Running head: A COMPREHENSION-BASED MODEL OF EXPLORATION

# A Comprehension-Based Model of Exploration

Muneo Kitajima
National Institute of Bioscience and Human-Technology, Japan

Peter G. Polson
Institute of Cognitive Science
University of Colorado

Muneo Kitajima is a cognitive scientist with an interest in cognitive modeling of various aspects in human-computer interaction; he is a senior researcher in the department of Human-Informatics of National Institute of Bioscience and Human-Technology, Agency of Industrial Science and Technology, Ministry of International Trade and Industry.

Peter G. Polson is a cognitive psychologist with interests in computer simulation models of skill acquistion and usability evaluation methods; he is Professor of Psychology and a Fellow of the Institute of Cognitive Science at the University of Colorado at Boulder.

## ABSTRACT

The LInked model of Comprehension-based Action planning and Instruction taking (LICAI) simulates performing by exploration tasks using applications hosted on systems with graphical interfaces. The tasks are given to the user as written exercises that contain no information about the correct action sequences. LICAI's comprehension and action-planning processes are based on Kintsch's construction–integration theory for text comprehension. The model assumes that comprehending instructions is a strategic process; instruction texts must be elaborated using specialized strategies that guide goal generation. LICAI comprehends the instructions and generates goals which are then stored in memory. The action-planning processes are controlled by goals retrieved from memory. Representations of goals that can guide exploration are restricted by the construction–integration architecture. The model predicts that successful exploration requires linking of the goal representation with the label on the correct object. The model is evaluated by comparing its predictions with results from an experiment al study of learning by exploration by Franzke (1994, 1995). We discuss the implications of LICAI for designing instruction materials and interface s that facilitate exploration.

# CONTENTS

# 1. INTRODUCTION

This paper presents a model, **LI**nked model of **C**omprehension-based **A**ction planning and **I**nstruction taking (LICAI),[1] of the cognitive processes involved in following incomplete instructions for using applications hosted on systems with graphical user interfaces. The instructions describe a task to be performed but do not contain any information about action sequences. Thus, a new user must perform the task by exploration. LICAI simulates the processes involved in comprehending the instructions and then generates, by exploration, an action sequence that performs the task.

A focus of recent research on skill acquisition in human–computer interaction has been on learning by exploration (Carroll, 1990; Howes, 1994; Rieman, 1994, 1996; Rieman, Young, & Howes, 1996). Experienced users of a given environment (e.g., Macintosh or Windows 95) learn new applications or extend their knowledge of applications they already know by task-oriented exploration. Formal training is rarely available, and there are usability problems with training and reference documentation (see Compeau, Olfman, Sein, & Webster, 1995). Most users prefer to acquire new skills by exploration as they perform new tasks relevant to their work (Carroll, 1990; Rieman, 1994, 1996).

A majority of current models of exploration are based on problem-solving

---

[1] When LICAI is pronounced like “lick eye” or “Lee CHI (χ)”, the pronunciation represents two-kanji character Japanese word, 理解, which means comprehension.

architectures like Soar (Newell, 1990; Rieman et al., 1996) or ACT-R (Anderson, 1993; Rieman, 1994; Rieman, Lewis, Young, & Polson, 1994). Many of these models attempt to account for the label following strategy (Engelbeck, 1986; Polson & Lewis, 1990; Franzke, 1994, 1995), which is one of the most frequently used problem-solving heuristics by users at all levels of expertise. Label following involves using the overlap between users' goals and labels on menus, buttons, and other interface objects to guide search during exploration. Interface objects with overlapping labels are acted on (e.g., by dropping a menu) in attempts to discover correct action sequences.

LICAI, in contrast, focuses on goal formation. Previous models of the label following strategy have taken goals as given and then described the resulting search behavior (e.g., Rieman et al., 1996). However, models of exploration should go further and define the processes that generate the goals that guide exploratory behavior. Normally, tasks are given to users through various forms: narratives, written instruction texts, help messages, graphic images, or combinations of these forms. Whatever form is used to specify the task, the user must comprehend the meaning of an initial task description and then formulate goals that guide interaction with the interface. In LICAI, comprehension (i.e., goal formation process) and exploration processes are modeled using the construction–integration cognitive architecture (Kintsch, in press). Goal-formation is simulated using a version of Kintsch's (1988) model of word problem solving. Exploration is simulated using an extension of Kitajima and Polson's (1995) display-based action-planning model.

The construction–integration architecture has evolved from a detail model of skilled, text comprehension—a highly automated collection of cognitive processes

that make use of massive amounts of knowledge stored in long-term memory (Kintsch, 1988, in press). This is a very different foundation from the other cognitive architectures that underlie other models described in this special issue. For example, one of the primary foundations of Soar (Newell, 1990) was the General Problem Solver (Ernst & Newell, 1969), a model of deliberate cognition (i.e., problem solving and action planning) in situations where the problem solver has limited background knowledge.

This paper is an extended version of Kitajima and Polson (1996) and includes new simulation results.

## 1.1 Outline of LICAI

What motivated us to develop LICAI — a comprehension-based model of exploration — was Franzke's (1994, 1995) experiments in which experienced Macintosh users were introduced to a new application. Participants had to learn the application by carrying out a series of written exercises by exploration. The instructions carefully described a task but did not provide information about an action sequence that would accomplish the task. LICAI simulates such users.

LICAI is a *synthesis* and *extension* of previous models that have been developed based on the construction–integration architecture (Kintsch, in press). The processes of comprehending task instructions to form goals are modeled using an extended version of Kintsch's (1988, in press, chapter 10) model of word problem solving. LICAI incorporates a goal selection process to deal with situations where the instructions describe several subtasks, and the reader must retrieve from long-term memory the goal for each task cued by information generated by the application interface. The retrieval process is modeled by a version of the Kintsch

and Welsch (1991) model of cued-recall. The retrieved goal controls the exploration processes which are modeled by an extended version of Kitajima and Polson's (1995) model of skilled, display-based, human–computer interaction.

The construction–integration theory, however, is an incomplete cognitive architecture (cf. Soar or ACT-R; see Newell, 1990, or Anderson, 1993, respectively) because it does not incorporate explicit mechanisms for learning and problem solving. Wharton and Kintsch (1991) and Kintsch (in press, chapter 11) outline how the theory could be extended. The action-planning models (Kitajima & Polson, 1995; Mannes & Kintsch, 1991) contain the elements of a problem-solving architecture, including goal formation, action selection, and the ability to react to the consequences of actions.

Although LICAI does not incorporate search mechanisms, our results do refine and extend Franzke's (1995) analysis based on the model of exploration (Polson & Lewis, 1990) underlying the Cognitive Walkthrough (Polson, Lewis, Rieman, & Wharton, 1992; Wharton, Rieman, Lewis, & Polson, 1994). Our simulation accounts for the initial success or failure of the goal-formation process, and we show that there are strong constraints on the exact goal form that enables the action-planning processes to generate the correct action sequence for an exercise. Thus, LICAI partitions instructions given to Franzke's participants into (a) those that enable the model to do the task with little or no trial-and-error search and (b) those that the model cannot perform because it cannot formulate effective goals from the instructions.

## 1.2 The Graph Task and Franzke's Experimental Procedure

Franzke (1994) and the simulation reported in this paper, as well as Kitajima and Polson (1995) and Rieman, Young, and Howes (1996), employed the following task: Graph a set of data contained in an application document as a line graph. The graphing task was divided into two subtasks, and Franzke used different instructions for each subtask. The first subtask was to create a default line graph. The instructions provided participants with the information necessary to perform the several steps necessary to generate the default graph including variable names and graph style. Franzke's participants, however, were faced with two problems: (a) extracting the information that would enable them to discover these steps, and (b) sequencing this information so that the steps were attempted in the correct order. The second subtask was to edit the default line graph created during the first subtask. The edits were to be done in a specific order, and the instructions were short and telegraphic. The third problem for participants was to make the inferences necessary to generate a comprehensible task description that could successfully guide exploration. The goal-formation and selection processes incorporated in LICAI solve these three problems posed by the instructions used by Franzke.

Franzke's participants were experienced Macintosh users who had never used a graphing application. They performed two isomorphic tasks in which different sets of variables and file names and graph styles were provided, and each task was

accomplished by using one graphing application—Cricket Graph I[2] or III[3] or one of two forms of EXCEL 3.0[4]—by exploration. If a participant had not progressed after 2 minutes on a particular step, he or she was given brief hints like “select line graph from the graph menu,” or “double-click on legend text.”

One group of Franzke’s participants were asked to accomplish the task using Cricket Graph I. The data are stored in a document file containing a spreadsheet whose columns are labeled with variable names. The task is to graph the data in the column labeled ‘Observed’ as a function of numbers in the column labeled ‘Serial Position.’ The initial step is to double click on the document icon to launch the application and display the data. The next step is to pull down the **Graph** menu and release on the item **Line** so that the dialog box with two scrolling lists appears (see Figure 1). The column labels are displayed in two scrolling lists, representing the x- and y-axis. ‘Serial Position’ and ‘Observed’ appear in both lists. The dialog box partially occludes the table, but the column of numbers labeled ‘Serial Position’ is visible in the background. To plot ‘Observed’ as a function of ‘Serial Position,’ participants must click on and highlight ‘Serial Position’ in the x-axis scrolling list and ‘Observed’ in the y-axis scrolling list. The final step is to click on ‘New Plot,’ causing a default version of the graph to be displayed. The remaining steps in the task involve editing this default version.

---

[2] CA Cricket Graph, version 1.3.2, 1989.

[3] CA Cricket Graph III, version 1.01, 1992.

[4] MS EXCEL, version 3.0, 1990.

Insert Figure 1 about here

## 1.3 Outline of the Paper

In Section 2, we describe Kintsch's (1988, in press) construction–integration architecture and its applications to several cognitive processes, and their relations to LICAI. We introduce a single construction–integration cycle and then describe several processes defined by multiple construction–integration cycles. These processes include text comprehension (Kintsch, 1988), word problem solving (Kintsch, 1988), memory retrieval (Kintsch & Welsch, 1991), action planning (Mannes & Kintsch, 1991), and skilled, display-based human–computer interaction (Kitajima & Polson, 1995). Section 3 describes LICAI by tracing a simulation of Franzke's (1994, 1995) task. In Section 4, we evaluate LICAI using Franzke's (1995) results, and in Section 5, we review recent models of exploration in human–computer interaction in light of our results. In Section 6 we summarize the implications of our results for practice, especially those for the design of interfaces and training materials that support exploration (Carroll, 1990; Wharton et al., 1994).

# 2. THE CONSTRUCTION–INTEGRATION ARCHITECTURE

The cognitive processes specified in LICAI are implemented using the construction–integration architecture (Kintsch, 1988, in press; Wharton & Kintsch, 1991). As we said earlier, LICAI is a synthesis and extension of several models that have evolved from Kintsch's (1988) original model of text comprehension. This section presents a general introduction to the construction–integration architecture and the various models realized using the architecture. Section 2.1

provides an overview. Section 2.2 describes basic ideas underlying the construction–integration architecture. Section 2.3 focuses on the cognitive processes modeled on the construction–integration architecture and describes how these processes are incorporated into LICAI.

Our treatment is very different from Kintsch's (in press). We characterize the basic construction–integration cycle as roughly analogous to the decision cycle of Soar (Newell, 1990), and regard the various models as special cases of this general process. Soar views all cognitive behavior as problem solving, search through a problem space. However, the models described in this section actually evolved from a model of skilled reading. The title of Kintsch (in press) is *Comprehension: A paradigm for cognition*. He views LICAI and the Kitajima and Polson (1995) model of skilled performance as generalizations of a theory of comprehension.

## 2.1 Overview

The basic process assumed by the architecture is a two-phase, construction–integration cycle. The cycle is a cognitive process that manipulates knowledge represented as propositions. Three ideas characterize the construction–integration cycle. First, a cycle selects between alternatives. Thus, the construction–integration cycle is roughly analogous to the decision cycle of Soar (Newell, 1990), which selects among alternative operators. However, Soar uses different processes and representations.

Second, there are different construction–integration cycles defined by various models:

- Text-comprehension cycles select between alternative interpretations of sentences (Kintsch, 1988). (Section 2.3.1)

- Problem-model construction cycles, a strategic form of text-comprehension cycles, generate representations that control solution of a problem described in a text (Kintsch, 1988). (Section 2.3.2)
- Memory-retrieval cycles select one out of many possible representations from long-term memory in a cued-recall paradigm (Kintsch & Welsch, 1991). (Section 2.3.3)
- Action-selection cycles select between alternative actions (Mannes & Kintsch, 1991). (Section 2.3.4)
- Attention cycles select a subset of the information in a complex visual display for further processing by action-selection cycles (Kitajima & Polson, 1995). (Section 2.3.5.3)

Third, many processes involve multiple construction–integration cycles. These processes may include the same cycle or a series of different cycles. For example, reading a short paragraph is modeled as a series of text-comprehension cycles (Kintsch, 1988). Reading a text describing a word problem and building a representation that controls successful problem solving is simulated as a series of problem-model construction cycles (Kintsch, 1988). Skilled performance of a task using a computer with a graphical user interface is simulated as a series of pairs of attention and action-selection cycles (Kitajima & Polson, 1995).

LICAI simulates the processes involved in reading instructions and performing a task using three types of cycles: problem-model construction cycles for generating goals, memory-retrieval cycles for selecting a goal, and one or more pairs of

attention-action-selection cycles for finding an action sequence that satisfies the selected goal.

## 2.2 A Single Construction–Integration Cycle

A construction–integration cycle consists of a *construction* phase and an *integration* phase. The construction phase generates a network of propositions that contains the surface representation of a stimulus (text or visual display), alternative interpretations of the surface representation, possible actions, or other alternatives. The network also incorporates the knowledge necessary for selecting among the alternatives. This knowledge includes goals, information retrieved from long-term memory, and information carried over from previous construction–integration cycles.

Various models within the construction–integration architecture assume parsers that map surface representations into propositional semantic representations (text in Kintsch, 1988, visual displays in Kitajima & Polson, 1995). Generating alternatives is assumed to be a bottom-up, weakly constrained, rule-based process. The rules are not context sensitive; thus, the alternatives represented in the network may not be consistent with the current context.

The alternatives and all other information in the network are represented as propositions (Kintsch, 1974, 1988, in press; van Dijk & Kintsch, 1983). A proposition is a unit of knowledge that is represented as an ordered tuple. The first element is a predicate and the remaining elements are arguments of the predicate. For example, the goal “graph data” is represented by the proposition (PERFORM GRAPH DATA). The predicate PERFORM defines the proposition as representing a goal of the form *perform an action on an object* where the first argument is the

action and the second is the object. Propositions in the network are linked by their shared arguments. Thus, any proposition containing the arguments GRAPH or DATA will be linked to (PERFORM GRAPH DATA).

The integration phase selects an alternative by integrating information represented in the network generated during the construction phase. Integration can be thought of as a constraint satisfaction process. The network of interconnected propositions defines a collection of constraints that are satisfied by the selected alternative. Integration is performed using a spreading activation process. In this process, the nodes in the network can be partitioned into sources of activation, targets of activation, and links between sources and targets. Goals and the representation of the current context (i.e., text or visual display) are typical sources, and the targets are alternatives. The linking information comes from long-term memory and other sources, and the spreading activation process is controlled by the pattern of links in the network. When the integration phase terminates, the most highly activated alternative represents the result of the construction–integration cycle that satisfies the constraints. Because propositions in the network are linked by shared arguments, the pattern of argument overlap plays a key role in the results of the integration phase.

## 2.3 Specific Versions for Reading, Memory Retrieval, and Action Planning

This section summarizes various models that are based on the construction–integration architecture and that we modified and incorporated into LICAI. Our descriptions are based on abstract characterization of the construction–integration cycle presented in Section 2.2.

### 2.3.1 Text-Comprehension Cycles

The initial version of the architecture was a construction–integration model of text comprehension (Kintsch, 1988), and this model retained many basic assumptions from Kintsch's earlier work on text comprehension (Kintsch, 1974; Kintsch & van Dijk, 1978; van Dijk & Kintsch, 1983). Reading a text is simulated by a series of construction–integration cycles. During each construction phase, the model takes as input a representation of key elements of the text comprehended so far and a propositional representation of the next sentence or major sentence fragment. The model incorporates knowledge relevant to the input text as elaborations, which come from two sources: (a) propositions retrieved from long-term memory by a stochastic, associative retrieval process (Raaijmakers & Shiffrin, 1981), and (b) inferences generated by schemata triggered by propositions in the original input text. At the end of the construction phase, the network incorporates multiple interpretations of the input text.

During the integration phase, the model selects an interpretation of the input sentence that satisfies the constraints defined by the network. The sources of activation are the carried-over propositions and the propositions representing the input text. The targets are all the propositions in the network. The most highly activated subset of propositions in the network represent the reader's interpretation of the text. These nodes are typically the most highly interconnected subset.

### 2.3.2 Problem-Model Construction Cycles

One of the fundamental issues for a theory of comprehension is to show how readers can use information contained in texts to solve problems, or how users can perform tasks described in instructions. The text must be transformed into a

representation that can mediate successful problem solving or action planning. Kintsch and his collaborators (e.g., Kintsch & Greeno, 1985) have developed models of word problem solving that transform texts describing arithmetic or algebra problems into special forms that specify arithmetic or algebra operators. Typically, the reader must elaborate the original text with specialized inferences that guide problem solving. LICAI incorporates analogous comprehension processes that generate goals that control action-planning processes. Section 3.3 describes the processes in detail.

Kintsch (1988) showed that the construction–integration architecture can be extended to problem-model construction for word arithmetic problem solving by incorporating versions of assumptions made by Kintsch and Greeno (1985) and related work (Cummins, Kintsch, Reusser, & Weimer, 1988; Delarosa, 1986; Fletcher, 1985). Kintsch's (1988) word arithmetic model and the earlier work all incorporate the assumption that reading is a strategic process (van Dijk & Kintsch, 1983). Strategies generate inferences required to construct a representation that satisfies a reader's goals (e.g., solve a word problem). The knowledge used by the strategies is represented as *comprehension schemata*.

Kintsch (1988) incorporated the comprehension schemata needed to solve word problems by assuming that they operate during the construction phase by adding propositions to the network. These schemata elaborate the original text propositions by generating special interpretations of the text as well as representations that mediate problem solving. For example, one schema identifies noun phrases as sets, and comprehension schemata generate problem models by specifying alternative hypotheses about the role of each set in a problem description

(e.g., part-set, whole-set). Representations of all possible problem models are incorporated into the network. The alternative problem models are linked to other components of the network, including information about time, order of possession, location, and other aspects of the situation described in the problem text.

At the end of the integration phase, the most highly activated problem model is the one consistent with the situation described in the problem text. In the spreading activation process, the original text propositions are the sources of activation, and the alternative problem models are the targets. Propositions describing the situation link the original problem text with problem models if they are consistent. These supporting links define constraints that lead to the selection of one problem model. The selected problem model is then operated on by problem-solving mechanisms (arithmetic, algebra, or action planning) to generate a solution to the problem described in the text.

### 2.3.3 Memory-Retrieval Cycles

Franzke's (1994) instructions for the first subtask, creating a default graph, contained information used in several different steps required to perform this subtask. The comprehension processes incorporated into LICAI read the complete text before attempting to perform the subtask, and store goals and other information needed to perform the subtask in long-term memory. LICAI retrieves and sequences these goals using the displays generated by the application as the retrieval cues.

Kintsch and Welsch (1991) showed that the construction–integration architecture can be extended to account for memory for text. The model assumes that each text-comprehension cycle is followed by a process that transfers the

results of the comprehension cycle to episodic memory. Retrieval from episodic memory is modeled as a construction–integration cycle. During the construction phase, a retrieval cue is linked to the episodic memory network. During the integration phase, the retrieval cue (activation source) activates the nodes in the episodic memory network (targets). Kintsch and Welsch (1991) showed that the activation value of each node in the episodic memory correlates with the speed of recall of that node. LICAI uses a modified version of the Kintsch and Welsch (1991) model to simulate retrieval processes, described in Section 3.5.1.

### 2.3.4 Action-Selection Cycles

Mannes and Kintsch (1991) extended the construction–integration model of text comprehension to action planning using human–computer interaction as a task domain by assuming that text comprehension and action planning are similar tasks. Readers integrate information from diverse sources to select one out of many alternative interpretations of a text. Similarly, users, as action planners, must integrate their goals and information from diverse sources and select one out of many competing actions.

Mannes and Kintsch's action-selection cycle modified the basic construction–integration architecture by adding rule-like action representations, called *plan elements*. The condition of a plan element is a set of propositions that describe its preconditions. The action of a plan element is one or more propositions that describe the consequences of performing a plan element in a simulated task. These plan elements are incorporated in the network as nodes represented by object–action pairs (e.g., EDIT FILE^LETTER, a proposition that represents the action EDIT applied to a FILE named LETTER).

During the construction phase, a network is generated to include propositions that represent the task description (i.e., the user's goal), the task context (including the consequences of previous cycles), knowledge retrieved from long-term memory, and plan elements. These plan elements are generated by the following procedure: For each cycle, the model is given a list of three objects in the current task context, and each of the three objects is combined with all possible actions. The propositions in the network are then connected by shared arguments.

Mannes and Kintsch also made important modifications to the integration phase. During the activation process, the model spreads activation over the network using the same activation process described by Kintsch (1988); the activation sources are the propositions describing the task and the current task context, and the targets are the plan elements. A decision process operates after the spreading activation process. The model evaluates the conditions of highly activated plan elements in order of their activation values. A condition is true if all of its propositions are found in the network. The model executes the most highly activated plan element whose condition is true.

### 2.3.5 Attention Cycles Combined With Action Cycles

Mannes and Kintsch (1991) simulated a task environment where there were few objects in the world that were candidates for possible actions. Kitajima and Polson (1995) extended Mannes and Kintsch's (1991) model of action planning to display-based, human–computer interaction. They incorporated into their model a complex representation of a large format display containing tens of objects that are targets for possible actions.

Kitajima and Polson (1995) simulated skilled users who possess the knowledge necessary to correctly perform a task. They made two major additions to the Mannes and Kintsch (1991) action planning model. First, they assumed that skilled users have two level representations of familiar tasks. Second, they added an attention process that focuses the action-planning processes' attention on just three of the possible targets for action. Their model was mapped onto Hutchins, Hollan, and Norman's (1986) analysis of direct manipulation, which provides a framework to describe action planning as a goal-driven process. The action-planning process evaluates the consequences of the last action and then generates the next action to be executed (see Figure 2). They simulated the process of a user comprehending the display (Hutchins et al.'s, 1986, evaluation stage) by using the elaboration process of the construction phase.

Insert Figure 2 about here

In the next three subsections, we describe Kitajima and Polson's (1995) model of skilled human–computer interaction in more detail, and show how LICAI extends it to simulate an experience user attempting to generate correct action sequences by exploration.

*2.3.5.1 Task and Device Goals*

Kitajima and Polson's (1995) action-planning model assumes that skilled users have a schematic representation of the task in the form of a hierarchical structure involving *task goals* and *device goals* (Payne, Squibb, & Howes, 1990). They assumed that each task goal is associated with one or more device goals. A task goal is of the form "perform a task action (e.g., graph, delete, put) on a task object (e.g., data, word, variable)." One or more device goals are associated with

each task goal and specify states of specific screen objects that must be achieved to satisfy an associated task goal. Examples include specifying that a variable name in a scrolling list, a menu item, or the like, should be highlighted. Device goals control the action-planning processes.

However, LICAI simulates a new user of an application. LICAI assumes that the instruction comprehension processes generate task goals. The goal selection process retrieves a task goal from long-term memory and passes it to the action-planning processes. The action-planning processes must attempt to generate an action sequence that will accomplish the task goal without knowledge of the device goals. We assume that device goals are learned by interacting with the application.

#### *2.3.5.2 The Evaluation Stage*

Kitajima and Polson's (1995) action-planning model is given a representation of a new display in the form of a large collection of screen objects; each screen object is described by several propositions. These descriptions provide limited information about the identity of each object and its appearance, including visual attributes (e.g., color, highlighting). The model simulates Hutchins et al.'s (1986) evaluation stage (shown in Figure 2) by elaborating the display representation with knowledge retrieved from long-term memory. The retrieval cues are the task and device goals and the propositions representing the current display. The probability that a cue retrieves a particular proposition representing a piece of knowledge from long-term memory is computed by Raaijmakers and Shiffrin's (1981) model.

The propositions in long-term memory represent knowledge about the screen objects. For example, if Object23 is the scrolling list item labeled 'Serial Position,' then the following knowledge items are stored in long-term memory about

Object23: Object23 has-label Serial_Position; Object23 is-part-of Line-Graph-Dialog-Box; Object23 can-be-pointed-at; and Object23 can-be-selected. The elaboration process is stochastic, and Kitajima and Polson (1995) discussed in detail the predictions and implications that follow from this stochastic elaboration process.

*2.3.5.3 The Execution Stage*

Kitajima and Polson (1995) model the execution stage of Hutchins et al.'s framework (1986) modeled as a pair of an attention cycle and an action-selection cycle.

### 2.3.6 Attention Cycles

The attention cycle selects three screen objects as possible candidates for action to be carried-over to the succeeding action-selection cycle. During the construction phase, a network is generated consisting of nodes representing all screen objects, the task and device goals, the elaborated display from the evaluation stage, and candidate object nodes of the form "Screen-Object-X is-attended." During the integration phase, the sources of activation are the task and device goals and the screen objects, and the targets are the candidate object nodes. When the spreading activation process terminates, the model selects the three most highly activated candidate object nodes. These nodes represent screen objects to be attended to during the action-selection cycle.

The activation pattern is determined largely by two factors: (a) the links from the task and device goals to propositions in the network that share arguments with the goals, and (b) the number of propositions necessary to link goals to candidate

objects. As a result, the attention cycle selects candidate objects closely related to the task and device goals.

Device goals can directly specify a screen object and thus can be linked directly to the screen object represented in the network. Task goals can be linked indirectly to screen objects through labels. Continuing with the previous example, consider the following task goal and its associated device goal:

Task goal: perform 'plot Serial_Position on x-axis'

Device goal: realize Object23 is-highlighted

The correct candidate object node "Object23 is-attended" has a direct link with the device goal. On the other hand, the task goal can be linked to the correct candidate object node via a proposition in the form "Object23 has-label Serial_Position." Note that the device goal focuses the model's attention on the correct screen object. The task goal *may* or *may not* perform this function. The only way it can is for an element of the task goal representation to match the label on the correct screen object.

### 2.3.7 Action-Selection Cycles

The action-selection cycle works like Mannes and Kintsch's (1991); however, the actions are not commands but operations on screen objects. The action-selection cycle selects an action to be performed on one of the three candidate objects. During the construction phase, the model generates a network that includes the task and device goals, all screen objects, the elaborated display, and representations of all possible actions on each candidate object (i.e., plan elements). Examples would include 'single-click Object23,' 'move Object23,' and the like.

During the integration phase, the sources are the task and device goals and the screen objects, and the targets are the nodes representing plan elements. At the end of the integration phase, the model selects the most highly activated plan element whose preconditions are satisfied as the next action to be executed. The process is dominated by the same two factors as in the attention cycle. However, the relevant interaction knowledge must be retrieved during the evaluation stage satisfying preconditions for the correct action. For example, the model must retrieve from long-term memory the fact that objects in a scrolling list can be selected (i.e., 'Object23 can-be-selected').

LICAI's action-planning processes operate without device goals to simulate a user who is dealing with a novel application. However, if LICAI manages to focus on the correct screen object, it is likely to perform the correct action. Recall that LICAI simulates skilled Macintosh users. The model's knowledge of the Macintosh interface conventions enables it to identify the attended-to screen object and retrieve from long-term memory knowledge about correct action for this object.

## 3. LICAI

This section provides a detailed description of LICAI. The model integrates extended versions of problem model construction, retrieval, attention, and action-selection cycles to simulate goal formation, goal selection, and exploration. Task goals coordinate the operations of these cycles and control the three processes defined in LICAI. Section 3.1 describes requirements for the representation of task goals that can lead the action-planning processes to successful exploration. Then we explain LICAI by tracing a simulation of the task to create a default version of a line graph. Section 3.2 introduces the task and the instructions. Section 3.3 presents the

specialized comprehension schemata used for elaborating the text representations with propositions describing task goals. Section 3.4 describes the simulation of the goal-formation processes and shows how the instructions are processed by applying the comprehension schemata and how the results of comprehension are stored in memory. Section 3.5 describes the simulation of the exploration processes, including retrieval of the task goal and action-planning.

## 3.1 Exploration Guided by Task Goals

LICAI simulates a skilled user of the Macintosh interface learning a novel application. The goal-formation processes of LICAI generate possible task goals from the instructions utilizing the comprehension schemata. For experienced users, device goals control the generation of action sequences that satisfy their associated task goals. However, new users do not have device goals and successful exploration in LICAI is controlled solely by a task goal. It is assumed that device goals, which are descriptions of the states of specific screen objects, must be learned by successful interaction with the application.

Kitajima (1996) reported on a series of simulation experiments that determined the constraints on the exact representation of task goals that mediate successful action planning. First, he found that the action-planning model always generated correct actions when given correct device goals. This was true even when there was no task goal. A device goal specifies a screen object and a desired attribute of the object. For example, the variable name ‘Serial Position’ in the x-axis scrolling list must be highlighted to accomplish the task goal of “perform ‘plot Serial_Position on x-axis.’” The corresponding device goal is “realize Object23 is-highlighted,” where Object23 is the model’s representation of column label ‘Serial Position’ in

the x-axis scrolling list. The direct link between the device goal and the correct screen object caused the attention cycle to include the correct screen object in the list of the three candidate screen objects for the next action. When selecting an object–action pair during the action-selection cycle, the model usually chose the correct action on the most highly activated screen object. The specification of the desired attribute guides selection of the correct action because the model has knowledge of the consequences of an action on an object.

Second, Kitajima found that the model could generate the correct actions when given a task goal without device goals. However, the action-planning process was not as robust as when device goals are provided. In order for the model to make correct object–action selections, the task goal had to be specific enough to establish a link with the correct screen object. For example, the representation of the task goal "perform 'plot Serial_Position on x-axis'" has a link to Object23 through its label. This link enables the action-planning process to select the correct screen object during the attention cycle. However, an equivalent task goal, "perform 'plot Observed as_a_function_of Serial_Position,'" will not work because the screen label 'x-axis' was missing. Because there are three screen objects in Figure 1 with the label 'Serial Position,' the link to the label of the x-axis scrolling list is necessary for the model to attend to the correct screen object. Typically, the link between a task goal and the correct object is established through the screen object's label. A label is text that is directly associated with a screen object. Examples include a menu label, a button label, or an icon label.

In summary, Kitajima (1996) demonstrated that the action-planning processes could generate correct action sequences in the absence of device goals, although

there are strong constraints on the form of successful task goals. Links between the task goal and the screen object to be acted on at this step must exist, and these links are established by a screen object's label. The words used to describe a task must correspond to the label on the screen object to be acted on. This constraint forces LICAI to predict that label following will be the only exploration strategy available to new users of an application.

## 3.2 Experimental Task and Instructions

LICAI simulates an experienced user of the Macintosh performing the graphing task described in Section 1.2. Our analysis focused on the first task, creating the default graph. Figure 3 shows the instruction texts used in our simulation. These instructions are a simplified version of the instructions used by Franzke (1994), who incorporated more information about the first task. For example, she explicitly stated that the variable 'Serial Position' was to be plotted on the x-axis and the variable 'Observed' on the y-axis.

Insert Figure 3 about here

In the simulation, each sentence listed in Figure 3 was processed by a single problem-model construction cycle, described in Section 3.4.1. The numbers in the second column indicate the problem-model construction cycle number. At the end of the integration phase of each cycle, the original propositional representation of each sentence and all propositions generated by the comprehension schemata were transferred to episodic memory, described in Section 3.4.2. LICAI assumes that the user reads all instructions and then attempts to perform the task described in the text, and that the display is the retrieval cue for the task goal that controls action

planning for the current step of the task. Successive displays generated by the application serve as retrieval cues for the sequence of task goals.

## 3.3 Comprehension Schemata

LICAI assumes that goal-formation processes are analogous to Kintsch's (1988) model for solving word problems. This model assumes that reading is a strategic process (van Dijk & Kintsch, 1983) and that strategies generate inferences required to guide problem solving. The knowledge used by the strategies is represented as comprehension schemata. The goal-formation processes in LICAI take a semantic representation of the task instructions as input and elaborate this representation with inferences generated by specialized schemata to construct task goals. The task goals control the action-planning processes that generate exploratory behavior.

LICAI incorporates three kinds of schemata. *Global instruction reading schemata* represent the top-level strategy used by a reader to process text that describes a given task. All verbs with the implicit subject YOU are mapped into a text base proposition in the form DO [YOU, verb, object]. *Task-domain schemata* elaborate DO propositions and generate a more complete description of a task. For example, Franzke (1994) used telegraphic instructions for her editing tasks. Editing task-domain schemata elaborate these initial telegraphic descriptions. *Task-goal formation schemata* transform DO propositions into propositions that represent task goals.

The task-domain and task-goal formation schemata are discussed in detail next, including how they elaborate propositional representations of the sentences shown in Figure 3. This text has been transformed into a propositional

representation—the text base—using the methods described by Bovair and Kieras (1985). In addition, the global instruction reading schema has been applied to transform all propositions of the form VERB[object] into the form DO [YOU, verb, object].

### 3.3.1 Task-Domain Schemata

Task-domain schemata describe users' specialized knowledge of a task domain that enables them to elaborate the description of the task contained in the text. These schemata represent users' task formulation knowledge that is independent of a particular application interface. If users have no such knowledge, only the description of the task contained in the text will be transformed into task goals. We have assumed that users have some knowledge of graph and editing tasks. The editing task knowledge is a generalization of their experience with word-processors.

#### *3.3.1.1 Example From Data Graph Task Domain*

The elaboration processes performed by the task-domain schemata is illustrated by tracing the elaboration of Sentence 5: "Your first exercise is to plot the variable 'Observed' as a function of the variable 'Serial Position.'" The original input sentence is propositionalized as follows:

P51 EXERCISE

P52 FIRST [P51]

P53 OBSERVED

P54 ISA [P53, VARIABLE]

P55 SERIAL-POSITION

P56 ISA [P55, VARIABLE]

P57 DO [YOU, PLOT, P58]

P58 AS-A-FUNCTION-OF [P53, P55]

P59 ISA [P57, P51]

The above text base is elaborated with the following three task-domain schemata from the data graph task domain:

*Plot Schema*

IF ( [AS-A-FUNCTION-OF [*ARG-1, ARG-2* ]) è

[ROLE [*ARG-1*, DEPENDENT-VARIABLE],

[ROLE [*ARG-2*, INDEPENDENT-VARIABLE]

*Put Dependent Variable Schema*

IF ([ROLE [*ARG*, DEPENDENT-VARIABLE]) è

ON [*ARG*, Y-AXIS]

*Put Independent Variable Schema*

IF (ROLE [*ARG*, INDEPENDENT-VARIABLE]) è

ON [*ARG*, X-AXIS]

The meaning of "as a function of" in the original text is elaborated by the plot schema and the two put schemata. Execution of these three task-domain schemata during the construction phase for Sentence 5 adds the following propositions to the network:

P60 ROLE [OBSERVED, DEPENDENT-VARIABLE]

P61 ROLE [SERIAL-POSITION, INDEPENDENT-VARIABLE]

P62 ON [OBSERVED, Y-AXIS]

P63 ON [SERIAL-POSITION, X-AXIS]

At this stage, the proposition P57 is elaborated by the following:

P57-0 AND [P57-1, P57-2]

P57-1 DO [YOU, PLOT, P62]

P57-2 DO [YOU, PLOT, P63]

*3.3.1.2 Example From Editing Task Domain*

The instructions for Franzke's (1994) second task, editing the default graph, were terse descriptions of each edit (e.g., "Change the legend text to Geneva, 9, bold"). This cryptic instruction must be elaborated before the edit task can be understood and executed. For example, "Geneva" must be identified as the new font of the edited legend text. We assume experienced users of word-processing systems have specialized task-domain schemata termed *text attributes schemata* that transform such terse editing commands into comprehensible instructions. The text base for the original editing instruction follows:

P90 DO [YOU, PERFORM, P91]

P91 AND [P92, P94, P96]

P92 DO [YOU, CHANGE-TO, LEGEND, P93]

P93 PROPERTY [TEXT, $, GENEVA]

P94 DO [YOU, CHANGE-TO, LEGEND, P95]

P95 PROPERTY [TEXT, $, 9]

P96 DO [YOU, CHANGE-TO, LEGEND, P97]

P97 PROPERTY [TEXT, $, BOLD]

The original text does not identify explicitly which text attributes have the values Geneva, 9, and bold. These inferences are generated by the following text attributes schema:

*Text Attributes Schema*:

IF (PROPERTY [TEXT, $, *ARG* ] & ISA [*ARG*, FONT-NAME]) è

PROPERTY [TEXT, FONT, *ARG*]

IF (PROPERTY [TEXT, $, *ARG* ] & ISA [*ARG*, NUMBER]) è

PROPERTY [TEXT, SIZE, *ARG*]

IF (PROPERTY [TEXT, $, *ARG* ] & ISA [*ARG*, STYLE-NAME]) è

PROPERTY [TEXT, STYLE, *ARG*]

As the results of application of the schema, the original text base is elaborated by the following propositions:

P93-1 PROPERTY [TEXT, FONT, GENEVA]

P95-1 PROPERTY [TEXT, SIZE, 9]

P97-1 PROPERTY [TEXT, STYLE, BOLD]

### 3.3.2 Task-Goal Formation Schemata

Two task-goal formation schemata generate task goals used by the action-planning processes: *Task* and *Do-It*. These schemata elaborate propositions of the form DO [YOU, *VERB*, *OBJECT*] in the text base into propositions that represent task goals. Task schema elaborates propositions in the task domain, and Do-It schema elaborates propositions to perform a specific action on a screen object.

#### *3.3.2.1 Task Schema*

Standard parsing strategies and the global instruction reading schema generated the following text base for Sentence 1 (“In this experiment you are going to learn a new Macintosh application, Cricket Graph, by exploration”). The original text base is:

P10 IN-EXPERIMENT [P17]

P11 YOU

P12 NEW [P13]

P13 MACINTOSH [P14]

P14 APPLICATION

P15 CRICKET-GRAPH

P16 REF [P12, CRICKET-GRAPH]

P17 DO [YOU, LEARN, CRICKET-GRAPH]

P18 BY-EXPLORATION [P17]

By applying task schema to P17, the following propositions are added to the current network:

TASK-10 PERFORM [S10, S11]

S10 TASK-ACTION [LEARN]

S11 TASK-OBJECT [CRICKET-GRAPH]

The propositions TASK-10, S10, and S11 jointly define a task goal.

The task schema requires the *VERB* argument in the proposition DO [YOU, *VERB*, *OBJECT*] to be a description of TASK-ACTION, like PLOT, CHANGE, and CREATE. The task schema has the following form:

IF (DO [YOU, *VERB*, *OBJECT* ] & ISA [*VERB*, TASK-ACTION]) è

PERFORM [TASK-ACTION, TASK-OBJECT, TASK-SPECIFICATION],

TASK-ACTION [*VERB* ],

TASK-OBJECT [*OBJECT* ]

TASK-SPECIFICATION [*list of specifications* ]

The arguments in TASK-SPECIFICATION refer to propositions that modify *OBJECT*. The propositions in the consequence part, PERFORM [.], TASK-ACTION[.], TASK-OBJECT[.], and TASK-SPECIFICATION[.], jointly define a task goal for the action-planning processes.

*3.3.2.2 Do-It Schema*

In Franzke's (1994, 1995) experiment, participants were given instructions like "Please click on 'General Instructions' to start the experiment." Following such instructions involves significant inferences. For example, 'click on' must be mapped onto the action, 'single-click with the mouse button after moving the mouse cursor to the required screen object.' 'General Instructions' must be mapped onto the screen object with the label 'General Instructions.'

When *VERB* in propositions of the form DO [YOU, *VERB*, *OBJECT*] is a DEVICE-ACTION like click, drag, or release, the do-it schema elaborates such propositions into propositions representing task goals. The do-it schema has the following form, and the arguments in DEVICE-SPECIFICATION refer to propositions that modify the *OBJECT*.

*Do-It Schema I*:

IF (DO [YOU, *VERB*, *OBJECT* ] & ISA [*VERB*, DEVICE-ACTION]) è

PERFORM [DEVICE-ACTION, DEVICE-OBJECT, DEVICE-SPECIFICATION]

DEVICE-ACTION [*VERB* ]

DEVICE-OBJECT [*OBJECT* ]

DEVICE-SPECIFICATION [*list of specifications* ]

The text “Please click on ‘General Instructions’ to start the experiment” is propositionalized as follows:

P20 EXPERIMENT

P21 $

P22 HAS-LABEL [P21, General Instructions]

P23 DO [YOU, CLICK, P21]

P24 BY [P25, P23]

P25 DO [YOU, START, EXPERIMENT]

In P23, CLICK is a kind of DEVICE-ACTION; thus, the do-it schema generates the following propositions representing a task goal:

PERFORM [S21, S22, S23],

S21 DEVICE-ACTION [CLICK],

S22 DEVICE-OBJECT [$] ; undefined

S23 DEVICE-SPECIFICATION [DEVICE-LABEL [General Instructions]]

There are cases where “click” may be replaced with a representation of general actions (e.g., “act on”) that do not directly indicate device actions, whereas a screen object is indicated. Another version of the do-it schema exists for these cases. In the example instruction, “Example Data” in Sentence 3 can trigger the rule:

*DO-IT-Schema II*:

IF (HAS-LABEL [*OBJECT*, *LABEL*]) è

PERFORM [$, DEVICE-OBJECT, DEVICE-SPECIFICATION]

DEVICE-OBJECT [*OBJECT*]

DEVICE-SPECIFICATION [DEVICE-LABEL [*LABEL*],*list of specifications*]

## 3.4 Reading Instructions

The simulation of reading the instruction text (provided in Figure 3), generating task goals and then storing these goals, and carrying-over a few nodes, entails processing the text sentence-by-sentence in a series of problem-model construction cycles. For each sentence, the comprehension schemata defined in the previous section are applied to generate an elaborated text base of the original sentence. The program developed by Mross and Roberts (1992) simulates the construction–integration cycles for the elaborated text base of each sentence. The simulation of reading the first sentence in Figure 3 is described in subsections, 3.4.1-3. The result of reading the whole instructions is presented in 3.4.4.

### 3.4.1 Construction and Integration of the Network

As described in Section 3.3.2.1, the text base representing Sentence 1 consists of nine propositions, P10 - P18, and three additional propositions generated by the task schema. Figure 4 illustrates these results. The oval nodes represent the original text, and the output of the task schema are depicted as rectangles. These propositions are connected by their shared arguments. For example, P14: APPLICATION and P13: MACINTOSH[P14], which is shorthand for MACINTOSH[APPLICATION], are linked by the shared argument APPLICATION. The simulation program reads the list of propositions as input and then constructs the linked network, represented as a matrix, **C**. The initial values for the link strengths are set to 1.0.

Insert Figure 4 about here

The network is then integrated by the following procedure. $\mathbf{A}^{(0)}$ is a row vector that represents the activation values and has the same size as matrix **C**. Initially, the activation values for the source nodes, P10–P18, are set to 1, and the activation values for the elaborations are set to 0; $\mathbf{A}^{(0)}$ = (1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0). The activation vector for the first iteration of the integration cycle $\mathbf{A}^{(1)}$ is computed by first postmultiplying $\mathbf{A}^{(0)}$ by **C** (i.e., $\mathbf{A}^{(0)} \times \mathbf{C}$) and then normalizing the resulting vector. The normalization process sets negative activation values to 0 and then scales the positive values so that the activation value of the most highly activated node equals 1.0. The iteration process continues until the network converges, namely, until the average change in the activation vectors for two successive iterations becomes less than an arbitrarily determined threshold value, say, 0.001. The activation vector for the last iteration represents the outcome of the comprehension process. The numbers in bold face next to the nodes in Figure 4 are activation values for the network that reached convergence after 11 iterations.

### 3.4.2 Forming Episodic Memory

When the network is integrated, the results are transferred to episodic memory. The process, based on Kintsch and Welsch's (1991) model, is carried out automatically by Mross and Roberts' (1992) program. The episodic memory is represented as a collection of propositions—the unique propositions generated during the problem-model construction cycles. In our example, when the first cycle is completed, the episodic memory is empty; thus, copies of nodes in the working memory shown in Figure 4 comprise episodic memory. Each node in episodic

memory has a self strength, reflecting the activation value of the corresponding node in the integrated network. Likewise, each link in episodic memory has a link strength, reflecting the activation values of the corresponding nodes in the integrated network. Figure 4 shows the values of self strengths and link strengths in parentheses as calculated by Mross and Roberts' program.

### 3.4.3 Carrying Over Propositions to Maintain Coherence

The purpose of the sequence of problem-model construction cycles is to generate a representation of the text, not just individual sentences. Linking the representations of individual sentences establishes textual coherence. In addition, the model takes advantage of its understanding of the instruction that has been achieved so far. The process of establishing of textual coherence and accumulating knowledge about the task is modeled by carrying over propositions from one cycle to another. The following carry-over process was used by Kintsch (1988).

After storing the result of comprehension of Sentence *n* and before processing Sentence *n+1*, the model carries over a few nodes in working memory to maintain coherence and accumulate knowledge about the task. Textual coherence is maintained by carrying over the three most highly activated nodes to the next processing cycle. Accumulation of knowledge about the task is accomplished by carrying over a set of propositions that originated from a task-goal formation schema. For example, after processing Sentence 1 the program carries over P17, P15, and P16 and TASK-10, S10, and S11 to the cycle for Sentence 2.

### 3.4.4 Results of Problem-Model Construction Cycles

The six sentences shown in Figure 3 were processed by a sequence of problem-model construction cycles, and Figure 5 shows task-goal propositions

generated by these cycles.

Insert Figure 5 about here

## 3.5 Exploration

After processing the instructions, LICAI enters the goal-selection and action-planning processes that generate exploratory behavior. These processes involve a memory-retrieval cycle followed by one or more attention-action-selection cycles. The memory-retrieval cycle uses display as a retrieval cue to retrieve a task goal. The retrieved task goal is passed to Kitajima and Polson's (1995) action-planning model, operating without device goals, to generate one or more actions. The resulting display is used as a new cue and the process continues. LICAI simulated the first three steps of Franzke's (1994) first task, and Figure 6 shows the initial sequence of displays generated during exploration.

Insert Figure 6 about here

### 3.5.1 Goal-Selection Processes

Each display contains a collection of screen objects, and each screen object is represented as a few propositions (see Kitajima & Polson, 1995, for details). In the simulation of the memory-retrieval cycle, propositions representing screen objects that link task goals in episodic memory were incorporated in the network with all of the task goals. The memory-retrieval cycle was simulated using Mross and Roberts' (1992) program.

Figure 7 is a schematic representation of the network for the memory-retrieval cycles. The display cues are represented as a set of propositions that convey perceptual information, such as the label of a screen object and its state (e.g.,

highlighted). For example, Display I (shown in Figure 6) is represented as seven propositions:

Insert Figure 7 about here

D-10 OBJECT-10

D-101 HAS-LABEL [OBJECT-10, CRICKET-GRAPH]

D-102 ISA [OBJECT-10, APPLICATION]

D-20 OBJECT-20

D-201 HAS-LABEL [OBJECT-20, EXAMPLE-DATA]

D-202 ISA [OBJECT-20, DOCUMENT]

D-203 CONTAIN [OBJECT-20, DATA]

Links are formed between display propositions and propositions in episodic memory that share arguments. These links, defined by overlapping arguments, are given a strength of 1.0. In the network integration process, the node representing the display object itself, $D_n$, serves as a permanent activation source, whose activation value is always reset to 1.0 before an iteration of the spreading activation process. Other nodes in the display representation are initially set to 1.0, but their activation values can change on each iteration. The nodes in episodic memory are set to their self strengths as their initial activation values. At the end of the integration phase of the memory-retrieval cycle, the most highly activated task goal is passed to the action-planning processes.

Figure 8 shows the results of the goal-selection processes by the activation values for each task goal proposition on each of the three display conditions. The actual activation values of each task goal proposition can be calculated by multiplying each entry by 0.0001.

Insert Figure 8 about here

### 3.5.2 Action-Planning Processes

Simulation of the action-planning processes was performed by examining the selected task goal in the light of the result by Kitajima (1996) to predict the anticipated result of the action-planning processes.

The correct action for the initial step of the first task is to move the mouse cursor to the document icon labeled "Example Data" and double click on it to open the document. Display I (shown in Figure 6) was the retrieval cue. As shown in Figure 8, the most highly activated task goal was PERFORM [$, $, SPEC [LABEL [EXAMPLE-DATA], DOCUMENT]], DO-IT-31. This proposition represents the task goal of performing a yet-to-be-specified action on an unspecified screen object—a document labeled "Example Data." The argument EXAMPLE-DATA overlaps with the label on the correct document icon in Display I. In addition, the argument DOCUMENT reinforces the link between the task goal and the correct screen object. This link is through the proposition D-202: ISA [OBJECT-20, DOCUMENT]. The argument DOCUMENT had significant effect on the resulting activation pattern. Unless DOCUMENT were specified in DO-IT-31, a competing task goal, TASK-10, would have been equally activated as DO-IT-31. These links enable the attention cycle to select the correct screen object.

The correct action in the second step is to pull down the **Graph** menu. LICAI retrieved TASK-40, PERFORM [CREATE GRAPH] using Display II as a retrieval cue. The link between the screen object label representing the **Graph** menu and the GRAPH in TASK-40 caused this task goal to become the most highly activated. This

same link causes the action-planning processes to attend to the **Graph** menu item, move the cursor to it, and pull it down.

The links from Display III to TASK-61, PERFORM [PLOT, OBSERVED, Y-AXIS], and TASK-62, PERFORM [PLOT, SERIAL-POSITION, X-AXIS], caused both task goals to become equally highly activated. There are multiple links from screen objects that make up the variable selection dialog boxes to these propositions. However, there is no mechanism in LICAI to resolve a tie. Kitajima (1996) showed that each goal would cause the action-planning processes to select the correct variable name in each dialog box. Thus, in this case each task goal could generate correct actions regardless of the order in which the highest activated task goals were passed to the action-planning processes.

### 3.5.3 Summary

These simulation results clearly demonstrate LICAI's dependence on various manifestations of the label following strategy in the exploration processes. Links between task goals and object labels on the display play identical key roles in mediating correct performance. Labels for the correct screen objects form links that control successful retrieval of the correct task goal and that guide successful action planning. In addition, Kitajima and Polson (1997) have shown that these same links are involved in successful retrieval of action sequences.

## 4. EVALUATION OF LICAI

We evaluated LICAI using Franzke's (1994, 1995) results. The model only makes coarse predictions about the behavior of Franzke's participants. The instructions and comprehension schemata assumed by LICAI might enable the model to generate correct task goals, but the model does not describe the search

behavior that occurs if the task-goal construction process fails (Rieman et al., 1996). In fact, Rieman and colleagues found that people will do some exploration of the interface even when they can construct the correct task goal and execute the correct action within 15 to 30 seconds.

## 4.1 Task Descriptions, Object Labels, and Number of Screen Objects

Franzke (1995) presented an analysis in which she collapsed her data across interfaces and tasks. She focused on the relationships among task descriptions for each step, the correct object labels, and the number of screen objects that were possible targets. She analyzed the time it took participants to complete a step. If participants failed to discover the correct action within 2 minutes, she gave a hint.

Franzke partitioned the possible relationships between a task description and the label of the correct screen object into four categories: (a) a perfect match between the label and the task descriptions; (b) the labels were synonyms of the task description; (c) an inference was required to link the two; and (d) no label on the object. All direct manipulation actions (like double-click to gain access to an editing dialog box) were in the fourth category. Franzke also used a coarser classification (good, poor) where the first two categories are the good labels. We did not simulate all of the tasks and interfaces used in Franzke's experiments because our goal was to demonstrate that LICAI can provide a qualitative account of Franzke's major results. Recall that LICAI will fail to discover the correct action if there is not a link from the task description to the correct screen object.

### 4.1.1 Label Following

Franzke (1995, Figure 5) found strong support for the label following strategy

(Polson & Lewis, 1990; Polson et al., 1992). The time required to perform the correct action was well under 30 seconds for the match (a) and synonym (b) categories, and the time required for the other two categories (c and d) was significantly longer. In fact, a large fraction of the no label (d) relationship steps required hints. During exploration, participants used overlap between task descriptions contained in the instructions and labels on menus, buttons, and other interface objects.

The degree of success on first attempt at a task was also dependent on the number of screen objects. Franzke (1995, Figure 6) showed that the more objects displayed, the longer participants took to complete the step. Franzke (1995, Figure 7) also analyzed action times for discovering correct actions and found an interaction between number of objects (2–10) on the screen and quality of the label match (good or poor). There was no effect of number of objects for good labels and a large effect for poor labels. A unique, good label caused a person to attend to the correct screen object regardless of the number of screen objects.

These results are consistent with LICAI. If an instruction contained a description of either an action or an object that matched a screen object label, those labels are preserved when the propositional representation of the instruction is mapped into a task goal. The links between the task goal and the correct screen object can mediate performance of the correct action if the matching label is unique, even when there are a large number of screen objects. The correct screen object will be included in the set of candidate objects in the attention cycle because the unique links established by the matching label will cause selective activation of the correct node.

We did not simulate the success of synonyms in mediating successful label following. However, the result are consistent with the construction–integration architecture. Common synonyms would be retrieved during the elaboration process and added to the network. These synonyms would then enable the model to construct links between task goals and plan elements required in the action-planning processes.

### 4.1.2 Direct Manipulation

Franzke (1995, Figure 8) found that participants had trouble discovering direct manipulation actions when there were no label links between the correct screen object and the task description. Examples included double-clicking on a title or an axis label to edit it, clicking on an object in the tool bar, and dragging and dropping items. In a majority of first encounters with an example of such interactions, Franzke's (1994) participants had to be provided hints after 2 minutes of futile exploration. Franzke (1995) argued that participants did not have the knowledge necessary to infer that an object could be edited by double-clicking it. Thus, participants could not perform such tasks even if they had generated the correct task goals from the instructions. LICAI predicts such failures because links between the task goal inferred from the instructions and the correct screen object are missing.

## 4.2 Franzke's (1994) Instruction Texts

Franzke (1994) used two types of instructions in her experiments. The initial instructions enabled participants to generate the default graph and were more detailed than the instructional text used in our simulation. Participants could easily generate multiple task goals from such texts. The texts that described the edits to be performed on the default graph, on the other hand, were short and telegraphic.

People's comprehension problem here was to make the inferences necessary to generate an understandable task description and construct task goals that overlapped with screen object labels.

### 4.2.1 Task-Domain Schemata

The critical step in performing the first task, creating of the default graph, is interacting with the dialog box, which contains two scrolling lists for the x- and y-axis variable labels. LICAI transformed the original problem statement "Plot 'Observed' as a function of 'Serial Position'" into "Plot 'Observed' on the y-axis" and "Plot 'Serial Position' on the x-axis" by using task domain schemata (described in Section 3.3.1.1).

When Terwilliger and Polson (1997) measured the time experienced Macintosh users, who had never used a graphing application, took to interact with two forms of the variable selection dialog box, they found clear evidence that people also make such transformations. Terwilliger and Polson constructed two versions of task instructions for Franzke's first subtask. The XY instructions read "create a graph with 'Serial Position' on the x-axis and 'Observed' on the y-axis," and the FN instructions read "create a graph of 'Observed' as a function of 'Serial Position.'" Terwilliger and Polson also created two versions of the dialog box. In the XY version, the left selection list was labeled "X Axis" and the right selection list was labeled "Y Axis." In the FN version, the left list was labeled "Plot" and the right list was labeled "As a function of." Terwilliger and Polson found that tasks took less time with the XY dialog box for XY and FN instructions. In addition, the think-aloud protocols recorded while participants performed the task revealed that they verbalized the transformation of FN to XY. These results support the concept

of task-domain schemata, as well as the assumption that the label on the interface must match user descriptions of a task.

To perform the second task, “change the legend text to Geneva, 9, bold,” participants had to first double-click on the legend to open a dialog box. Most participants had trouble discovering this initial action and had to be given hints. Once the dialog box was open, however, participants had no trouble completing the task. The dialog box contained three scrolling lists labeled Font, Size, and Style. LICAI also generated the three task goals necessary to interact with the three scrolling lists (see Section 3.3.1.2) and performed each task specified by the original text. The text attributes schema had generated the task goals that link to a scrolling list title and to the relevant item in the scrolling list in the dialog box.

### 4.2.2 Long Instructions and Multiple Task Goals

Franzke’s (1994) instructions for the first task, creating the default graph, were more detailed than the instructions in our simulation and included general information on the experiment as well as details about which variables to place on the x- and y-axis. Even so, participants in the Cricket Graph I and III conditions still had some difficulty with these initial steps and had to reread the instructions and obtain critical information like variable names. Their difficulty is somewhat surprising because the instructions used terms that provided a perfect match for the label following strategy. As LICAI predicts, however, these instructions would generate multiple task goals and retrieval of these goals is a brittle process, heavily dependent on the specific display representation.

## 4.3 Summary

We have shown that LICAI accounts for the major results of the Franzke's (1994, 1995) extensive study of learning by exploration. A good match between one or more elements of a user's task and a label on a menu, button, or other screen object can guide successful exploration. LICAI's goal-selection and exploration processes are completely dependent on such overlaps. Thus, our model provides a good account of her results.

The goal-formation processes are more complex. Various comprehension schemata can transform the original problem statement in complex ways. Thus, the problem statement "plot 'Observed' as a function of 'Serial Position'" is transformed into "put 'Observed' on the y-axis" and "put 'Serial Position' on the x-axis." Kitajima (1996) showed that this transformation is necessary for LICAI to be able to perform the subtask of generating the default graph by exploration. Terwilliger and Polson (1997) found direct evidence that users will transform a problem statement of the form "plot ... as a function of..." in the manner consistent with LICAI's task-domain schemata.

However, LICAI will map the problem statement in the text almost directly into a task goal representation in the absence of any domain specific comprehension schemata that could transform or elaborate the original text, generating multiple task goals. Selecting a correct task goal could become a difficult task.

# 5. MODELS OF EXPLORATION

This section compares LICAI — a comprehension-based model of exploration — with another class of models of exploration, IDXL by Rieman, et al. (1996), a search-based model of exploration build on the Soar architecture. The key

difference is that comprehension is a highly automated collection of cognitive processes that utilizes large amount of knowledge whereas search-based models employ mechanisms that enable them to deliberately search the interface and memory for relevant cues and linking knowledge. Section 5.1 characterizes the comprehension-based models. Section 5.2 describes three differences between search-based models and LICAI.

## 5.1 Comprehension-Based Models

Kintsch's (1988) and Kintsch and Greeno's (1985) models of arithmetic word problems are instances of a general class of models that have been proposed repeatedly in the problem solving and skill acquisition literature (Greeno & Simon, 1988). These models have a comprehension-based problem representation building component and a problem solver that ultimately generates the solution. An early example of this class was Hayes and Simon's (1974) UNDERSTAND model that processed instructions for tasks like the Tower of Hanoi and generated a representation that was input to the General Problem Solver (Ernst & Newell, 1969).

LICAI and Kintsch's models of arithmetic word problems are extreme versions of this class of models in that they have no problem-solving mechanisms. Many of the cognitive operations that other investigators would characterize as problem-solving activities are instead done in the comprehension component. This difference is justified by Kintsch (1988), who argued that children's difficulty with arithmetic word problems is not in performing the basic arithmetic operations but in comprehending the problem description, and by Hudson (1983), who showed that linguistic factors can have a powerful effect on problem-solving success.

Because Kitajima and Polson's (1995) action-planning model requires explicit task goals to generate correct actions, the form of useful goals is strongly constrained by the surface features of the interface—like labels in dialog boxes and on menus. Assuming that skilled users have specialized comprehension knowledge directly analogous to Kintsch and Greeno's arithmetic schemata, the schemata should enable them to construct the required, specific goal representation from text descriptions of their task. On a more general level, the flexibility and power of an expert user's skills in using a specific application are in the problem-formation component and not in the action-planning component.

## 5.2 Search-Based Models and LICAI

The construction–integration theory was developed to account for reading, a highly skilled behavior. It was then extended to account for action planning in skilled computer users (Kitajima & Polson, 1995; Mannes & Kintsch, 1991). In contrast to the ACT-R (Anderson, 1993) or Soar (Newell, 1990) architectures, the construction–integration theory has a primitive control structure, but it does have components that enable it to exhibit search behavior, including goals, a mechanism to choose between alternative actions, and the ability to react to results of a selected action.

LICAI starts in comprehension mode by reading the instructions and storing alternative task goals in memory. It then switches to action-planning and attempts to execute the correct actions by using cues generated by successive displays to retrieve task goals from memory. LICAI, however, cannot interleave comprehension and action planning. The underlying construction–integration architecture does not support the structures necessary to pass control between

action-planning and comprehension modes based on the current state of comprehension or action-planning processes.

Alternative theories of exploration do have such control structures, however. IDXL, a Soar model developed by Rieman et al. (1996), simulates learning by exploration of the Cricket Graph task and integrates much recent research on learning by exploration (Howes, 1994; Howes & Young, 1996; Rieman, 1994; Rieman et al., 1994).

There are three important differences between LICAI and IDXL. One is grain size. IDXL accounts for the actual search behavior by modeling the user's scanning of the display and examination of pull-down menus. Rieman (1994) conducted a detailed analysis of the actual exploratory behavior of Franzke and Rieman's (1993) participants who learned Cricket Graph I and found that participants will explore the display even when a menu with a label matching the current goal is present. Participants exhibit a form of "iteratively deepening attention" (Rieman et al., 1996). The following menu scanning behavior is an illustration. On first encounter, a menu is pulled down, quickly scanned, and then the mouse cursor is moved to another screen object. During a later encounter, participants study menu items by highlighting and pausing on each one in succession.

IDXL has a primitive model of attention that focuses on one screen object at a time. The action-planning operations include scanning, pointing, dropping pull-down menus, releasing on a menu item, and the like. The comprehension processes operate on the current attended-to screen object and compare its label to a current task goal. The output of the comprehension operators include forming an exact match, recognizing a synonym, recalling, constructing an analogy, asking for

instruction, and envisioning consequences.

LICAI's action-planning processes model the same behavior but at a more abstract level. The model retrieves a single task goal, considers all screen objects concurrently, and generates an action sequence associated with the task goal. IDXL can account for situations where there is not a perfect match between the current task goal and the correct screen object label; the user discovers the correct action after a significant amount of search. LICAI simply fails to generate the correct action sequence.

A second difference is that IDXL interleaves comprehension and action planning. Both are forms of progressive deepening search mechanisms and both operators are ordered by cost. Scanning actions, such as moving the mouse cursor and pulling down a menu, have low cost, but releasing on an incorrect menu item can be costly. Comprehension operators are applied to screen objects put in the focus of attention by scanning operators. Initially, low cost comprehension operators are applied like a test for an exact match to the current task goal. Failure causes IDXL to continue scanning. When the model's attention returns to the same screen object, results of the previous comprehension operators are recalled and lead to application of more costly comprehension operators. IDXL will act when the representation of the proposed action, generated by successive application of more costly comprehension operators, has generated a good match to the current task goal.

The third difference is that IDXL does not comprehend the original task instructions. IDXL is supplied with a task description in working memory that is a multipart, hierarchical representation in a subject–verb–object format. IDXL

assumes a shifting internal focus of attention, although it has not been modeled yet, to define a current task goal (Rieman et al., 1996).

It is clear that developments of LICAI must extend the underlying architecture with control mechanisms that enable the interleaving of comprehension and search. A principled account of the search behavior described by Rieman (1994) and simulated by IDXL must be developed. However, the resulting model would be different from IDXL. IDXL's comprehension operators elaborate the user's representation of objects on the interface with a fixed representation of a current task goal. These operators are similar to the elaboration phase in Kitajima and Polson's (1995) action-planning model. An extension of LICAI, in contrast, would be to search for alternative interpretations of the instructions that enable the action-planning mechanisms to make progress on the task.

Franzke's (1994) instructions for the task simulated by IDXL were similar to but more detailed than the instruction input to LICAI. In both cases, the comprehension processes generated multiple task goals. Our interpretation of much of the search behavior observed by Franzke and Rieman (1993) is that Franzke's participants are searching for possible interpretations of the instructions to find useful task goals. LICAI simulates a participant who processes all instructions. However, Hayes and Simon (1974) observed that participants have a strong tendency to skim instructions, and they attempt to solve the problem without all of the necessary information. They are then forced to return to the instructions, often several times, to acquire the necessary knowledge. Thus, part of the search behavior described by Franzke and Rieman is caused by incomplete and/or multiple interpretations of the instructions.

# 6. DISCUSSION

We have developed and evaluated LICAI, a model of display-based, human–computer interaction that has goal-formation and action-planning processes, both based on Kintsch's construction–integration architecture. The goal-formation processes transform initial task descriptions into goals that drive the action-planning processes and are specialized comprehension strategies that employ comprehension schemata to construct the required goal.

## 6.1 Comprehension Schemata for Instruction Taking

The schemata, triggered by specific features of the text base, add specialized elaborations and inferences to the network during the construction phase of each cycle. Two schemata are defined in terms of their content. The task-domain schemata elaborate the original statement of the task description. Examples include plot, put dependent variable, put independent variable (see 3.3.1.1), and text attributes schema (see 3.3.1.2). These schemata describe the user's specialized knowledge of the task domain, and they are independent of the application interface. Task-goal formation schemata (task schema and do-it schema) generate goals used by the action-planning processes. LICAI claims that instruction reading is a strategic process that uses these schemata. The plot and put dependent variable, put independent variable schemata are supported by Terwilliger and Polson's (1997) results.

## 6.2 Label Following

Label following is simulated in LICAI by links between arguments in propositions representing goals and propositions representing the labels on screen

objects (e.g., menu items and button labels). As Kitajima's (1996) simulation showed, behavior of the action-planning processes is fragile when there is no device goal. New users of an application cannot generate device goals, and LICAI predicts that label following is the only strategy for successful exploration.

LICAI also predicts that successful exploration will occur only under circumstances where the instruction comprehension processes generate task goals that satisfy the label following constraint. However, the labels that define task goals are not necessarily present in the original instruction texts. The original instruction texts can be elaborated by task-domain schemata, such as the text attributes schema.

## 6.3 Multiple Task Goals

Another fundamental result dictated by the underlying architecture is that users studying instructions or an example will generate multiple task goals. Because the construction process is bottom up, there is no control process to force the model to incrementally generate a single task goal to be acted on after the instructions have been processed. Kintsch (1988) showed that selecting the correct problem model for arithmetic and algebra word problems was facilitated by both the situation context described in the problem text and the last question sentence of the problem description. For computer users interacting with a novel application, successive displays generated by the application serve as retrieval cues for task goals constructed during instruction processing. There are no control processes that generate the proper sequence of correct task goals.

## 6.4 Implications for Learning by Exploration

Our results have important implications for the development of training materials for interfaces that support learning by exploration (Wharton et al., 1994).

A user's background knowledge constrains the content of instructional materials, and the interfaces that support learning by exploration and the instructions that intend to facilitate this form of learning are dependent on one another.

### 6.4.1 The Minimalist Instruction Paradigm

Carroll (1990) summarized an influential research program on the design and evaluation of training materials for application programs. This work led to the development of the Minimalist Instruction paradigm, which was stimulated by the then surprising result that carefully designed, detailed training and reference materials for early 1980s word-processors were unusable (e.g., Mack, Lewis, & Carroll, 1983). Carroll's (1990) solution to the usability problems demonstrated by Mack et al. (1983) and other researchers was to refine the Minimalist Instruction paradigm. The minimalist approach focuses on users' tasks rather than on describing a system function-by-function, minimizes the amount of written materials, and tries to foster learning by exploration rather than provide step-by-step instructions. This approach also supports error recognition and recovery.

Carroll's (1990) emphasis on active users' desires to learn by exploration are supported by Rieman's (1994) results. However, LICAI shows that following written instructions is complex, analogous to solving arithmetic and algebra word problems. Successful instruction following requires application of the appropriate schemata to extract information from the text to guide generation of correct task goals. Thus, for a new user following even the most carefully prepared instructions is a difficult and error-prone process.

In addition, LICAI predicts that comprehending and following instructions becomes more difficult, especially for individuals with limited background

knowledge. Mack et al.'s (1983) participants were new users and therefore did not have the necessary schemata or action-planning knowledge assumed by LICAI. Thus, instructions for new users must be either detailed or incorporate extensive pretraining (like the Guided Tour[5] for the Macintosh interface) to explain how to use the mouse and select items from menus, and so forth. All this knowledge is incorporated in the action-planning component of LICAI.

LICAI can be used to develop explicit design guidelines for the content of minimalist instruction materials because the paradigm and the model both focus on the users' tasks. Task goals, representing what the user wants to achieve, drive the action-planning processes. Explicit, brief statements of relevant goals that can be understood by the user are critical ingredients in supporting learning by exploration. Many irrelevant task goals generated during the comprehension of the example instructions were caused by attempts to provide some motivation for the user's task. Carroll explicitly recommends deleting such irrelevant material.

LICAI also clarifies the constraints that must be understood to follow Carroll's (1990) design heuristic of minimizing the amount of written material contained in task instructions. Brief instructions solve the problem of dealing with multiple task goals, but comprehension of terse instructions, like those used by Franzke (1994), requires specialized background knowledge of the task and interface. For example, the task description, "change the legend text to Geneva, 9, bold," cannot be understood by someone who has no experience with a word-processor. Effectively

---

[5] Guided Tour is a tutorial program that comes with the Mac OS 7.0 and introduces users to the basic Macintosh interface conventions.

minimizing the amount of written material requires careful attention to the action and display knowledge and the schemata assumed in the target population. A minimalist version of an instruction manual for word-processors would need to assume that users have well-developed versions of task and do-it schemata. These schemata represent significant background knowledge (i.e., at least 6 month's experience).

Carroll's (1990) Minimalist Instruction paradigm focuses almost exclusively on training materials. However, there are important constraints that must be satisfied by the interface for learning by exploration to even be possible: The label following strategy must work. LICAI clarifies the intimate relationship between the content of minimalist instructions and the details of user interfaces that support learning by exploration. The key is the label following strategy. The model predicts that successful minimalist instructions must guide users to form task goals that contain terms that overlap with the labels of the correct screen objects. Such task goals enable the action-planning processes to select the correct action or action sequence without any previous instruction and without detailed step-by-step guidance. If the interface does not support label following, then the document designer must use step-by-step instructions, although Carroll (1990) and numerous other investigators have shown that people are reluctant, if not unwilling, to read and follow such instructions.

### 6.4.2 Cognitive Walkthroughs

Cognitive walkthrough evaluates interfaces for ease of learning by exploration. The walkthrough is organized like the structured walkthroughs widely used in the software development community (Yourdon, 1989) and is based on an earlier theory of exploratory learning (Polson et al., 1992). This method evaluates the

effectiveness of the label following strategy and characterizes the background knowledge necessary to infer correct actions (Wharton et al., 1994). For each action required to perform a task, a designer must show that users have a correct goal that guides selection of the next correct action. The label following strategy is the primary action-selection guide. The method also forces designers to specify the task and interface knowledge required by users to infer correct actions. For example, it is possible to select an object in a scrolling list by single clicking on it.

The theoretical analysis presented in this paper reinforces the importance of the label following strategy. In addition, it points to some limitations of the cognitive walkthrough method. A majority of the successful applications of the cognitive walkthrough have been on walk-up-and-use interfaces like automated teller machines or phone-based applications with voice menus. In all of these situations, there are a limited number of isolated tasks, and the interface guides the user through the procedure necessary to accomplish each task. The instruction for each step is brief and contains only the information necessary for the user to select the next action. Thus, the interface controls the processes involved in interleaving goal formation and action planning. More complicated tasks involving document preparation or the generation of data graphs give the user far more degrees of freedom and are more complex as a result.

In many instructional situations, users are given instructions of a paragraph or two similar in form and content to the example used in our simulations. LICAI generates multiple task goals when processing such instructions. Cues provided by the interface must control the retrieval processes to generate the proper sequence of task goals. These retrieval processes are not robust. A minor change in wording in

instructions could lead to the retrieval of incorrect task goals.

## 7. CONCLUSIONS

This paper has presented a model of exploration based on a cognitive architecture, the construction–integration framework, that has no detailed model of deliberate cognition; it cannot engage in serious search behavior. In spite of the model's limitations, it provides insight into the difficulties individuals have in comprehending detailed technical documentation. The model also provides insights on the strengths and limitations of the minimalist instruction paradigm as well as the cognitive walkthrough interface evaluation procedure.

Another option that we explored and rejected was to extend the construction–integration architecture with a Soar-like control structure to support sophisticated search. Living within the limitations of the construction–integration architecture, we were forced to focus on comprehension and the processes of goal formation and showed the background knowledge necessary to generate the highly constrained goals required by the model. LICAI is closely related to successful models of word arithmetic problem solving (Kintsch, 1988), and the major thrust of both models is that following instructions to carry out some procedure is a difficult task and that comprehending instructions requires specific, specialized background knowledge.

# 8. NOTES

## 8.1 Acknowledgments

Richard Young provided us with a detailed review of a much longer, early version of this paper, and he gave us excellent guidance in his role as action editor. We thank him for helping us clarify many issues and shorten the paper. John Rieman, Clayton Lewis, and Walter Kintsch also made important contributions. We also thank Suzanne Mannes and two anonymous reviews for their criticisms and suggestions.

## 8.2 Support

The authors gratefully acknowledge research support from the National Aeronautics and Space Administration under grant NCC 2-904. The opinions expressed in this paper are those of the authors and not necessarily those of NASA.

## 8.3 Authors' Addresses

Muneo Kitajima, National Institute of Bioscience and Human-Technology, 1-1 Higashi Tsukuba, Ibaraki 305, Japan. Tel: +81-298-54-6730. Email: kitajima@nibh.go.jp

Peter G. Polson, Institute of Cognitive Science, University of Colorado, CB 344, Boulder, CO 80309-0344, USA. Tel: +1-303-492-5622. Email: ppolson@psych.colorado.edu

## 8.4 HCI Editorial Record

This paragraph will be supplied by the Editor.

## 9. REFERENCES

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

Bovair, S., & Kieras, D. E. (1985). A guide to propositional analysis for research on technical prose. In B. K. Britton and J. B. Black (Eds.), *Understanding expository text.* Hillsdale, NJ: Erlbaum.

Carroll, J. M. (1990). *The Nuremberg funnel: Designing minimalist instruction for practical computer skills.* Cambridge, MA: MIT Press.

Compeau, D., Olfman, L., Sein, M., & Webster, J. (1995). End-user training and learning. *Communications of the ACM.* **38**, 24-26.

Cummins, D. D., Kintsch, W., Reusser, K., & Weimer, R. (1988). The role of understanding in solving word problems. *Cognitive Psychology*, **20**, 405–438.

Delarosa, D. (1986). A computer simulation of children's arithmetic word problem solving. *Behavior Research Methods, Instruments, and Computers*, **18**, 147–154.

Engelbeck, G. E. (1986). *Exceptions to generalizations: Implications for formal models of human-computer interaction.* Unpublished Masters Thesis, Department of Psychology, University of Colorado, Boulder.

Ernst, G. W., & Newell, A. (1969*). GPS: A case study in generality and problem solving*. New York: Academic Press.

Fletcher, C. R. (1985). Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, and Computers*, **17**, 565–571.

Franzke, M., & Rieman, J. (1993, September). Natural training wheels: learning and transfer between two versions of a computer application. *Proceedings of Vienna conference on human–computer interaction '93*. Vienna, Austria.

Franzke, M. (1994). *Exploration, acquisition, and retention of skill with display-based systems.* Unpublished Dissertation, Department of Psychology, University of Colorado, Boulder.

Franzke, M. (1995). Turning research into practice: Characteristics of display-based interaction. In *Proceedings of human factors in computing systems CHI '95* (pp. 421–428). New York: ACM.

Greeno, J. G., & Simon, H. A. (1988). Problem solving and reasoning. In R. C. Atkinson, R. Herrnstein, G. Lindzey, and R. D. Luce (Eds.), *Steven's handbook of experimental psychology* (pp. 589–639). New York: John Wiley.

Hayes, J. R., & Simon, H. A. (1974). Understanding problem instructions. In L. W. Gregg (Ed.), *Knowledge and cognition*. Hillsdale, NJ: Erlbaum.

Howes, A. (1994). A model of the acquisition of menu knowledge by exploration. In *Proceedings of human factors in computing systems CHI '94* (pp. 445–451). New York: ACM Press.

Howes, A., & Young, R. (1996). Learning consistent, interactive, and meaningful task-action mappings: A computational model. *Cognitive Science,* **20**, 301–356.

Hudson, T. (1983). Correspondences and numerical differences between disjoint sets. *Child Development,* **54**, 84–90.

Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1986). Direct manipulation interfaces. In D. A. Norman & S. W. Draper (Eds.), *User centered system design* (pp. 87-124). Hillsdale, NJ: Erlbaum.

Kintsch, W. (1974). *The representation of meaning in memory.* Hillsdale, NJ: Erlbaum.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction–integration model. *Psychological Review,* **95**, 163–182.

Kintsch, W. (in press).*Comprehension: A paradigm for cognition.* Cambridge University Press.

Kintsch, W., & Greeno, J. G. (1985). Understanding and solving word arithmetic problems. *Psychological Review,* **92**, 109–120.

Kintsch, W., & Welsch, D. M. (1991). The construction–integartion model: A framework for studying memory for text. In W. E. Hockley and S. Lewandowsky (Eds.), *Relating theory and data: Essays on human memory* (pp. 367–385). Hillsdale, NJ: Erlbaum.

Kintsch, W., & van Dijk, T. A. (1978). Toward a model of text comprehension and production. *Psychological Review,* **85**, 363–394.

Kitajima, M. (1996). *Model-based analysis of required knowledge for successful interaction with a novel display* (ICS technical report 96-03). Boulder, CO: Institute of Cognitive Science.

Kitajima, M., & Polson, P. G. (1995). A comprehension-based model of correct performance and errors in skilled, display-based human–computer interaction. *International Journal of Human–Computer Systems,* **43**, 65–99.

Kitajima, M., & Polson, P. G. (1996). A comprehension-based model of exploration. In *Proceedings of human factors in computing systems CHI '96* (pp. 324–331). New York: ACM.

Kitajima, M., & Polson, P. G. (1997). LICAI+: A comprehension-based model of learning for display-based human–computer interaction. In *Conference companion of human factors in computing systems CHI '97*. New York: ACM.

Larkin, J. H. (1989). Display-based problem solving. In D. Klahr & K. Kotovsky (Eds.), *Complex information processing: The impact of Herbert A. Simon* (pp. 319-342). Hillsdale, NJ: Erlbaum.

Mack, R. L., Lewis, C. H., & Carroll, J. M. (1983). Learning to use word-processors: Problems and prospects. *ACM Transactions on Office Information Systems*, **1**, 254–271.

Mannes, S. M., & Kintsch, W. (1991). Routine computing tasks: Planning as understanding. *Cognitive Science*, **15**, 305–342.

Mross, E. F., & Roberts, J. O. (1992). *The construction–integration model: A program and manual* (ICS technical report 92–14). Boulder, CO: Institute of Cognitive Science.

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

Norman, D. A. (1986). Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.), *User centered system design* (pp. 31-61). Hillsdale, NJ: Erlbaum.

Norman, D. A. (1986). Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.), *User centered system design* (pp. 31-61). Hillsdale, NJ: Erlbaum.

Payne, S. J., Squibb, H. R., & Howes, A. (1990). The nature of device models: The yoked state hypothesis and some experiments with text editors. *Human–Computer Interaction*, **5**, 415–444.

Polson, P. G., & Lewis, C. (1990). Theory-based design for easily learned interfaces. *Human–Computer Interaction*, **5**, 191–220.

Polson, P. G., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive Walkthroughs: A method for theory-based evaluation of user interfaces. *International Journal of Man–Machine Studies*, **36**, 741–773.

Raaijmakers, J. G., & Shiffrin, R. M. (1981). Search of associative memory. *Psychological Review*, **88**, 93–134.

Rieman, J. (1994). *Learning strategies and exploratory behavior of interactive computer users*. Ph.D. dissertation, Department of Computer Science, University of Colorado, Boulder.

Rieman, J. (1996). A field study of exploratory learning strategies. *ACM Transactions on Computer–Human Interaction*, **3**, 189–218.

Rieman, J., Lewis, C., Young, R. H., & Polson, P. G. (1994). Why is a raven like a writing desk? Lessons in interface consistency and analogical reasoning from two cognitive architectures. In *Proceedings of human factors in computing systems CHI '94* (pp. 438–444). New York: ACM.

Rieman, J., Young, R. M., & Howes, A. (1996). A dual space model of iteratively deepening exploratory learning. *International Journal of Human–Computer Studies*, **44**, 743–775.

Terwilliger, R. B., & Polson, P. G. (1997). Relationships between users' and interfaces' task representations. In *Proceedings of human factors in computing systems CHI '97.* New York: ACM.

Wharton, C., & Kintsch, W. (1991). An overview of the construction–integration model: A theory of comprehension as a foundation for a new cognitive architecture. *SIGART Bulletin,* **2**, 169–173.

Wharton, C., Rieman, J., Lewis, C., & Polson, P. G. (1994). The Cognitive Walkthrough method: A practitioner's guide. In J. Nielsen and R. Mack (Eds.), *Usability inspection methods*. New York: John Wiley.

van Dijk, T. A., & Kintsch, W. (1983). *Strategies of discourse comprehension*. New York: Academic Press.

Yourdon, E. (1989). Structured walkthroughs (4th ed.). Englewood Cliffs, NJ: Yourdon Press.

## 10. FIGURE CAPTIONS

*Figure 1.* Variable selection dialog box from Cricket Graph 1.

*Figure 2.* The action-planning model of correct performance and errors in skilled, display-based HCI (Kitajima & Polson, 1995), mapped onto Norman's (1986) action theory framework.

*Figure 3.* Instruction texts: A version from Franzke (1994).

*Figure 4.* A network representation for the first sentence, "In this experiment you are going to learn a new Macintosh application, Cricket Graph, by exploration."

*Figure 5.* Task goals generatedfrom the example instructions by the problem-model construction cycles.

*Figure 6.* Description of displays used for the simulation of goal-selection processes.

*Figure 7.* Retrieval of task goals, PERFORM [ACTION, OBJECT, SPEC], cued by external screen representations.

*Figure 8.* Activation values of potential task goals after a memory-retrieval cycle (unit 0.0001).

*Figure 1*. Variable selection dialog box from Cricket Graph I.

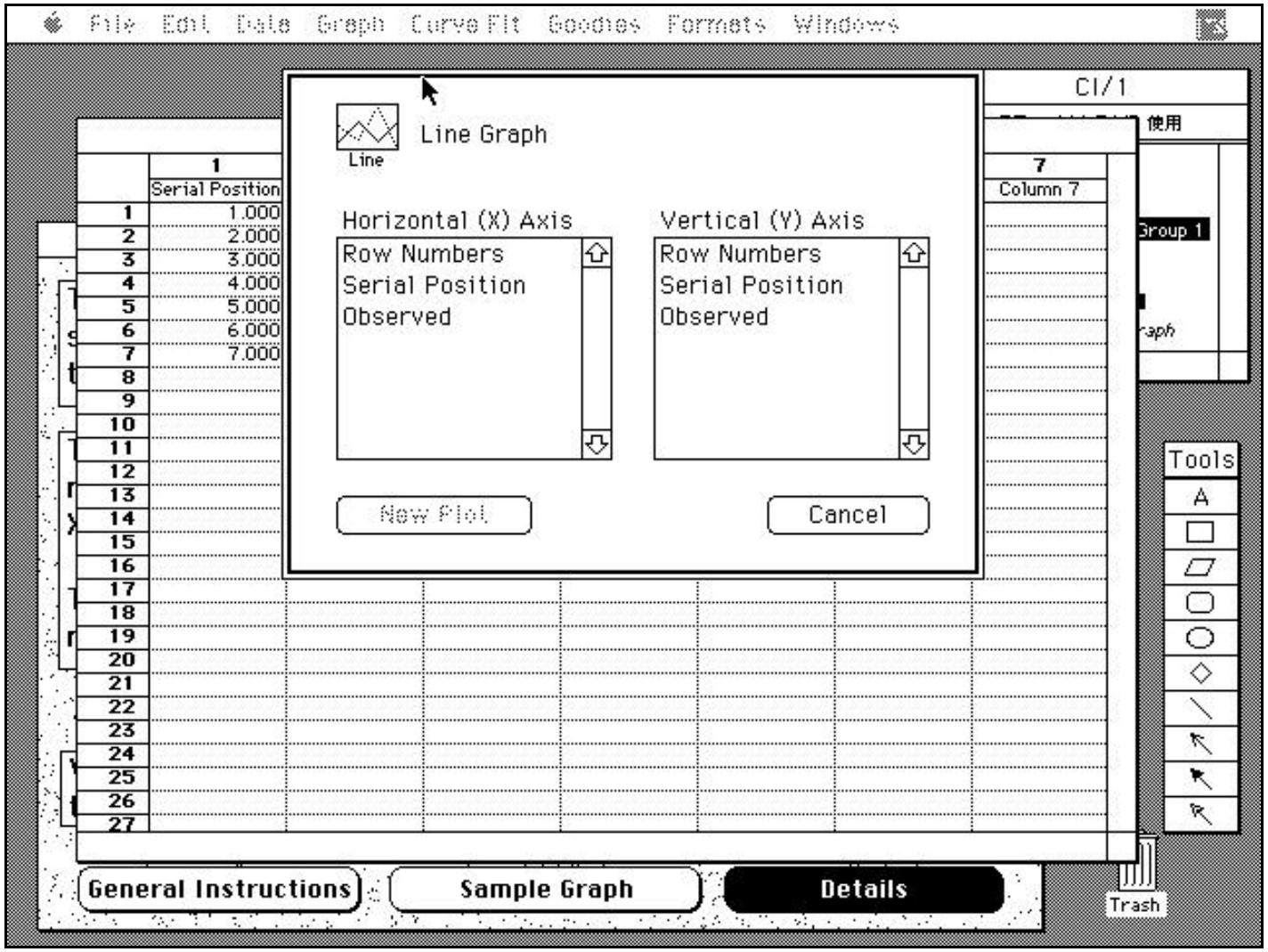


*Figure 2*. The action-planning model of correct performance and errors in skilled, display-based HCI (Kitajima & Polson, 1995), mapped onto Norman's (1986) action theory framework.

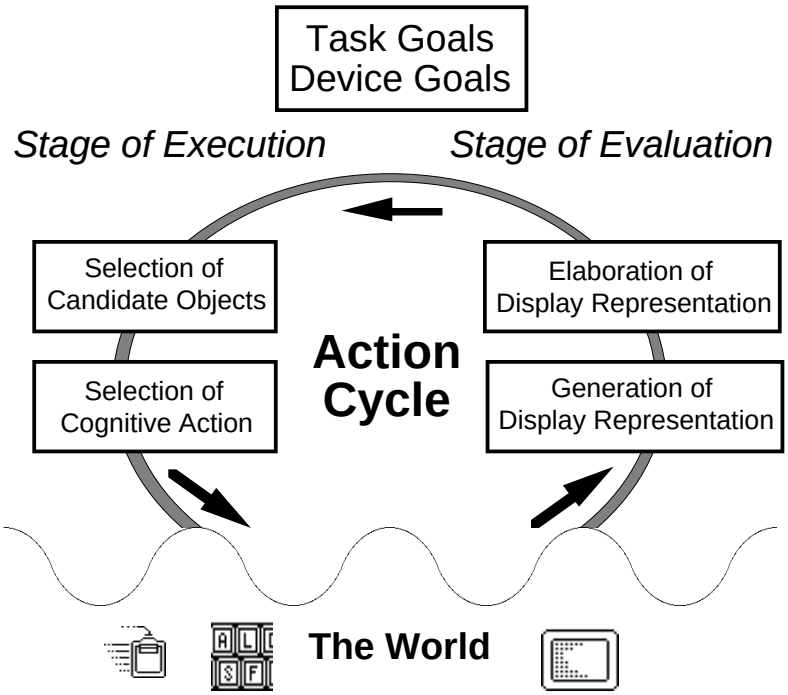


*Figure 3*. Instruction texts: A version from Franzke (1994).

| Instruction texts | Cycle number |
|---|---|
| In this experiment you are going to learn a new Macintosh application, Cricket Graph, by exploration. | 1 |
| The task you are going to perform will be presented to you as a series of exercises. | 2 |
| The data you are going to plot is contained in a Cricket Graph document, “Example Data.” | 3 |
| Your overall goal is to create a new graph that matches the example graph shown in the instructions. | 4 |
| Your first exercise is to plot the variable ‘Observed’ as a function of the variable ‘Serial Position.’ | 5 |
| After you have created a new graph, you will modify it so that it more closely matches the example given in your instructions. | 6 |

*Figure 4.* A network representation for the first sentence, "In this experiment you are going to learn a new Macintosh application, Cricket Graph, by exploration."

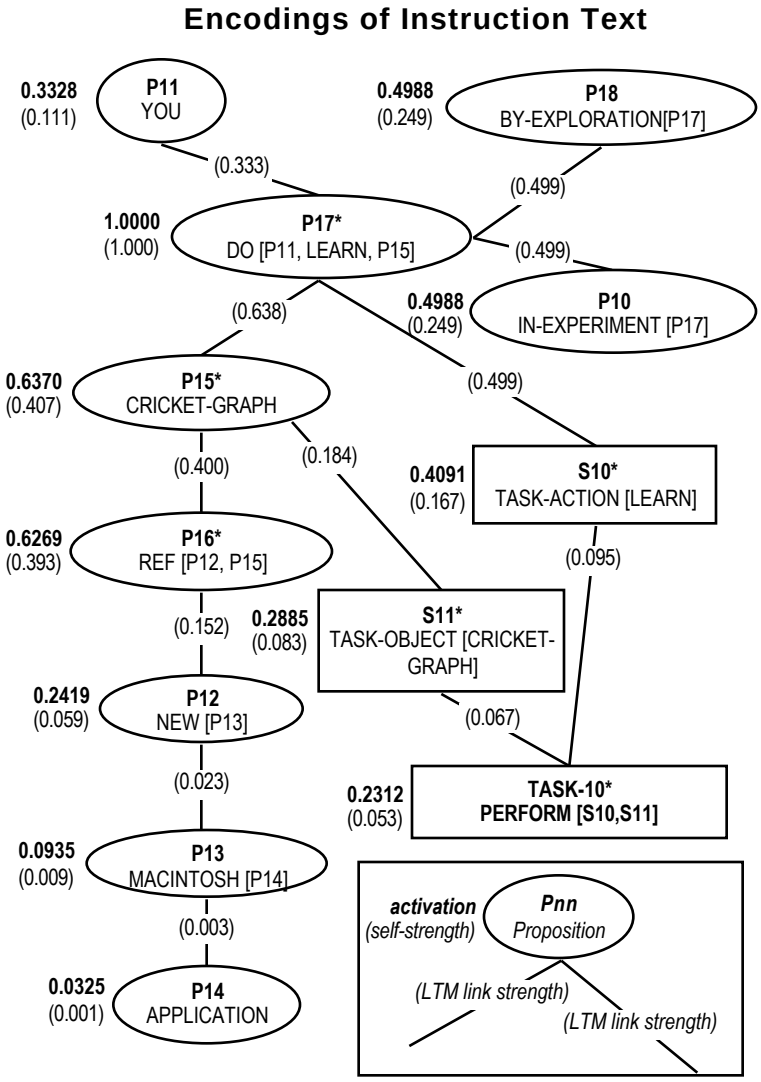


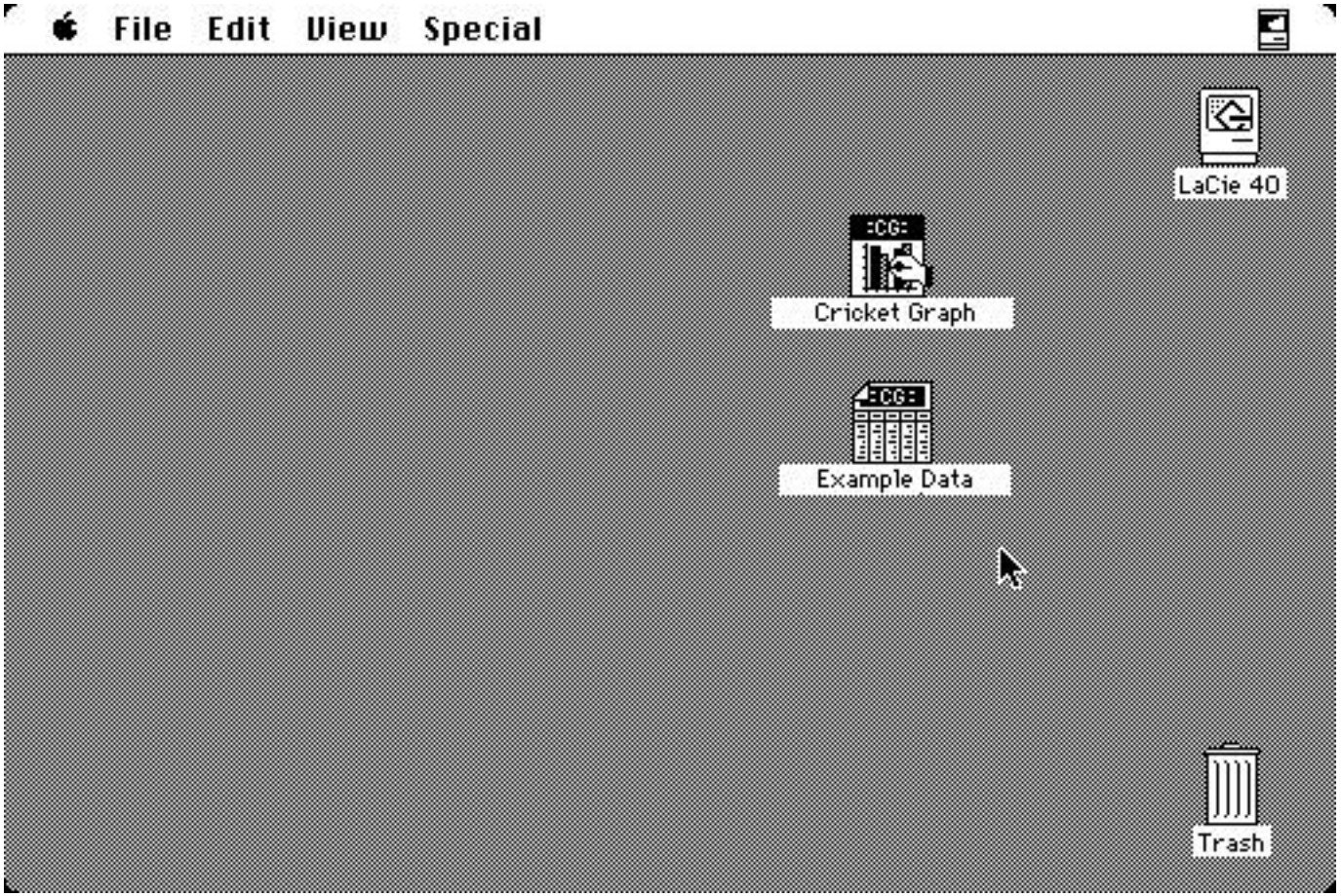
*Figure 5*. Task goals generated from the example instructions by the problem-model construction cycles.

| Sentence | Proposition label | Task goal (PERFORM [*ACTION OBJECT SPECIFICATION* ]) |
|---|---|---|
| 1 | TASK-10 | PERFORM [LEARN, CRICKET-GRAPH] |
| 2 | TASK-20 | PERFORM [PERFORM, TASK] |
| 3 | TASK-30 | PERFORM [PLOT, DATA] |
| 3 | DO-IT-31 | PERFORM [$ $ SPEC [LABEL [EXAMPLE-DATA], DOCUMENT]] |
| 4, 6 | TASK-40 | PERFORM [CREATE, GRAPH] |
| 5 | TASK-60 | PERFORM [PLOT, $, AS-A-FUNCTION-OF [OBSERVED, SERIAL-POSITION]] |
| 5 | TASK-61 | PERFORM [PLOT, OBSERVED, Y-AXIS] |
| 5 | TASK-62 | PERFORM [PLOT, SERIAL-POSITION, X-AXIS] |
| 6 | TASK-70 | PERFORM [MODIFY, GRAPH] |

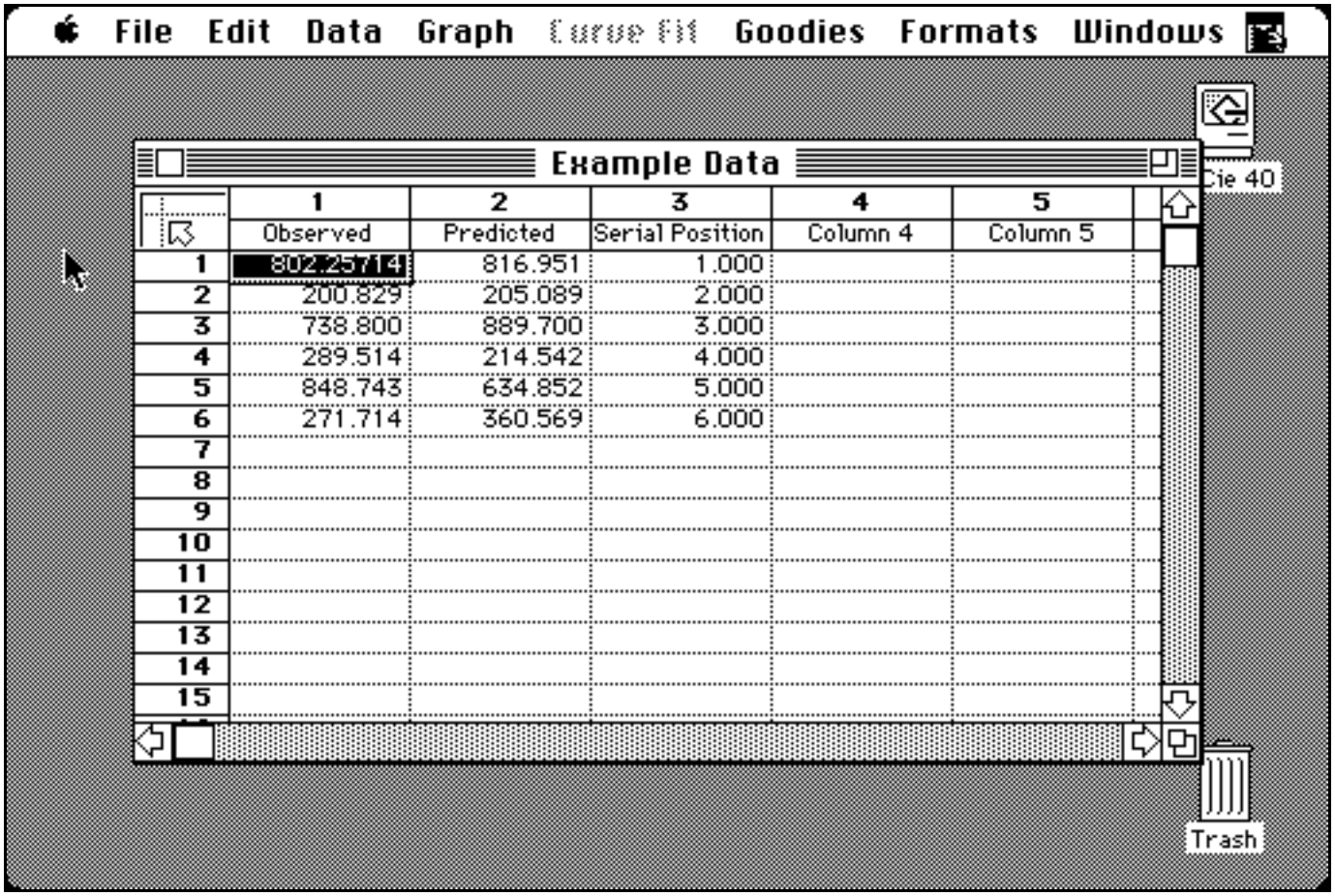
*Figure 6.* Description of displays used for the simulation of goal-selection processes.

| Description of display and part of display included in the simulation of goal-selection processes | Actual display |
|---|---|
| **Display I:** Beginning of task<br>Simulation of goal-selection process included<br>• Two desktop icons “Cricket Graph” and “Example Data” | **[ Insert photo-reduced copy of Screen-Shot 6-I for publication here]** |
| **Display II:** After launch of Cricket Graph I by opening “Example Data” document icon<br>Simulation of goal-selection process included<br>• Two items from the menu bar, “Data” and “Graph”<br>• Labels from spreadsheet, “Serial Position” and “Observed” | **[ Insert photo-reduced copy of Screen-Shot 6-II for publication here]** |
| **Display III:** The variable selection dialog box for assigning variables to x-axis and y-axis obtained by selecting “Line” from “Graph” menu<br>Simulation of goal-selection processes included<br>• Two pairs of “Serial Position” and “Observed” in the variables selection dialog box | **[ Insert photo-reduced copy of Screen-Shot 6-III for publication here]** |

*Note.* In the simulation of goal-selection processes, part of each display was used for retrieving a task goal from the episodic memory. Versions of displays in Franzke’s (1994) experiment are shown for reference.

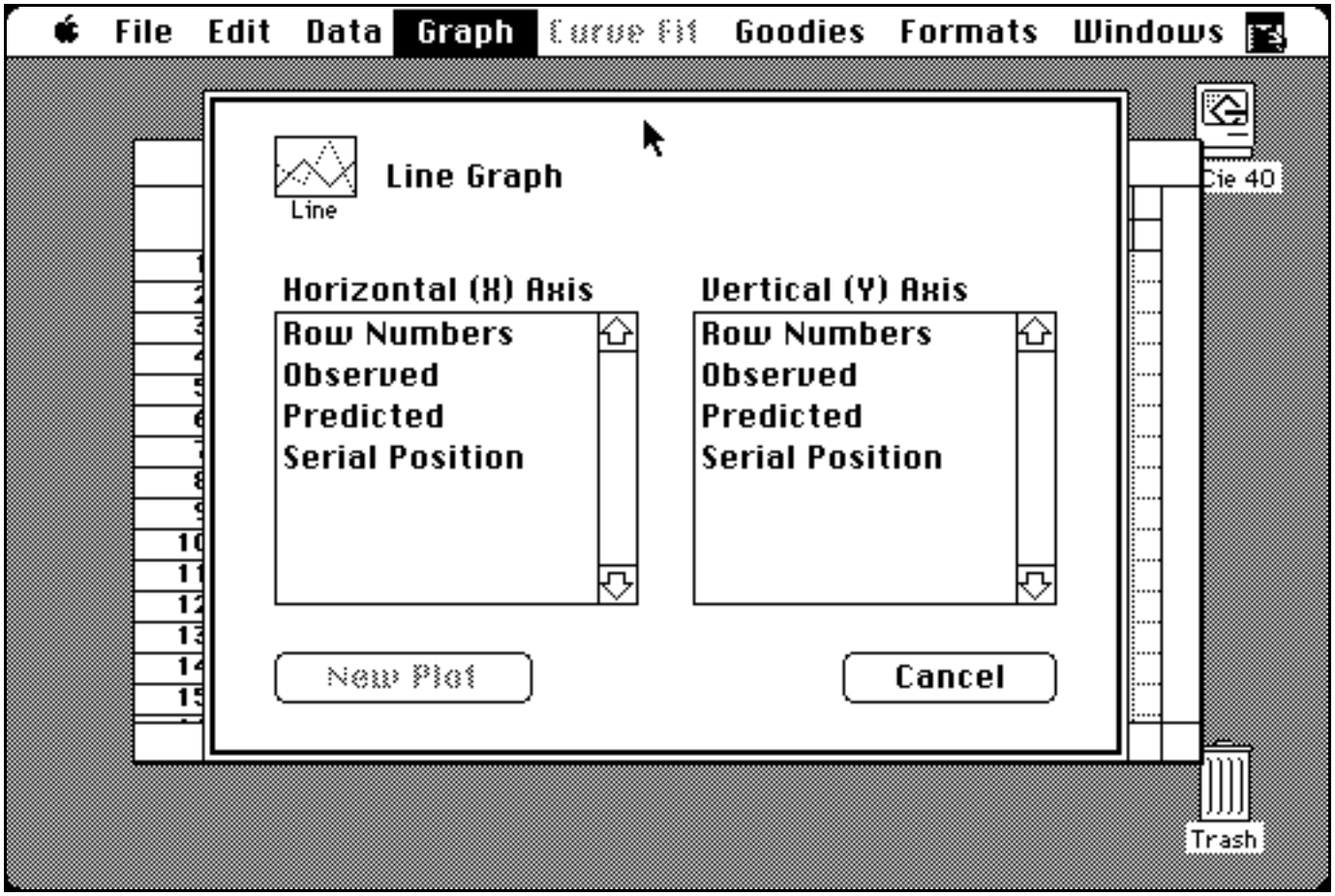



**[Screen-Shot 6-I]**




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | Observed | Predicted | Serial Position | Column 4 | Column 5 |
| 1 | 802.25714 | 816.951 | 1.000 | | |
| 2 | 200.829 | 205.089 | 2.000 | | |
| 3 | 738.800 | 889.700 | 3.000 | | |
| 4 | 289.514 | 214.542 | 4.000 | | |
| 5 | 848.743 | 634.852 | 5.000 | | |
| 6 | 271.714 | 360.569 | 6.000 | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |

**[Screen-Shot 6-II]**




**[Screen-Shot 6-III]**

*Figure 7*. Retrieval of task goals, PERFORM [ACTION, OBJECT, SPEC], cued by external screen representations.

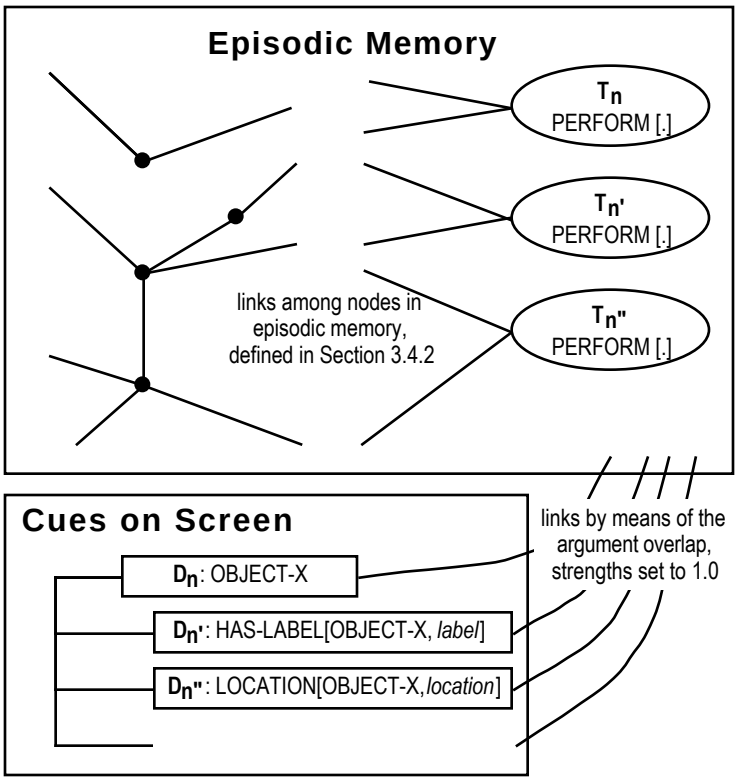


*Figure 8.* Activation values of potential task goals after a memory-retrieval cycle (unit 0.0001).

| Proposition label and task goal | Display I | Display II | Display III |
|---|---|---|---|
| TASK-10: PERFORM [LEARN, CRICKET-GRAPH] | 19 | 0 | 0 |
| TASK-20: PERFORM [PERFORM, TASK] | 0 | 0 | 1 |
| TASK-30: PERFORM [PLOT, DATA] | 8 | 3 | 0 |
| DO-IT-31: PERFORM [$ $ SPEC [LABEL [EXAMPLE-DATA], DOCUMENT]] | 51 | 0 | 0 |
| TASK-40: PERFORM [CREATE, GRAPH] | 3 | 39 | 14 |
| TASK-60: PERFORM [PLOT, $, AS-A-FUNCTION-OF [OBSERVED, SERIAL-POSITION]] | 0 | 1 | 4 |
| TASK-61: PERFORM [PLOT, OBSERVED, Y-AXIS] | 0 | 4 | 21 |
| TASK-62: PERFORM [PLOT, SERIAL-POSITION, X-AXIS] | 0 | 4 | 21 |
| TASK-70: PERFORM [MODIFY, GRAPH] | 0 | 19 | 7 |