opportunities they have to observe BigTrak behave (from 33 to 68 episodes),† and in the mean number of key-presses per episode (from 5·4 to 11·8).

However, the rate of key pressing is uncorrelated with either the number of episodes ($r(6) = -0{\cdot}13$, N.S.), or their duration ($r(6) = -0{\cdot}19$, N.S.). The subjects with the highest key-stroke rates (CB, JS and RD) vary widely over the range of number of episodes, and a similar dispersion occurs for rate-duration comparisons. Nor is it the case either that the subjects with the fastest fingers have the most to say, or that talking and button pressing are mutually incompatible: the correlation between keys per min and words per min is not significant ($r(6) = 0{\cdot}173$, N.S.).

3.2.2 Episode content

In discussing episode content, we make a distinction between *command length* (the number of legal command keys pressed per episode, including **CLR**, **GO**, **CK** and

† Recall that this is defined in terms of BigTrak behavior. Whenever a syntactically correct **GO** or **CK** is pressed, BigTrak behaves. This defines the episode delimiter.

CLS) and *program length* (the number of instructions executed when BigTrak actually behaves). Command length does not necessarily correspond to program length. Command length can exceed program length because it includes function keys that are not entered in program memory, such as **CLR, GO, CK** and **CLS**. For example, consider the excerpt from Appendix I (FC445-453). FC is exploring the function of **HOLD**, and he initially fails to give it a numerical parameter, so that the first **GO** yields no response from BigTrak. He corrects this, and at 453 correctly hypothesizes the semantics of **HOLD**. This episode contains: nine key presses, of which six are legal, and eight commands, of which five are legal. The program that BigTrak executes consists of one instruction: **HOLD 1**. Thus, the command length of this episode is 5, while the program length is 1. It is also possible for program length to exceed command length. This can occur when the subject fails to press **CLR** at the beginning of an episode, or when **CLS** is used during an episode. (Note that even though pressing **RPT** increases the number of actions that BigTrak does, this is still only a single command).

We use command length, rather than program length, as our unit of analysis because it provides a better index of the complexity of each episode, and hence of the potential amount of information that subjects can extract from episodes. Also, command length corresponds more closely to the mental processing that the subject is doing, rather than the work of BigTrak's computer.

Subjects employ fairly short episodes. As shown in the last column of Table 3, 50% of all the 363 episodes contain three or fewer commands, and for five of the seven subjects the modal command length is either 2 or 3. There are some notable deviations from this general pattern: RD had no episodes with fewer than three commands, while CB and EC had a relatively uniform distribution of episodes having from two to six

TABLE 3
Percentage of episodes having different command lengths

Program Length	Subjects							
	AM	CB	EC	FC	JS	RD	TE	Overall
1	24	2	4	0	5	0	14	7
2	35	16	17	20	6	0	34	18
3	15	16	15	31	34	24	38	25
4	9	22	20	11	18	24	6	16
5	4	16	13	9	10	12	0	9
6	4	11	15	22	6	18	2	11
7	6	7	4	4	6	6	3	5
8	1	0	9	2	3	6	0	3
9	1	7	0	0	5	0	0	2
10	0	0	0	0	2	0	2	1
11-20	0	4	2	0	5	0	2	2
21-30	0	0	0	0	0	9	0	1
N†	68	45	46	45	62	33	64	363
Mean	2·9	4·7	4·4	4·0	4·7	6·5‡	2·9	4·3

† Total number of episodes, e.g. 24% of AM's 68 episodes had command length equal to 1.
‡ If the three episodes having command length greater than 20 are eliminated, the mean command length for RD is 4·7.

commands. Even though a few subjects have skewed distributions, the mean command lengths vary only from about 3 to 5.† The between-subject variation in mean command length is similar to the two-to-one ratio between the maximum and minimum values of the other measures described earlier. Over all subjects, the most common episode length (25% of all episodes) was 3. This makes sense, since the canonical program is: **CLR** ⟨instruction⟩ **GO**. Fewer than 10% of all episodes had more than seven commands.

These measures support the view that subjects are conservative in running experiments. The longest programs that they can run have 16 instructions, and there is no upper limit to command length (e.g. subjects can include an unlimited number of **CK**s, **CLS**s, and **CLR**s in an episode). Nevertheless, subjects seem to realize that they will have difficulty in learning much from such complex episodes and tend to avoid them. (In some cases, subjects explicitly mention that they are going to run a simpler experiment in order to clarify some aspect of their schema).

The lack of correlations among these coarse-grained measures does not mean that instructionless learning is inexplicable, but it indicates that in order to understand it, we will have to move to a more detailed level of analysis. In the next sections (sections 3.3 and 4) we deal with the contents of the BigTrak schemas formed by learners, and in section 5 we propose mechanisms that may underlie the instructionless learning process.

3.3. WHAT NEEDS TO BE LEARNED: CLASSES OF KNOWLEDGE

In this section we establish a taxonomy of knowledge types which enables us to characterize how much is correctly learned about BigTrak and what sort of incorrect hypotheses are made in the course of learning. Both incorrect and correct hypotheses play a role in generating behavior and lead to behavior that, through analysis, leads to correct learning.

There are three broad classes of knowledge to be acquired about BigTrak: *syntactic knowledge* about the grammar of the programming language; *semantic knowledge* about the function of a particular key, program, or component of the device; and *model knowledge* about the internal mechanisms of the device, their interrelationships, capacity, possible states, etc. The latter corresponds to the device model described in section 1, while all three types comprise the device schema.

3.3.1. Syntactic knowledge

The BigTrak instructions have a specific syntax. Some commands must be followed by numerical arguments. Other commands do not allow arguments (**CLS, CK, GO, CLR**). Key-stroke sequences that violate the grammar are not entered into the program memory, and they are not rewarded by a beep. *Syntactic knowledge* is knowledge about what constitutes a legal sequence of keystrokes.

The confirmatory beep for syntactically correct key strokes is a simple device by which the subject can infer part of BigTrak's syntax: in particular, whether or not a command must be followed by a number. In the example shown in Appendix I, less than one third of the 41 key presses during the initial orientation phase (FC001-058) are syntactically correct. After that, almost all are.

Although syntactic knowledge is mastered fairly early in the course of the learning sessions, BigTrak's hint toward facilitating its acquisition (the confirmatory beep) is

† See note b in Table 3 for the one exception to this

rarely explicitly mentioned by most subjects. For example, in the entire protocol in Appendix I, the beep ("buzz") is only mentioned once, and even there it is incorrectly interpreted (FC049). However, the beep feedback is relied upon fairly often as a cue that something has been done incorrectly, and that BigTrak "wants" something else.

3.3.2. Semantic knowledge

Semantic knowledge is *knowing what a key does.* More precisely, the semantics of keys are the actions that BigTrak performs as a result of a particular key sequence. For example, the meaning of the ← and → keys are rotations to the left and right. Subjects often base their initial hypothesis about the semantics of a key on the semantics of a word (or symbol) it reminds them of. For example, they often assume (correctly) that **CLR** clears memory, and that **RPT** repeats some part of the action sequence. Thus it is possible to acquire some semantic knowledge in the absence of a correct syntax. However, note that the word "repeat" only provides a general guide to the particular meaning of the **RPT** key. This must be refined and verified—processes which we will describe in section 5.

More common are the cases where syntactic knowledge is almost entirely correct, but the semantics are still faulty. The protocol segment in Appendix I (FC153-164) provides an illustrative example. At this point, FC is focusing on how numerical arguments work. (Recall that numbers can represent distance, degrees, or number of program steps, depending on which command they follow). FC knows that commands require numerical arguments and that instructions can be entered in sequence to form a program. However, he has an erroneous view of the meaning of the numbers following each command. When **GO** is pressed (164), BigTrak goes forward 6 ft and fires its cannon four times. But FC expected it to go forward 3 ft and fire once because his device schema has a serious flaw: He views the numbers as corresponding to program step numbers (as in programming languages like BASIC), and he believes that he must follow each command with the memory location (or sequential position) in which it is to be stored. He does not know that commands can have parameters but instead attempts to go more than a single unit distance entering the single ↑ command repeatedly.

3.3.3. Device model knowledge

The view of BigTrak as a programmable device, and the details of how programs are entered, stored, modified and executed, constitute what we call *device model knowledge.* This includes a slightly wider definition of BigTrak semantics. Except for **GO** and **CK**, none of the key presses correspond to any immediately observable BigTrak action. Instead, they cause changes in the internal states of the device. A more accurate definition of semantics would be *knowing what a key's function is, as mediated by device model knowledge.*

For instance, subjects conceive of **GO** as the "releaser" of previously programmed physical actions. But this implies a device model of a system that can store and execute the program—it has memory and can behave in complex ways with respect to that memory. Similarly, in order to understand what **CLS**, or **CLR**, or **RPT** do, the subject has to understand BigTrak as a programmable device, rather than just an object that moves in the physical world.

This idea of program memory usually becomes explicit when BigTrak shows signs of remembering keystrokes from previous episodes. Consider the portion of protocol from 105 to 127 in Appendix I. Before 98, FC knew and used only single-instruction programs. (Recall that BigTrak will accumulate up to 16 instructions if the **CLR** key is not used between the programming episodes.) FC should have used **CLR** before 108 and 112. Since he did not, at 114, BigTrak executed the program: ↑ **1** ↑ **2** ← **1GO**. This was not expected, and he explained it by hypothesizing memory. The form of the next experiment is a direct result of this hypothesis. FC intentionally puts more than one instruction between the **CLR** (which he has previously used to start new experiments) and the terminating **GO** in order to discover whether more than one instruction can be used—as indicated by the hypothesis.

3.3.4. Summary of knowledge types

We have characterized three types of knowledge about BigTrak—syntactic, semantic and model—and given examples of subjects dealing with each of them. This taxonomy was devised in order to structure the analysis to be presented in the next section. However, we do not intend to imply that subjects make such a clear distinction when they are acquiring or using knowledge about BigTrak. In fact, as we have already indicated, the three types of knowledge interact in complex and subtle ways. We will explore those interactions more fully in section 5.

4. What subjects learn about BigTrak

In this section we deal with the question: "What is learned?" Our answer takes the form of a further analysis of BigTrak knowledge into distinct *knowledge elements* and an indication of the temporal course, stability and final state of the acquisition of the elements. We will discuss both correct and incorrect elements.

4.1 INFERRING KNOWLEDGE FROM THE PROTOCOLS

The verbal and behavioral protocols provide our only data base for what was learned: we use no supporting tests of subjects' knowledge about BigTrak. Thus, it is important to describe our assumptions about the information content of the protocols. Our most fundamental assumption is that the verbalizations are neither epiphenomenal nor disruptive. This is not a general claim about *any* form of verbal protocol, but rather a specific claim that our conditions meet the criteria described by Ericsson & Simon (1984) for pertinent and non-disruptive verbalizations. These conditions make it reasonable to assume that the verbal protocols provide a veridical "window" on the subject's mental processes at the level of specific hypotheses about how BigTrak works. Note that we do not ask our subjects to give us a theory about how they do this intuitive scientific reasoning—that is our task. Rather, we ask them to tell us what they are thinking about at the moment, which yields descriptions about how a button works, why BigTrak behaved in a certain way, etc.

Our second assumption is that the protocols have a local focus. That is, the verbalizations will be about the current knowledge elements under consideration. We follow Ericsson & Simon here, too, in assuming that only information in focal attention can be verbalized. Although the verbalizations have a local focus, in determining which of several plausible interpretations to give a particular local segment, we often use a

broader context, sometimes extending over several episodes both prior to and subsequent to the episode in question. In section 4.3 we will give an example of how we use several protocol segments in order to support assertions about what subjects know at particular points in the protocol and what they are focusing on.

We further assume that the total knowledge structure about the device is constructed incrementally. Subjects focus on discrete, semi-independent pieces of knowledge, and they acquire these knowledge elements sequentially. Some elements interact with the ability to acquire others, but for the purposes of the present analysis, the assumptions of decomposability and sequentiality are adequate.

4.2. ACQUISITION OF CORRECT KNOWLEDGE ELEMENTS

How much do subjects learn about BigTrak? In this section we present a detailed analysis of the amount of BigTrak knowledge that was acquired by two of our subjects. The knowledge is represented as a collection of *correct* elements that fall into the

TABLE 4

Acquisition of BigTrak knowledge elements by two subjects: AM and FC

	AM			FC		
	CODE	TIME1	TIME2	CODE	TIME1	TIME2
Simple command semantics						
Move (↑ or ↓)	+	3:30	3:30	+	5:00	5:00
Move argument	+	17:30	17:30	+	6:30	11:30
Turns (← or →)	+	7:30	7:30	+	4:00	15:30
Turn argument	+	17:30	17:30(?)	+	4:30	16:30
Fire	+	6:00	6:00	+	9:30	9:30
Fire argument	+	25:30	25:30	+	9:30	11:30
Hold	!	33:00		+	29:30	29:30
Hold argument	?			+	29:30	29:30
Complex command semantics						
Repeat	+	5:30	5:30	+	18:00	18:00
Repeat argument	!	5:30		!	23:00	
CLS	!	7:00		+	20:30	20:30
CLR	+	5:00	5:00	+	3:30	3:30
GO	+	3:30	3:30	+	3:30	3:30
CK	−	7:00	32:30	!	13:30	
Programs	+	10:30	32:30	+	4:00	4:00
Syntax						
Instruction syntax	+	3:30	39:30	+	4:00	4:00
Model knowledge						
General number meaning	+	10:00	17:00			
Multi-step programs	+	10:00	10:00	+	8:00	8:00

+, The element was ultimately used correctly one or more times. Time1 is the time that the element was first explored, and Time2 is the point after which the element was never used incorrectly. The two times for an element are equal when the subject used the element correctly initially, with no subsequent error. The two times are unequal when the element was used correctly, then incorrectly, and ultimately correctly.

−, The element was used correctly at least once, but subsequently used incorrectly, and without ultimate recovery. In this case, Time2 is the point at which the subject *last* generated an hypothesis about the element.

!, The element was explored, but it was never used correctly. Time2 is not meaningful here.

?, There is no evidence that the element was explored or acquired.

general categories of syntactic, semantic and model knowledge described earlier. These elements, shown as the row headings in Table 4, define what it means to have successfully "figured out" how BigTrak works. *Instruction syntax* is knowledge that most instructions consist of a command key and a numerical argument. *Semantic knowledge* is divided into two groups—simple and complex. *Simple command semantics* includes knowledge about the function of the keys that cause a simple observable action to occur. For example, a *move* element is knowledge that the ↑ key means "forward motion" and/or that the ↓ key means "backward motion." *Move argument* is knowledge that a directional key and its argument mean a certain amount of forward or backward motion. *Complex command semantics* involves the keys that directly deal with internal states, rather than with the motion of BigTrak. Included here is knowledge that a well-formed program is entered by **CLR**, followed by a series of instructions, terminated by **GO**. (The **CLR** is not mandatory. If it is not used, then subsequent instructions are appended to those currently in program memory). *Model knowledge* deals wth the concept of BigTrak as a multistep programmable device, in which instructions are entered sequentially, and stored until cleared, and executed, starting with the first, when **GO** is pressed.†

There are two stages to our analysis. The first stage involves an inference by us about changes in the subject's knowledge state during the session. Every keypress or collection of keypresses is annotated with a formal representation of the knowledge and goals which we infer to have been responsible for that behavior. These inferences are made by interpreting the behavioral and verbal context surrounding the segment and represent various levels of aggregation of the protocols. (FC's protocol with these annotations is nearly 4 times the size of the raw protocol). The second stage is performed by a computer program which processes the annotations; essentially reading the protocol as a problem behavior graph. Figure 2 shows a very small sample of this annotation for FC's protocol segment between 29:00 and 30:00. He learns about **HOLD** during this segment consisting of two explorations and two experiments.

The results of this analysis are shown in Table 4. The entries indicate the order and degree of acquisition and the stability of each correct elements for subjects AM and FC. The following notation is used:

+—The element was ultimately used correctly one or more times. **Time1** is the time that the element was first explored, and **Time2** is the point after which the element was never used incorrectly. The two times for an element are equal when the subject used the element correctly initially, with no subsequent error. The two times are unequal when the element was used correctly, then incorrectly, and ultimately correctly.

—The element was used correctly at least once, but subsequently used incorrectly, and without ultimate recovery. In this case, **Time2** is the point at which the subject *last* generated a hypothesis about the element.

!—The element was explored, but it was never used correctly. **Time2** is not meaningful here.

?—There is no evidence that the element was explored or acquired.

† Not included here, but probably acquired by all subjects, are knowledge elements about the physical dimensions of BigTrak (color, size, material, etc.), its power source, the velocity at which it moves, and so on

```
| | | | | (time |29:00|)
| | | | (end resolve number for rpt unsuccessful)
| | | (end resolve rpt unsuccessful)
| | | (begin resolve hold)
| | | | (begin exploration)
| | | | | (×1 (to (reset)⇒(press clr)))
| | | | | (press clr+)
| | | | | (press hold+)
| | | | | (press hold)
| | | | | (press hold)
| | | | | (×1 (to (start system) (⇒) (press go)))
| | | | | (press go)
| | | | (end exploration)
| | | | (begin exploration)
| | | | | (×1 (to (reset)⇒(press clr)))
| | | | | (press clr+)
| | | | | (press hold+)
| | | | | (press 1+)
| | | | | (×1 (to (start system)⇒(press go)))
| | | | | (press go+)
| | | | | (bt ((nothing 0 1)))
| | | | (end exploration)
| | | | (×1 (hypothesis (to (obtain pause)⇒(use hold operator))))
| | | | (begin experiment)
| | | | | (×1 (to (reset)⇒(press clr)))
| | | | | (press clr+)
| | | | | (×1 (to (enter step)⇒(enter operator) (enter argument)))
| | | | | (press ↑+)
| | | | | (press 1+)
| | | | | (time |29:30|)
| | | | | (×1 (to (obtain pause)⇒(use hold operator)))
| | | | | (press hold+)
| | | | | (press 2+)
| | | | | (press ↑+)
| | | | | (press 1+)
| | | | | (−1 (to start system)⇒(press go)))
| | | | | (press go+)
| | | | | (bt ((move ↑ 1) (nothing 0 2) (move ↑ 1)))
| | | | (end experiment successful)
| | | | (×1 (hypotheses (number expresses time for hold)))
| | | | (begin experiment)
| | | | | (×1 (to reset)⇒(press clr)))
| | | | | (press clr+)
| | | | | (×1 (to (enter step)⇒(enter operator) (enter argument)))
| | | | | (press hold+)
| | | | | (×1 (number expresses time for hold))
| | | | | (press 1+)
| | | | | (press 0+)
| | | | | (×1 (to (start system)⇒(press go)))
| | | | | (press go+)
| | | | | ((bt ((nothing 1 0)))
| | | | (end experiment successful)
| | | (end resolve hold successful)
| | | (×1 (to reset)⇒(press clr)))
| | | (press clr+)
```

FIG. 2. Segment of annotated protocol [See Appendix I (442–469)].

We illustrate how time and degree of acquisition are determined with excerpts from FC's protocol (Appendix I). (Recall that we are only considering the systematic investigation phase, so any knowledge acquired before the first BigTrak action—i.e. during the orientation phase—is listed as if it were acquired at the very beginning of the systematic investigation phase.)

+Move [↑ or ↓]5:00 5:00. FC's systematic investigation phase starts after approximately 3:30, when he presses **GO** at 058, so any previous presses of ↑ or ↓ are not included here. At about 5:00 he first uses ↑, and does so correctly. In none of his subsequent 37 uses of ↑ or ↓ does he give any evidence (such as an incorrect prediction or a statement of surprise) that he does not understand the semantics of these two keys. Note that even when his prediction is disconfirmed by BigTrak's behavior (FC084-089), he runs another experiment (FC094-098) to reaffirm his hypothesis. (The problem here is that he neglected to press **CLR** at FC086, so that at 089 BigTrak turns, and then goes forward; but this is a failure to understand the cumulative effect of non-cleared programs, rather than a confusion about the semantics of ↓ and ↑.)
+Move-argument 6:30 11:30. The meaning of numerical arguments for moves is initially correct (FC105-111), but it is immediately replaced (FC123-155) by the program-step hypotheses discussed in section 5.3. However, at about 11:30, FC returns to the correct understanding of what move-argument means.
!CK13:30. The first explicit statement about **CK** occurs around 13:30, but FC pays little attention to that key until much later (30:00). He never does figure it out.

The results in Table 4 are typical of all of our subjects and support a few generalizations. First, there tend to be elements that are acquired earlier (e.g. **GO**, **CLR**, instruction syntax, etc.) and some that are acquired later (e.g. **HOLD**). Second, the acquisition of a key's semantics does not necessarily correlate with the acquisition of the meaning of its argument (e.g. FC's **RPT** vs **RPT**-argument; AM's **HOLD** vs **HOLD**-argument). These are separable concepts. Third, most of the relevant knowledge is eventually acquired. Finally, the meaning of the **RPT** argument seems to be very difficult to learn. Neither AM nor FC learn it correctly, although they both try on several occasions.

4.3. ACQUISITION OF INCORRECT KNOWLEDGE ELEMENTS

A significant part of the protocols consists of learners' analyses of incorrect hypotheses, which generate evidence enabling them to induce correct hypotheses. In this section, we will briefly discuss some of the incorrect hypotheses that people formulate and then look in detail at one particular example.

4.3.1. Examples of incorrect hypotheses

It is possible for subjects to think anything about BigTrak. One might imagine that pressing **GO** causes the device to explode or fly away. However, our subjects generate a reasonably small range of alternatives—leading, finally to the correct subset of hypotheses. Several examples of subjects' incorrect hypotheses are listed below.

About the semantics of the directional arrows:

Several subjects believed that the arrow keys had a permanent 1-ft distance associated with them. For the left and right arrows, the subject seems to have expected BigTrak to turn 90° in the appropriate direction, and then move 1 ft.

All subjects, except TE, suggested, initially, that the unit for turn arguments was in degrees.

AM believed that the **HOLD** was how one made BigTrak turn "in place".

About the command and program syntax and semantics:

AM believed that pressing the arrow keys composed a pattern that BigTrak would execute. Each pattern is then given numerical name. To enter a pattern you press the arrows in the appropriate order, and then press the number which becomes that pattern's name. After discovering that one had to put a number after each command key, she simply duplicated the numerical name, retaining the belief that BigTrak was going to behave according to the semantics only by the arrow keys pressed.

RD believed that one had to enter a two-digit number, the first being a distance, and the second being something like a direction.

AM believed that the way to get BigTrak to make long turns was to enter a command/argument pair, and press **GO**, and then press the same argument (a single digit number) again and **GO** again.

About the semantics of **CLS**:

AM believed that one could put two different programs into BigTrak, each of only a few (two) instructions. Pressing **CLS**, "switched" BigTrak's attention from one of these two program memories to the other. That is, one could put in two instructions, then press **CLS**, then put in two more instructions, then pressing **GO** would execute the second program; then pressing **CLS**, **GO** would execute the first program, etc. AM later recognized that BigTrak was losing instructions but instead of attributing it to pressing **CLS**, she attributed it to the time-out beep that BigTrak gives if one does not interact with it for about 30 sec.

About the semantics of **RPT**:

FC went through several different theories of **RPT**. He believed at various times in his examination of **RPT** that the argument meant: the number of times to repeat the whole preceding program; the step to which to return and then run through the program once again; the single step which it would repeat once again; the step to which to go back to (as above) but that in executing the sequence again, it would only do fire—but not movement—instructions.

4.3.2. A detailed misconception: FC's vector/BASIC schema

In section 3.3.2 we used a segment of FC's protocol to illustrate that the learner can know the correct syntax of command but have a faulty understanding of the semantics. In the 5 min and 30 sec between lines FC100 and FC144 in Appendix I, FC performs nine experiments and goes through three different understandings of the overall function of the numerical arguments in commands. (Note that he *never* makes a syntactic error during this period.) We will analyse each experiment in order to determine what motivates it and what FC learns from it.

At FC100, he has just learned that the ↑ key followed by a number "makes it go straight". He seems to know that the number refers to the distance (" . . . I don't want it to go that far . . .") although he appears to be uncertain about what the exact units are. The experiment in FC105-107 results in BigTrak moving forward 1-ft, and FC

develops the hypothesis that the numbers refer to feet. ("That's about a foot. Maybe that's how it works."). He tests this by the experiment at FC108-110. Because he did not press **CLR**, BigTrak goes forward 3 ft, but since the floor is unmarked, FC accepts this as confirming his prediction (" . . . couple of feet, okay.").

Next, FC moves on to explore the ← key. He extends his recently acquired knowledge of the semantics of the numbers for forward movement to the leftward direction and expects that BigTrak will do something like "turn left then [move 1 ft?] . . .". Instead, BigTrak moves forward 3 ft (the preceding program hasn't been cleared) and makes a very small left rotation. It is not clear whether or not he detects this slight rotation, but what he does recognize is that ". . . it did the same thing I told it to do last time . . ." and he tries to come up with an account for this: " . . . even though I pushed different buttons . . .".

At this point he forms an entirely new hypothesis: that the numbers are line numbers (as in BASIC) and that the arrow keys have built-in 1-ft semantics. There are several sources of evidence for our inference about what FC knows about BigTrak at this point in the protocol. We will get to these later episodes shortly, but it is worth describing the protocol support for our inferences at this time.

> In subsequent episodes, FC instructs BigTrak with monotonically increasing numerical arguments: following the first command is a **1**, following the second, a **2**, and so on (see FC122-127 and FC131-139).
>
> Instead of accomplishing a specified distance with a single instruction of ↑ *n*, he uses a sequence of ↑ commands. He might as well use ↑ **6** to obtain the result in FC156-164; and →**6 for** →**1**→**2**→**3** in FC130-136, if these are what he has in mind.
>
> In FC165 he is surprised and puzzled by BigTrak's behavior. His verbalization trails off and he pauses for a considerable time. The experimenter interrupts to ask him to talk more (FC170).
>
> In FC170-173, when he learns the correct semantics of the system, he says, almost explicitly, that the new knowledge is different from what he had previously thought.

As a result of this learning episode, FC starts to use **CLR** consistently before instructing the toy, in order to be sure that the previous program is cleared.

In FC122-127 he sets out to verify his most recent inference. Note that he still thinks that the right/left arrows make the tank turn and move 1 ft. The program →**1** ↑ **2**, which he believes will make BigTrak move to the right 1 ft and forward 1 ft, instead makes it turn six degrees to the right and move forward 2 ft. FC interprets this as if BigTrak were moving along the vector resultant of adding together the two orthogonal 1 ft vectors (" . . . that's the resultant thing maybe . . ."). This analysis is reasonable, given the data, although it is not correct.

FC expresses some doubt about this ("I don't know."), and he runs an experiment to verify this hypothesis (FC130-139). Again he is lucky, but this time BigTrak turns right 36° ($6 \times (1+2+3)$) and moves forward 4 ft. Again, not very good, but apparently close enough ("Okay, that's like over and or up three and over one or something . . ."), and he goes on to explore a new button: **FIRE** (FC146-149).

His exploration of the fire action succeeds without trouble and he goes on to exercise what he has learned by putting several commands into a program (FC155-164). In response to this program, BigTrak behaves in a way that cannot be reconciled (to FC) with his beliefs: it rolls forward 6 ft (hitting the far wall of the lab) and fires four times

There is an unusually long pause. Unfortunately, we can only guess what he is thinking between this behavior (after FC164) and the start of the next episode (FC165). He might have noticed the **FIRE 4** and the four firing behaviors of the toy. In any case, he ends up with the correct understanding of the semantics of numbers and the way that programming takes place ("... it just matters what order you put them in ...").

The two experiments in FC155-169 are two of the few cases in all of the protocols of what appear to be experiments which can differentiate alternative theories. Did FC construct these with that goal in mind or did he simply construct a test of the most recent hypothesis? It seems that at least the first one is meant to be a discriminating experiment since he says: "...if you push the two then it goes forward two *instead of*...". Unfortunately, he didn't finish the sentence.

5. Aspects of the instructionless learning process

After about 30 min of interaction, our subjects are able to construct a detailed device schema for BigTrak. In order to do so, they must bring to the session some general and powerful methods for hypothesis formation, experiment design, and result analysis. In this section we describe the parts of our theory of instructionless learning that relate to the data presented in the preceding sections.†

5.1. FORM OF HYPOTHESES

Subjects form hypotheses and design experiments to confirm them. They make predictions about how BigTrak should behave if their hypotheses are correct, and they replace the hypotheses in the face of contradictory evidence.

Hypotheses differ in their *degree of refinement*. Consider the following hypotheses, informally stated:‡

(1) The **RPT** key, followed by a number, *n*, will do something.
(2) The **RPT** key, followed by a number, *n*, will do something with respect to the steps before it.
(3) The **RPT** key, followed by a number, *n*, will cause the preceding *N* steps to be run again.
(4) The **RPT** key, followed by a number, *n*, will cause the preceding step to be repeated *N* times. (This is false.)

Hypotheses (1) and (2) contain unrefined terms (e.g. "something") and will lead to episodes whose goal will be to uncover behavior from which fully refined hypotheses like (3) or (4) can be constructed.

† A complete integrated theory of instructionless learning, as well as a computer implementation are presented in Shrager (1985).

‡ Although we describe hypotheses informally, it is clear that the processes of refinement and hypotheses formation will require a more formal representation. The details of the actual representations used by the implementation of our theory of instructionless learning are not relevant here. They are presented in Shrager (1985). Briefly, our characterization of the contents of a model bears strong resemblance to Moran's (1981) Command Language Grammar. That representation was meant to apply to non-programmable complex system interfaces, but it contained the notion of a script that is carried out by either party in the interaction (user or machine) and the important notion of a device model. It is not very difficult to extend the CLG to apply to programmable systems of moderate complexity, in particular, to most of our subjects' correct and incorrect models of BigTrak.

The *scope* of an hypothesis is determined by the range of things that can be bound to its nominal arguments. All of the above hypotheses refer specifically to the **RPT** instruction. The following are of broader scope:

(5) Some instructions (e.g. moves, turns, fires) require a numerical argument and some (e.g. clear last step) do not.
(6) All the direction keys (forward, backward, left, and right) take numbers that refer to the linear distance that they move in the direction indicated. (This is false.)

Hypotheses (5) and (6) are as well specified as (3) and (4). The variables in hypotheses (5) and (6) are set specifications, not unrefined variables. "Some" and "all" here refer to specific classes (e.g. "direction keys"). There are related hypotheses that specify class membership:

(7) The backward arrow key is of the same type as the turn arrows and forward arrow in the syntactic case of requiring a numerical argument.

Before the refinement process can start, some kernel of relevant knowledge must exist (a highly unrefined hypothesis). For example, one must at least know that **RPT** is of type "key", and that pressing keys will do something. An hypothesis containing unrefined variables leads to an episode (called an "exploration") which then provides data that enables the learner to refine the variables in the hypothesis. It may be necessary to run several explorations, obtaining several refinements, before having an hypothesis that can be confirmed by experiment. It is also possible for "mental refinement" to occur, without any overt exploration, if there is already sufficient knowledge to infer the values of particular terms (variable types, or their bindings). For example, hypothesis (1) could be refined to hypothesis (3) or (4) via a reasoning process (e.g. search) without requiring any explorations. Ultimately, an hypothesis containing no unrefined variables will lead to an experiment.

5.2. STRUCTURE OF THE PROCESS

Behavior during the systematic investigation phase consists principally of a standard cycle of hypothesize-experiment-observe. We have already given several examples of this cycle from FC's protocol and it is typical of all subjects, as illustrated by the following examples:

AM:
I think it makes a pattern [. .] **[CLR]** clear it. I want it to go [←] left. Then I'll push the left [←] [. . .] and a seven [7]. Okay, that's pattern number 7. Go. **[GO]**. It should go left . . . backwards . . . wait it's supposed to go [laugh] [. .] Okay. That wasn't quite right.
. . . so if I push two different numbers, I can get to do two different commands. So I'll try three commands and three numbers. [←] left [7] seven [. . .] **[GO]** it should go left, backwards, then go right. [S picks up the tank to stop it from moving.] It didn't go right yet.

CB:
Okay, I'm goanna make it try to turn left **[1 ← GO]**. No.
So C.L.S might mean that it, uhhh, it doesn't do its last maneuver . . . clear **[CLR]** forward [↑] [. .] c.l.s. **[CLS]** go **[GO]**. Okay, that's what it means.

FC:

Let's see. This button says go [**GO**]. Didn't do anything.
Let's see. [↑, **1, 0, GO**] should go straight . . . no . . .
If I want it to go forward then fire what do I do? I push [↑] forward [**FIRE**] I can't press that yet. One [**1**] fire [**FIRE, 2**]. I did that right. [**GO**]. Okay, it went forward and then fired twice.

RD:

[**CLR**] I want it to go right [→, 1] 1 ft [**GO**] Hmmm. [. . .] I expected it to go 90° to the right. [. . .]
[. . .] [**HOLD**] maybe it holds for a number of seconds. So, I try hold for 5 sec [**5**] followed by [↑] going forward for one more [**1**] foot. See if that helps. [**GO**]. And it did do what I thought it would do.
Looks like their degrees are set up [**CLR**] so that 20 is approximately equal to 90 real degrees [. . .] I'm gonna see if 20, 40, 60, 80 will bring it all the way around [. . .] rotating it [→] 80 [**8, 0, GO**]. Nope.

5.2.1. Calibration episodes

In one common class of episodes, subjects attempt to determine the effect of the numerical parameters that must accompany many of the commands. These *calibration* episodes produce two kinds of knowledge. The first deals with the functional relation between the command and the number: for example, numbers correspond to distance units for ↑ and to rotational units for ←. The second kind of knowledge deals with the measure of the unit: for example, the unit is tenths of a second for **HOLD** and feet for ↑. If knowledge about the functional relation is already correct, then the construction and evaluation of experiments to determine the unit measure is relatively straightforward. A few parametric variations quickly determine the unit, at least approximately. However, if the functional relation is incorrect, then calibration experiments lead to unexpected and occasionally confusing results.

AM's confusion about "patterns" and FC's confusion about the role of numerical arguments (as step numbers, rather than as parameters for a command) are examples of the complexity of the interaction between these two kinds of calibration experiments. Another example comes from RD, in the protocol segment shown in Table 5. In order to formulate hypotheses about the semantics of **RPT**, RD keeps revising his representation of the entity to which **RPT** applies. At first the entity is implicit: there is the notion of a single unit behavior of the device which is somehow a result of the things we have entered *to this point*. In RD319 he refers to this implicitly as the target of the repeat—the thing that is going to be repeated "one time". When this fails, he is forced to consider the components of that unit (the steps) and re-interpret the function of **RPT** (the behavior of BigTrak at RD320) in terms of these things rather than the whole sequence. His experiment, although not a critical test of the hypothesis, confirms his new understanding of the argument of **RPT** with respect to the structure of the toy's memory. Note that the simple semantics of **RPT** as "causing something to happen again" (to "repeat") was fixed by RD311 and did not change between RD311 and RD329. What did change was the meaning of the argument referring to *what* to repeat.

FC treats **RPT** similarly. He first uses an argument (FC354) simply because he has to in order to get the toy to accept the **GO** command. Then he observes its behavior (with **RPT 1**) and infers that it is repeating the whole sequence twice. There are several

TABLE 5

RD studying RPT

RD306: + CLR	Clear it. Send it forward
RD307: + ↑	one
RD308: + 1	increment and RPT
RD309: + RPT	whatever that is, 5
RD310: + 5	fine . . .
RD311: + GO	I guess that means repeat, so I'm gonna
RD312: + CLR	check that assumption out I'm gonna make it go
RD313: + ↑	forward one
RD314: + 1	foot and rotate
RD315: + →	forty
RD316: + 4	
RD317: + 0	then repeat one
RD318: + RPT	
RD319: + 1	time.
RD320: + GO	So it should go forward 1 ft, turn around go forward . . . I guess the repeat repeats the number of steps previous to it. So if you hit RPT one it would repeat the step previous to it. Now I'm gonna see
RD321: + CLR	if RPT 2 would repeat two steps ahead of it so I'm going forward
RD322: + ↑	1 ft again and make it rotate
RD323: + 1	
RD324: + →	I tried 40 and it brought the thing around so now I'm going to try 38 to see if that's one revolution in the thing's idea of degrees. So
RD325: + 3	
RD326: + 8	then RPT
RD327: + RPT	
RD328: + 2	two . . . see if it does it
RD329: + GO	and it did what I thought it would do so I'm happy about that again. Okay

possible interpretations of this and he initially thinks that "Maybe the numbers correspond to the step . . .". He performs an experiment (FC367-378) in order to examine this analysis: **CLR ↑ 1 → 15 FIRE 2 RPT 2**. Before entering **RPT** in this experiment he indicates his expectation: ". . . and then I want it to turn right fifteen again." (FC374). This is not what takes place so he changes his hypothesis: "So it goes back to the second step and plays it all through again." This is still incorrect—the target of repeat, to him is still the step number in memory, not the number of preceding steps. FC makes several more attempts to determine the semantics of **RPT** and finally gives up.

5.2.2. Confounds and bias in experimental design

A consistent finding in studies of intuitive science (e.g. Mynatt, Doherty & Tweney, 1977) is that naive scientists are biased to perform experiments that confirm their hypotheses. This tendency is contrary both to the Popperian edict of hypothesis disconfirming experimentation (Popper, 1959) and the normative scientific philosophy of hypothesis discriminating experiments. Hypothesis discriminatory experiments are conducted when there are two hypotheses to discriminate (as in the example given in FC's segment), but our subjects rarely consider two hypotheses simultaneously. Rather,

they replace an old, apparently bad, hypothesis with a new one and then proceed to try to confirm the new one.

Various complications arise from the lack of care in controls and the failure to consider alternative hypotheses. For example, suppose that a subject is trying to to test hypothesis (4) above. He enters these instructions: [**CLR, ↑ 1, ↑ 1, ↑ 1, RPT 3, GO**]. BigTrak moves forward 6 ft, thus confirming the incorrect hypothesis. However, this behavior is also consistent with hypothesis (3), which is correct. A better test would have been something like: [**CLR, ↑ 1, ← 15, ↓ 1, RPT 3**]. One can not consider all possible alternative hypotheses, but as noted above, there is little indication that our subjects ever consider more than one at a time.

5.3. OBSERVATION AND INTERPRETATION

Hypotheses are evaluated by comparing a prediction with BigTrak's behavior. If the prediction matches what happened, the hypotheses is accepted. However, if something happens which was predicted, than the learner might do one of the following:

Reject the behavior and repeat the experiment.

Note that the behavior is incorrect, but otherwise ignore it, reject the hypothesis, and try to form another one to replace it, from what is already known; or simply abandon the line of consideration and move on to resolve some other aspect of the schema.

Undertake an analysis of the actual behavior of the device, and form new hypotheses that are based on that analysis. This is the most common, and most interesting result of the failure of a prediction. This explanation process and hypothesis refinement are the two principle sources of schema change.

What does the learner observe in a mispredicted behavior? The answer is complicated by observational biases. The observer is prone to interpret BigTrak's behavior as confirming the prediction. As a result, he tends to encode nearly correct BigTrak behaviors as if they indeed matched the prediction. (Sometimes this leads to slight distortion of the original prediction to fit the behavior). A more fundamental error results when the observer uses the wrong analysis to parse BigTrak behavior. FC's vector model, discussed in section 4, is a perfect example of this sort of difficulty. His experiments are designed to confirm his vector addition expectations, and when they merely come close (e.g. FC122-144), he seems to accept the confirmation.

Confirmation bias produces memory errors as well as encoding errors. Since the learner was not expecting what actually did happen, he is unlikely to remember it veridically when it comes to analysis. In fact, large chunks of behavior seem to be either forgotten or ignored by subjects in analysis of failures. However, none of these biases or observation errors pose a theoretical problem. They simply act to confuse the subject for a time, by seeming to confirm possibly incorrect hypotheses.

Note that one cannot observationally distinguish hypotheses that are actually ambiguous with respect to the obtained behavior. For instance, because BigTrak's velocity is about 1-ft/sec, its behavior in response to the program: [**CLR ↑ 10 GO**] matches both the prediction that it will travel 10 ft and that it will move for about 10 sec. This difficulty serves to confound episode interpretation. It is not an observational difficulty.

5.4 THE ROLE OF THE DEVICE MODEL

In general, learners invoke the device model in order to explain the causes of unexpected behavior. For example, memory is offered as an explanation when BigTrak initially performs the same steps as those in the preceding episode, when different buttons have been pressed in the present episode. The principle use of the model seems to be in supporting inferences and suggesting the means of accomplishing goals that are internal to the system.

A part of our subjects' intuitive science seems to be a desire to "wash the test tube" before conducting an experiment. Before learning about memory, they can do this by turning BigTrak off and on. However, once they learn about memory, they almost uniformly realize that **CLR** is the way to clear that memory. (Often they have been using **CLR** before in this capacity, but inconsistently. When they realize the relationship between **CLR** as mediated by the memory model, they are more consistent with its use. We saw this in the detailed analysis of FC's learning segment in section 4.)

Another example of an inference that is supported by a device model is FC's hypothesis (FC320, FC332-336) that since **CLS** means to take one step off the program stack, it might be the case that one can repeat this operation until all the program steps are gone. He tests this experimentally (FC341-342).

It is clear that learners do not always think in terms of the device model, because they are all able to get BigTrak to behave long before the point at which they are likely to have anything resembling a device model. Apparently, TE *never* obtains a device model. Although she is not a terribly successful learner, she is able to make BigTrak turn, fire, and execute multiple-step programs. Furthermore, when we asked subjects, in the post-task interview, to tell us about BigTrak, almost all of their explanations consisted of the rules of interaction: "How-to"; rather than explanations of the knowledge contained in the device model—knowledge that they held at some earlier point during the learning session. Finally, it is important to recognize that almost everything that one normally does with BigTrak can be done either by reasoning in the domain of action plans, or the domain of the device model. Even **RPT** and **CLS** instructions can be reasoned about in terms of the actions that BigTrak will undertake in the real world. (**CLS** removes the last action and **RPT** copies actions to the end of the program.)

6. Conclusions

We have demonstrated that adults with no prior exposure to a particular complex system can learn about most of that system without instructions or assistance. We have attempted to account for their performance by proposing some of the underlying mechanisms. Learners acquire a schema for BigTrak that consists of hypotheses about the syntax of interaction, the semantics of various functions, and the contents of the device which mediate interaction. Most learners infer a device model and then use it to help them predict and coordinate aspects of the schema—to generate rules of interaction and problem solving for BigTrak. Although the details of the BigTrak schema are specific to this device, syntactic, semantic, and model knowledge are likely to play a role in most reasoning about complex systems (e.g. Moran, 1981; Young, 1981, 1983).

In this concluding section, we address some broader issues raised by this investigation. First, how much of subjects' behavior does our analysis account for? Second, how general are these results with respect to other sorts of interactions with devices? Finally, we have seen that instructionless learners are particularly bad scientists—at least according to normative principles of scientific reasoning. How, then, do they manage to succeed at this task?

6.1. GENERALITY

We believe that the mechanisms of hypothesis construction and the planning and performance of experiments will be found in all instructionless learning contexts. In the one studied here, the rate of interaction is high and the cost of interaction is low. If one were to raise the cost, we expect the quality of the process to change, but not its basic components.

The particulars of the model content (the use of a device model and the form of what is learned) are probably specific to complex devices. (We include most non-general programmable systems in this set: bank tellers, calculators, text editors, BigTraks, etc.) General purpose programmable systems probably also include a model, syntactic, and semantic elements, but the generality of their language, and the complexity of the model contents probably make them rather different in content from the present domain, and fundamentally more difficult to learn for someone without richly applicable prior schemas to draw on.

6.2. COVERAGE

We have observed an unusually rich piece of learning and problem-solving behavior. How much of it have we been able to account for? In answering this question, we make a distinction between "covering" the behavior in terms of being able to map it into taxonomy, and "covering" it in terms of being able to account for the cognitive processes that generated it.

6.2.1. Parsing the protocols

For the two fully analysed protocols (AM and FC), we were able to account for virtually every piece of behavior in the systematic investigation phase. That is, given the objective criteria presented earlier, we were able to categorize every utterance and keypress into an episode, and we can infer the hypotheses that initiated them.

Preliminary analysis of the remaining protocols indicates that, in all but one case, we can get the same coverage for them, too. The exception is TE. At several points during the systematic investigation phase, she appears to abandon all current goals, and falls back into searching for beep-producing keystrokes. Our current scheme is unable to parse this kind of behavior.

6.2.2. Invisible hypotheses

The formation and refinement of hypotheses constitutes the core of our learning process. However, most hypotheses are not explicitly mentioned in the protocols: we infer them from the broader context of the subject's behavior. Assessing the "coverage" of the theory with respect to hypothesis formation poses some difficulties. First, we only count hypotheses which are followed by an experiment, but there remains the possibility that the learner briefly considers and rejects hypotheses that never have any surface

manifestations. Second, our analysis of what particular hypotheses generated a piece of behavior is under determined, since, in principle, there can always be some other set of hypotheses or some other mechanisms that might generate the same specific behavioral episode. That is, it is always possible that, in a given episode, a subject simply ignores the behavior of BigTrak and starts to work on an hypothesis that coincidentally leads to behavior that we interpret as an attempt to explain a failed prediction.

However, we are fairly sanguine about this "in principle" indeterminacy. For one thing, we have discovered that our seven subjects assert the same small set of things about BigTrak, and they tend to run similar experiments to refine similar hypotheses (even though they vary widely in their verbalization of such hypotheses). This leads us to believe that subjects are indeed attending to the BigTrak behaviors that we assume are driving their refinement process. If we were very wrong here, we would expect to see much greater individual differences in erroneous hypotheses, as well as in the time it takes to induce the correct device schema. We conclude that the coverage of hypotheses is fairly good. We are able to account for nearly all of the hypotheses made by FC and AM during the systematic investigation phase and support their existence with an an analysis of button pressing in addition to verbalizations. Additionally, preliminary analysis of the remaining protocols appears to maintain the same level of coverage.

6.3. WHY INSTRUCTIONLESS LEARNING WORKS

Aspects of our model of instructionless learning resemble "intuitive" scientific reasoning (Nisbett & Ross, 1980). Subjects evaluate predictions derived from their hypotheses by experimentation. Explorations are used to refine hypotheses. Parts of the schema that have already been acquired are used as "tools" to address aspects whose semantics are not directly observable (e.g. aspects of the device model). However, in many regards instructionless learners are very poor scientists. They entertain mutually inconsistent hypotheses, they design primarily confirmatory and confounded experiments, they make some gross observational errors, and they form conclusions on the basis of inadequate evidence. Nevertheless, most subjects manage to master BigTrak in a reasonable amount of time. Several properties of the device, the human information processing system, and our particular context all contributed to the general success of our subjects:

BigTrak follows a relatively simple model of computation. Even though our subjects were mostly computer-naive, most of them had little trouble understanding this model, once they came upon it, and it was not far from the first model that they suggested. The notion of a stack memory and running off the contents of that stack seem to match naive models of how something like BigTrak might work.

The interface is suggestive of the semantics of the components. The keys are labeled reasonably. Although we have seen that it is possible to misinterpret these labellings (e.g. FC's vector model), they do help trim the search space.

The semantics of BigTrak are relatively decomposable. The use of a simple device model is sufficient to enable users to reason about the features of the system that are not decomposable. These features minimize the potential combinatorics that the learner would otherwise face.

BigTrak's interface is responsive in the ways that the learner needs to accomplish his task. The beep feedback and simple releasing procedures (**GO** and **CK**) permit the learner to obtain the results of experiments easily.†

Our analysis holds that alternative possible explanations and contradictory evidence are not considered carefully by subjects in forming hypotheses. This has the positive effect of lessening working memory load which has been linked with error-prone performance in some complex domains (e.g. Anderson, Farrell & Sauers, 1984).

Even with all of these advantages, we have seen that the learning process is fraught with error. One of the most important aspects of our instructions learning situation—which permits the learner to succeed in the face of these problems— is the high rate of interaction. This high interaction rate results from the low cost of experimentation with BigTrak. Subjects can afford to hold numerous incorrect hypotheses because they can evaluate a large number of them in a relatively short time.

People succeed in learning about a complex system without instruction, even in the face of many obstacles. They bring powerful reasoning mechanisms to bear on the problem. The present work begins to provide an empirical basis for formulating a coherent theory for this interesting and important phenomenon.

This work was supported in part by an IBM Research Fellowship granted to the first author, and by grants from the National Science Foundation (BSN81-12743) and the Spencer Foundation to the second author. We are indebted to our colleagues John Anderson, Tom Moran, Allen Newell, Herbert Simon and numerous others for discussing many of the ideas in this paper with us. Thanks, also, to Danny Bobrow, David Hill, Kurt van Lehn, Clayton Lewis, Jay McClelland, Matthew Lewis, Peter Pirolli, Don Norman and many reviewers for comments on previous drafts of the manuscript. Finally, we thank Jill Larkin for first suggesting that we use BigTrak as context for research, and Margaret Kinsky for her meticulous transcription of the protocols.

References

ANDERSON, J. R., FARRELL, R. & SAUERS, R. (1984). Learning to Program in Lisp. *Cognitive Science*, **8**, 87–130.

BOTT, R. A. (1979). A study of complex learning: theories and methodologies. University of California at San Diego, Center for Human Information Processing. Report no. 7901.

ERICSSON, A. & SIMON, H. A. (1984). *Protocol Analysis: Verbal Reports as Data*. Cambridge, Massachusetts: MIT Press.

GENTNER, D. & GENTNER, D. R. (1983). Flowing waters or teeming crowds: mental models of electricity. In GENTNER, D. & STEVENS, A. L. Eds, *Mental Models*, pp. 99–129. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

HALASZ, F. G. (1984). Mental models and problem solving in using a calculator. *Ph.D. thesis*, Stanford University.

HALASZ, F. G. & MORAN, T. P. (1983). Mental models and problem solving in using a calculator. *Proceedings of the Second Annual Conference on Human Factors in Computer Systems*, pp. 212–216. New York: Association for Computing Machinery.

KIERAS, D. E. & BOVAIR, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science*, **8**, 255–273.

MORAN, T. P. (1981). The Command Language Grammar: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, **15**, 3–50.

† As a commercially available children's toy. BigTrak was designed to be easily learnable, and our results confirm the intuitions of its designers. (Although they do provide a 25-page instruction manual.)

MYNATT, C. R., DOHERTY, M. E. & TWENEY, R. D. (1977). Confirmation bias in a simulated research environment: an experimental study of scientific inference. *Quarterly Journal of Experimental Psychology*, **29**, 85–95.

NISBETT, R. & ROSS, L. (1980) *Human Inference: Strategies and Shortcomings of Social Judgement*. Century Psychology Series. Englewood Cliffs, New Jersey: Prentice Hall.

POPPER, R. (1959). *The Logic of Scientific Discovery*. London: Hutchinson.

SHRAGER, J. (1985). Instructionless learning: discovery of the mental model of a complex device. *Ph.D. thesis*, Carnegie-Mellon University, Dept. of Psychology.

YOUNG, R. M. (1981). The machine inside the machine: user's models of pocket calculators. *International Journal of Man-Machine Studies*, **15**, 51–85.

YOUNG, R. M. (1983) Surrogates and mappings: Two kinds of conceptual models for interactive devices. In GENTNER, D. & STEVENS, A. L. eds, *Mental Models*, pp. 35–52. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Appendix I: FC's raw protocol

001:	I'll just turn it on here [S: Switches BT on]
002:	Let's see. This button says go.
003: GO	Didn't do anything. Ok.
004: +>	I'll push this thing.
[0:30]	
005: >	
006: >	
007: GO	Doesn't do anything. Ok.
[1:00]	
008: CK	
009: RPT	
010: +CLS	Try them all nothing happens
011: +HOLD	
012: ↑	
013: >	
014: V	
015: <	[Exp: what are you thinking about?] I was thinking maybe
016:	if I push
017:	one of these [indicates green buttons] and then go it
018:	would do what I said, but it hasn't done that yet
[1:30]	
019:	so I'm pushing anything
020:	to try to get it to do something
021: RPT	
022: GO	
023: ↑	I'll try the numbers.
024: 0	
025: 0	
026: GO	
027: >	
[2:00]	
028: +CLR	Like a calculator clear I guess start over again
029:	push a direction
030: +↑	and then
031: GO	go . . . and nothing happens
032: HOLD	
033: +CLR	Clear again. Now one
034: 1	one and then
[2:30]	
035: +>	a direction
036: GO	
037: +CLR	
038: +CLR	

039: +FIRE All the buttons are right here, right? [Exp: yes]
040: okay. [Exp: And
041: you can ignore the ones covered with white stuff.]
042: CK
043: GO
044: RPT
045: +CLS
046: CLS
047: CLS
048: GO It only
[3:00]
049: buzzes the first time you hit it.
050: +CLR
051: CLS
052: CLS
053: +CLR
054: +<
055: FIRE
056: +4
057: +4
058: +GO
{<264}
[3:30]
059: Forty-four makes it turn around. [Exp: Put it back in
060: it's . yeah.] Okay. That's something
061: +CLR
062: 0
063: 0
064: GO
065: Okay wait a second. Hit this one
066: +> and then 44
067: +4
068: +4 and then go
069: +GO
{>264}
[4:00]
070: [Exp: What are you thinking about?]
071: So I hit the arrow that way to
072: turn that way. Forty-four made it do something
073: Alright
074: +CLR Clear to start over
075: +> Right. 10
076: +1
077: +0
{>60}
[4:30]
078: +GO Maybe its degrees or something. I don't know.
079: +CLR
080: +> Lets try 90 maybe it'll turn all the way around.
081: +9
082: +0
083: +GO
{>540}
[5:00]
084: Okay its not degrees
085: Let's see
086: +↑
087: +1
088: +0
089: +GO
{>540↑10}
[5:30]

090: Should go straight no [Exp: You can turn it
091: off to stop it]
[6:00]
092: [off] Okay [S turns tank off.] It just sort of took off there
093: CLR [S turns tank back on]
[on]
094: +CLR Try that again
095: +↑
096: +1
097: +0
098: GO
{↑ 10}
[6:30]
099: It went straight that time [S turns tank off]
100: [off] Alright that makes it go straight
101: Maybe if I push I don't want it to go
102: that far I could push
103: ↑ [S tries to turn tank on but tries the wrong direction on the
104: [on] switch Exp corrects him He switches it on]
[7:00]
105: +↑
106: +1 one
107: +GO That s about a foot maybe that's how it works
{↑ 1}
108: +↑
109: +2
110: +GO
{↑ 3}
[7:30]
111: couple feet okay
112: +←
113: +1 should turn left then . .
114: +GO [Exp: what are you thinking about?]
{↑ 3 ← 6}
[8:00]
115: Does it I don't know maybe it remembers things or
116: something so that it just did the same thing I told it
117: to do last time even though I pushed different buttons
118: +CLR
119: +CLR Alright I guess you can like oh I see (?) program
120: steps into it or
121: something like that So if you push ummm
[8:30]
122: +CLR
123: +> right one
124: +1 then forward
125: +↑ two
126: +2
127: +GO Went straight and right a little bit
{> 6 ↑ 2}
128: Oh I see that's the resultant
[9:00]
129: thing maybe I don't know
130: +CLR
131: +>
132: +1
133: +>
134: +2
135: +>
136: +3
137: +↑
138: +4

139: +GO
{>36↑4)
[9:30]
140: Okay that's like over three or up three and over one or
141: something like that
142: Okay I don't know how these things
143: over here work [S indicates the right-hand column of
144: buttons]
145: This is a fire [indicates fire].
[10:00]
146: +CLR
147: +FIRE fire
148: +1 one
149: +GO Okay so you push fire so you match up a command
{FIRE 1}
150: either a direction.
151: or one of these things with a number and it does them in
152: order I guess
153: So you have . . if you want it to go forward
154: three units or something and then fire you'd go
[10:30]
155: +CLR clear I guess to start it over
156: +↑ forward one
157: +1 forward
158: +↑ two
159: +2 forward
160: +↑
161: +3 three fire
162: +FIRE four
163: +4
164: +GO
{↑6 FIRE 4)
[11:00]
165: where does it [mumble]
166: +CLR
167: +↑
168: +2
169: +GO
{↑2)
[11:30]
170: [Exp: you are going to have to talk more.] Okay I think it
171: works with the command then if you push the two then it
172: goes two forward or something like that instead of
173: it just matters what order you put em in
174: So let's try
175: +FIRE
176: +1
177: +↑
178: +1
179: +GO
{↑2 PAUSE)
{FIRE 1↑1)
[12:00]
180: [Note: S hit the white OUT button when he picked it up
181: after the preceding behavior so when he hit go this time
182: it went forward 2 then stopped for about 5 sec then
183: fired then went forward the last one]
184:
185: Okay. well it didn't fire at all so I guess I just cancelled
186: that one out so I guess [Note: Now it goes the remainder]
187: Oh! wait a minute . [laugh]
[12:30]

188: Okay ... wait we go forward
189: +↑ ... wait
190: +CLR
191: +↑ forward and then zero
192: 0
193: 0
194: 0 doesn't like that
195: +CLR forward then one
196: +↑
197: +1
198: +GO
{↑ 1}
[13:00]
199: Okay it went forward then one when I pushed forward one
200: +CLR Clear that If I push fire one
201: +FIRE
202: +1
203: +GO It fires once
{FIRE 1}
204: +CLR If I push fire two
205: FIRE
206: +2
207: +GO It fires twice
{FIRE 2}
208: +CLR
[13:30]
209: If I push forward one
210: +↑
211: +1 [Note: S begins to go for FIRE but presses CK.]
212: +CK [Exp: What are you thinking?]
{↑ 1}
213: I'm trying to figure out what this ck
214: button does. I can push it
215: +CLR I'll worry about that later, for a second (?)
[14:00]
216: If I want it to go
217: forward then fire what do I do? I Push
218: +↑ forward
219: FIRE I can't press that yet. one
220: +1 fire
221: +FIRE
222: +2 I did that right.
223: +GO Okay it went forward then it fired twice.
{↑ 1 FIRE 2}
[14:30]
224: +CLR I guess push clear every time to ...
225: GO
226: GO
227: GO
228: GO yeah that clears everything okay.
229: +↓ reverse one
230: +1 fire
231: +FIRE two
232: +2 reverse
233: +↓ three
234: +3
[15:00]
235: +GO Okay that's cool. I pushed them in order I
{↓ 1 FIRE 2 ↓ 3}
236: pushed backwards then fire
237: then backwards it went backwards fired then went
238: backwards again

239: I push
240: +CLR
241: +FIRE fire
242: +5
[15:30]
243: +GO
{FIRE 5}
244: [Exp: How do turns work?]
245: Turns? Well Let's see, I push left
246: +←
247: +1
248: +GO [mumble]
{FIRE 5←6}
[16:00]
249: I think one something
250: +CLR It had all the other ones I put in there
251: before it did that
252: fired five times
253: +CLR
254: +←
255: +2
256: +GO That's a little more
{←12}
257: +CLR Let's see if I can get it to turn 90° or whatever
[16:30]
258: +←
259: +1
260: +0
261: +GO Maybe it works like a clock, you know, minutes or something
{←60}
262: like that
263: +CLR try
264: +← left
265: +1 15
266: +5
267: +GO Okay that makes sense it turned 15 min I guess
{←90}
[17:00]
268: If I push
269: +CLR to the left
270: +← 30
271: +3
272: +0 it should turn around all the way back
273: +GO [Note: He predicts this before pressing GO]
{←180}
274: Okay so then if I want it to go like off in this direction
275: like one up and one over I'd push
[17:30]
276: +↑ forward . . no
277: +CLR
278: +CLR forward
279: +↑ one
280: +1 left
281: +← 15
282: +1
283: +5
284: +GO Okay so it went up one and then turned
{↑1←90}
285: If I want it to go over one then I add onto that
[18:00]
286: forward
287: +↑ one
288: +1 and that should add on to the end I think

```
289: +GO                Okay, got that working. Now. So what's cls do?
{↑1<90↑1}
290:                    Rpt.  repeat maybe
[18:30]
291: +RPT               So, if I do . . . if I add that on to the end. It might go
292:                    up one turn
293:                    and go over two
294: GO
295: GO                 It didn't do anything
296: +CLR
297: +CLR
298: +CLR               Hold . . . alright, let's try to get it to go
[19:00]
299:                    like left
300: +<                 10
301: +1                   minutes
302: +0                 Then forward
303: +↑                 two
304: +2                 then I want it to fire
305: +FIRE                three
306: +3                   times
307: +GO
{<60↑2 FIRE 3}
[19:30]
308:                    Alright. How about if I make it go up then come back?
309:                    It would be
310: +CLR                  up
311: +↑                 three
312: +3                    turn
313: +>
314: +3
315: +0                    30 min then forward
[20:00]
316: +↑                    three
317: +3                    I think that's what I pushed first.
318: +GO                   Good job.
{↑3>180↑3}
[20:30]
319:                    What are the rest of these buttons?
320: +CLR               Cls may be clear step. If I wanted it to go forward
321: +↑                 one
322: +1                    then fire
323: +FIRE                 What's fire one do?
324: +1
[21:00]
325: +GO                Okay it went forward one and then fired
{↑1 FIRE 1}
326:                    Okay so when I key them
327:                    in then
328: +CLR                   if I want it to go forward
329: +↑                 one
330: +1                    then fire
331: +FIRE                 one
332: +1                    and then say I change my mind and I want it
333: +CLS                  to go fire
334: +FIRE
335: +3                 three times
336: +GO
{↑1 FIRE 3}
[21:30]
337:                    So this button might just clear the step you just keyed in
338:                    or the last step that you have in there
```

339: This one . . [rpt] . . if
340: I push that twice . . I'll push that again
341: +CLS and then said go
342: +GO
{↑ 1}
[22:00]
343: Yeah it knocked off the fire step Okay That works
344: Repeat How does
345: that work. [Exp: What are you thinking about?] I'm trying
346: to work out this button here [rpt] and I . . . maybe if I make
347: it do something and then push that button
[22:30]
348: as the next step or something
349: +CLR try
350: +↑
351: +1 now I'm gonna push the button
352: +RPT
353: GO
354: +1
355: +GO [Exp: What are you thinking about?]
{↑ 2}
[23:00]
356: Maybe . . . okay it went forward two
357: I guess. When I pushed the repeat one So it went twice
358: as far as I told it the first time so
359: +CLR how about if I put fire
360: +FIRE twice
361: +2
362: +RPT repeat
363: +1
364: +GO
{FIRE 4}
[23:30]
365: Maybe the numbers correspond to the step You know like
366: +CLR
367: +CLR How about ummmmm I can go forward
368: +↑ one
369: +1 right
370: +> 15
371: +1
372: +5 then fire
373: +FIRE twice
374: and then I want it to turn right 15
375: +2
376: +RPT again, that was the
377: +2 second step So
[24:00]
378: +GO Oh it goes back to the second step and then plays it all
{↑ 1 > 90 FIRE 2 > 90 FIRE 2}
379: through
380: again Okay. I suppose I can handle that
381: So if wanted it to . . if I make it go
382: +CLR I want to make it go out and back and then out and back again
[24:30]
383: without having to go through it twice I can push
384: +↑
385: +3 fire
386: +FIRE
387: +1 have it
388: +> turn around
389: +3
390: +0 30° then forward . how many did I tell it to go?

391: +↑ three
392: +3 and then push
393: +RPT repeat from the beginning
394: +1 It should do it twice . . Hopefully
395: +GO
{↑3 FIRE 1>180↑6}
[25:00]
396: Wow . . it keeps going back Nope only repeated the first step It only repeated the first step
[25:30]
398: Not all of them
399: +CLR Something simpler Let's go forward
400: +↑ one
401: +1 and then turn
402: +>
403: +1
404: +5 15 and then fire
405: +FIRE twice
406: +2 then
[26:00]
407: +RPT repeat
408: +2 two Let's say . . . it should go out turn
409: fire and then turn again
410: +GO It fired anyway but then the other time it kept going but
{↑1>90 FIRE 2>90 FIRE 2}
411: it didn't go on after that so maybe it
[26:30]
412: fires anyway even though
413: even though it doesn't . . won't go forward
414: Let's say . . .
415: +CLR go forward
416: +↑ one
417: +1 then turn
418: +>
419: +2
420: +0 20 min Then go
[27:00]
421: +↑ one more
422: +1 then fire
423: +FIRE twice
424: +2
425: +GO Okay it did that Then what will this do? If I push repeat
{↑1>120↑1 FIRE 2}
426: +RPT two
[27:30]
427: +2 it should add on to the end of that . . .
428: +GO uhhh turn forward fires twice then it goes again and
{↑1>120↑1 FIRE 2↑1 FIRE 2}
429: fires again
[28:00]
430: so what does this button do?
431: +CLR forward
432: +↑ one
433: +1 fire
434: +FIRE one
435: +1 forward
436: +↑ one
437: +1 repeat one
438: +RPT
439: +1 from the beginning
440: +GO
{↑1 FIRE 1↑2}

[28:30]
441: Well it went forward one fired went forward another one.
442: Okay
[29:00]
443: We'll come back to that I can't figure that one out.
444: What does hold do?
445: +CLR
446: +HOLD
447: HOLD
448: HOLD
449: GO
450: +CLR
451: +HOLD
452: +1
453: +GO That makes it pause I think
{HOLD 0.1}
454: I'll try it again.
455: +CLR
456: +↑ forward
457: +1 one hold
[29:30]
458: +HOLD for two
459: +2
460: +↑ forward
461: +1 one
462: +GO Okay it paused for a second
{↑1 HOLD 0 2 ↑ 1}
463: I wonder if you can make it pause for a real long time.
464: +CLR Let's say hold
465: +HOLD
466: +1
467: +0 for 10.
468: +GO Okay.
{HOLD 1.0}
469: +CLR That makes it pause
[30:00]
470: What does this do? [indicates ck]
471: CK
472: +CLR
473: CK
474: CK
475: CK
476: CK "ck". [Exp: What?] I'm trying to think what ck could
477: stand for.
478: CK
479: 3
480: 3 It isn't a straight command like the rest of these things
481: 6
482: CK
483: 3 Cause it doesn't do anything.
484: +CLR Maybe its like this ..
485: +↑
[30:30]
486: +1
487: +↑
488: +2
489: GO [S tries to press CK as it's moving]
{↑ 3}
[31:00]
490: So I have these two left to
491: figure out. [ck rpt] I can basically do whatever I feel
492: like except for those two buttons.