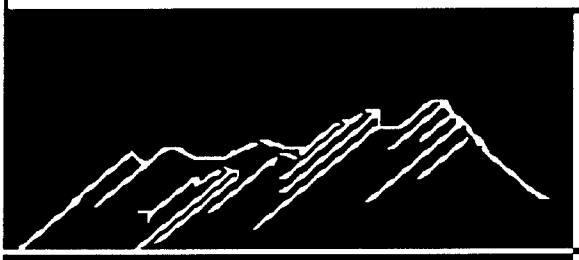
Institute of Cognitive Science



Technical Report

University of Colorado, Boulder

A Comprehension-Based Model of Exploration

Muneo Kitajima
National Institute of Bioscience and Human-Technology
1-1 Higashi Tsukuba Ibaraki 305, JAPAN

Peter G. Polson
Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0344, USA

ICS Technical Report 96-02

nodes in the integrated network are transferred to a network representing an episodic memory trace. The episodic memory trace accumulates the results of comprehension of each sentence.

The episodic memory trace is represented as a square matrix **M**. The rows and columns are all of the unique propositions generated during the sequence of construction–integration cycles required to process the text. The diagonal values of **M** represent the strength of the propositions in episodic memory. The off-diagonal values greater than zero are the strengths of the links between propositions that share one or more arguments.

Consider a proposition whose index in **M** equals i. Let a be the final activation value of the proposition and s be the self strength of the proposition in the constructed network. The new value of m_{ii} equals the old value plus sa^2. Let a' be the final activation value of another proposition in the current cycle whose index in **M** equals j, and let s_{ij} be the strength of the link in the constructed network between this proposition and the proposition with index i. The new value of m_{ij} equals the old value plus $s_{ij}aa'$. During the retrieval process, **M** is linked to nodes representing the retrieval cues. The resulting network is then integrated with the retrieval cues serving sources of activation. Kintsch and Welsch (1991) show that the activation value of each node in **M** correlates with the speed of recall of that node.

3. Action Planning Model of Display-Based Human–Computer Interaction

The LICAI model is an extension of Kitajima and Polson's (1995) model of action planning in display-based human–computer interaction. It is based on an extension of the construction–integration architecture to action planning (Mannes & Kintsch, 1991).

3.1 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 two kinds of goals: task goals and device goals (Payne, Squibb, & Howes, 1990). Kitajima and Polson assumed that each task goal is associated with one or more device goals. The device goals specify device states that must be achieved to satisfy an associated task goal. Kitajima and Polson (1995) input the sequence of task and device goals required by the model to perform a task.

The LICAI model describes the process of generating potential task goals from instructions. In simulating the experimental paradigm used by Franzke (1994), we assume that task goals are generated while reading instruction texts. However, device goals are learned during interactions with the interface. Thus, in exploration, the LICAI model assumes that users must generate task goals that enable them to generate correct actions *without* knowledge of the device goals.

3.2 Action Planning

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 include only limited information about the identity of each object and its appearance, including visual attributes (e.g., color, highlighting). They are elaborated by a stochastic long-term memory retrieval process which is taken from Kintsch (1988).

Action planning is modeled by two construction–integration cycles. The first construction–integration cycle selects three screen objects as possible candidates for next action. An important feature of Kitajima and Polson's action planning model is that the display representation is a detailed description of an actual large format display. Thus, the model's display representation can incorporate up to 100 screen objects. All screen objects are candidates for possible actions. During the initial construction phase, representations of all screen objects are combined with the goals and the elaborated display representation to construct a network. After integrating the network, the model selects the three most highly activated screen objects as candidates for the next action.

The second construction–integration cycle selects an action to be performed on one of the three candidate objects. During the construction phase of this second cycle, the model generates a network with representations of all possible actions on each candidate object. At the end of the second integration phase, the action planning model selects the most highly activated object–action pair as the next action to be executed.

Kitajima and Polson (1995) and Kitajima (1996) showed that the objects and action selected by each of these processes are largely determined by two factors:

1. Strong links from the goal propositions to propositions in the network that share arguments with the goal propositions.
2. The number of intervening propositions necessary to link goals to candidate objects or object–action pairs.

As a result, the action planning model selects candidate objects and object–action pairs closely related to the task and device goals. Device goals can directly specify a screen object, and thus can be directly linked to the screen object represented in the network. Task goals can be linked to screen objects through labels.

3.3 Action Planning Without Device Goals

Kitajima and Polson (1995) simulated a skilled user's performance of a version of the task learned by Franzke's (1994) subjects. Kitajima (1996) reported a series of simulation experiments that determined the constraints on the exact representation of task that will mediate successful action planning.

There were two findings in Kitajima's (1996) experiments. First, the action planning model always performed the correct actions when given a correct device goal. This was true even when there was no task goal. A device goal specifies a screen object and a desired attribute of the object (e.g., highlighted). The direct link between the device goal and the correct screen object caused the action planning model 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 second construction–integration cycle, the model almost always chose a single action on the most promising candidate object (i.e., the primary candidate object that was most highly activated in the first construction–integration cycle). The specification of the desired attribute could guide object–action selection when there were several possible actions.

Second, Kitajima (1996) showed that the model could generate the correct actions even when it was only given a task goal. However, the action selection process was not robust as compared with the case when the device goal was present. In order for the model to make stable correct object–action selection, the task goal had to be specific enough to establish a direct link to the correct screen object 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) showed that the action planning model could generate the correct action sequence with no device goal. Thus, Kitajima and Polson's (1995) action planning model could perform a task with a new program if we assume that users can generate required task goals. However, there are strong constraints on the exact form of successful task goals.

4. The LICAI Model

4.1 Overview

In this section, we present a detailed description of the LICAI model. The LICAI model simulates a situation where a user reads instructions and then attempts to perform a task using the application. The simulation consists of two phases: instruction taking and action planning. The model comprehends the instructions and stores task goals in long-term episodic memory. The model then retrieves task goals from episodic memory cued by successive displays generated by the application. The action planning processes generate the action sequence specified by the retrieved task goal. Goal formation and action planning are not interleaved because the application display is not available during the reading of the instructions, and the instructions are not available when the model is interacting with the application.

4.2 Experimental Task and Instructions

Franzke (1994) had experienced Macintosh users perform the task of creating a new graph with a novel graphing application, Cricket Graph I[2] or III[3], or one of two forms of the EXCEL 3.0[4] interface. The graphing task was divided into two subtasks. The first was to create a default graph by opening a document containing data to be plotted, selecting the correct graph style (e.g., line graph) from a menu, and assigning the designated variables to the X- and Y-axis.

Participants read the instructions and then attempted the first subtask. The second subtask was to edit the default line graph. The edits were done in a specific order. The descriptions of the edits were terse. Participants learned to do all subtasks by exploration. If they had not made any progress toward the next correct action for more than 2 minutes on a particular step, they were given brief hints to direct their attention to specific objects on the screen like graph menu or the legend.

Instructions used in our simulations are a simplification of the first instructions used by Franzke (1994). The following is the instruction text used in our simulation. It consists of six sentences:

1. In this experiment you are going to learn a new Macintosh application, Cricket Graph, by exploration.
2. The task you are going to perform will be presented to you as a series of exercises.
3. The data you are going to plot is contained in a Cricket Graph document, "Example Data."

---

[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.

4. Your overall goal is to create a new graph that matches the example graph shown here in the instructions.
5. Your first exercise is to plot the variable "Observed" as a function of the variable "Serial Position."
6. After you have created a new graph, you will modify it so that it more closely matches the example given in your instructions.

4.3 Schemata for Instruction Comprehension

The LICAI model assumes that goal formation processes are analogous to Kintsch's (1988; Kintsch & Greeno, 1985) model for solving word problems. The goal formation processes take a semantic representation of task instructions as input and combine this representation with inferences generated by highly specialized schemata to construct task goals.

Comprehending instructions to perform a task on a computer requires that the text base defined by each of the above sentences be elaborated. The LICAI model assumes three types of strategies for instruction comprehension, all described as schemata (Kintsch, 1988). The first type is a global *instruction reading schema* that represents the top level strategy used by a reader in processing text describing tasks to be performed by the reader. All verbs with an implicit subject of the reader [YOU] are mapped into a text base proposition of the form DO [YOU, verb, object]. The second type is *task-domain schemata*. These schemata generate elaborations of descriptions of the task to be performed. The third is *goal-formation schemata* that generate additional elaborations that are propositions specifying task goals which control the action planning process.

4.3.1 Task-Domain Schemata

Strategies defined by task-domain schemata are triggered when propositions in the original text base or by the elaborations from other task-domain schemata satisfy certain conditions. All these elaborations are added to the network during the construction phase of the current cycle. These schemata describe the users' specialized knowledge of the task-domain and are independent of the application interface.

4.3.1.1 Example from Data Graph Task-Domain

We illustrate the elaboration processes performed by task-domain schemata by tracing the elaboration of Sentence 5. The text for this sentence is:

> Your first exercise is to plot the variable "Observed" as a function of the variable "Serial Position."

The text base for Sentence 5 is[5]:

P51 EXERCISE
P52 FIRST [P51]
P53 OBSERVED
P54 ISA [P53, VARIABLE]
P55 SERIAL-POSITION

---

[5] We followed the guideline to encode textual sentences defined by Bovair and Kieras (1985). They provided a guide to propositional analysis for research on technical prose. For example, P51: EXERCISE, represents the existential. P52: FIRST [P51] is equivalent to FIRST [EXERCISE], representing "EXERCISE is-the-FIRST."

P56 ISA [P55, VARIABLE]
P57 DO [YOU, PLOT, P58]
P58 AS-A-FUNCTION-OF [P53, P55]
P59 ISA [P57, P51]

According to Kieras and Bovair (1985), P57 would be expressed as PLOT [P58]. The global instruction reading strategy assumes that verbs that are possible actions of the reader have an implied subject of [YOU] and that P57 is represented as DO [YOU, PLOT, P58].

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

Plot Schema

IF ([AS-A-FUNCTION-OF [*ARG-1, ARG-2*]) ==>
[ROLE [*ARG-1*, DEPENDENT-VARIABLE],
[ROLE [*ARG-2*, INDEPENDENT-VARIABLE]

Put Dependent Variable Schema

IF ([ROLE [*ARG*, DEPENDENT-VARIABLE]) ==>
ON [*ARG*, Y-AXIS]

Put Independent Variable Schema

IF (ROLE [*ARG*, INDEPENDENT-VARIABLE]) ==>
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 the above 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 propositions.

P57-0 AND [P57-1, P57-2]
P57-1 DO [YOU, PLOT, P62]
P57-2 DO [YOU, PLOT, P63]

4.3.1.2 Example from Text Editing Task-Domain

The instructions for Franzke's (1994) second subtask, editing the default graph, were very terse descriptions of each edit. An example is:

Editing Instruction: *Change the legend text to Geneva, 9, bold*

This cryptic instruction has to be elaborated before the edit task can be understood and executed. For example, "Geneva" has to be identified as the new font of the edited legend text. We assume experienced users of modern word processing systems have specialized task schemata called *Text Attributes Schemata* that transform such terse editing commands into comprehensible instructions.

The text base for the original editing instruction is:

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 explicitly identify which text attributes are to have the values Geneva, 9, and Bold. The inferences are guided by the following *Text Attributes Schema*.

Text Attributes Schema:

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

IF (PROPERTY [TEXT, $, *ARG*] & ISA [*ARG*, NUMBER]) ==>
PROPERTY [TEXT, SIZE, *ARG*]

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

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

P93-1 PROPERTY [TEXT, FONT, GENEVA]
P95-1 PROPERTY [TEXT, SIZE, 9]
P97-1 PROPERTY [TEXT, STYLE, BOLD]

4.3.2 Task Goal Formation Schemata

Task goal formation schemata generate task goals used by the action planning processes. We assume two kinds of goal formation schemata, *TASK Schema* and *DO-IT Schema*. 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 task domain. *DO-IT Schema* elaborates propositions representing information about device.

4.3.2.1 TASK Schema

The following is the text base for Sentence 1 generated by standard parsing strategies and the global instruction reading schema. The text for this sentence is:

> In this experiment you are going to learn a new Macintosh application, Cricket Graph, by exploration.

The text base is:

P10 IN-EXPERIMENT [P17]
P11 YOU
P12 NEW [P13]
P13 MACINTOSH [APPLICATION]
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.

TASK Schema requires *VERB* in DO [YOU, *VERB*, *OBJECT*] be a kind of TASK-ACTION. Examples of TASK-ACTION would include PLOT, CHANGE, and CREATE.

TASK Schema has the following form:

TASK Schema:

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

4.3.2.2 DO-IT Schema

In Franzke's (1994, 1995) experiment, participants were given instructions like "Please click on 'General Instructions' to start the experiment." Observe that following such instructions involves nontrivial inferences. '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 that is on the button on the bottom of the first page of the instructions.

When *VERB* in propositions in the form DO [YOU, *VERB*, *OBJECT*] is a kind of DEVICE-ACTION, such as click, drag, and release, *DO-IT Schema* elaborates these propositions into propositions representing task goals.

DO-IT Schema has the following form:

DO-IT Schema I:

IF (DO [YOU, *VERB*, *OBJECT*] & ISA [*VERB*, DEVICE-ACTION]) ==>
PERFORM [DEVICE-ACTION, DEVICE-OBJECT, DEVICE-SPECIFICATION]
DEVICE-ACTION [*VERB*]
DEVICE-OBJECT [*OBJECT*]
DEVICE-SPECIFICATION [*list of specifications*]

The arguments in DEVICE-SPECIFICATION refer to propositions that modify *OBJECT*.

The instruction text used in Franzke (1994, 1995) "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, *DO-IT Schema* is used to generate the following propositions for representing a task goal.

PERFORM [S21, S22, S23],
S21 DEVICE-ACTION [CLICK],
S22 DEVICE-OBJECT [$] ; undefined
S23 DEVICE-SPECIFICATION [DEVICE-LABEL [General Instructions]]

There would be cases where "click" in the above text were replaced with a representation of general actions such as "act on" that do not directly indicate device actions, whereas a screen object to be acted on is indicated. We assume another version of *DO-IT Schema* for these cases.

DO-IT-Schema II:

IF (HAS-LABEL [*OBJECT*, *LABEL*]) ==>
PERFORM [$, DEVICE-OBJECT, DEVICE-SPECIFICATION]
DEVICE-OBJECT [*OBJECT*]
DEVICE-SPECIFICATION [DEVICE-LABEL [*LABEL*], *list of specifications*]

4.4 Comprehension of Instruction Texts

The LICAI model processes the instructions sentence by sentence. Global instruction reading schema, task domain schemata, and goal-formation schemata elaborate the text bases defined by the original sentences. We have already described how the elaborated text base is constructed. The propositions representing the elaborated text base are then linked and integrated. The result of integration of the current network is carried over to the processing of the next sentence. In the following, we describe how the propositions in the elaborated text base are linked and integrated,

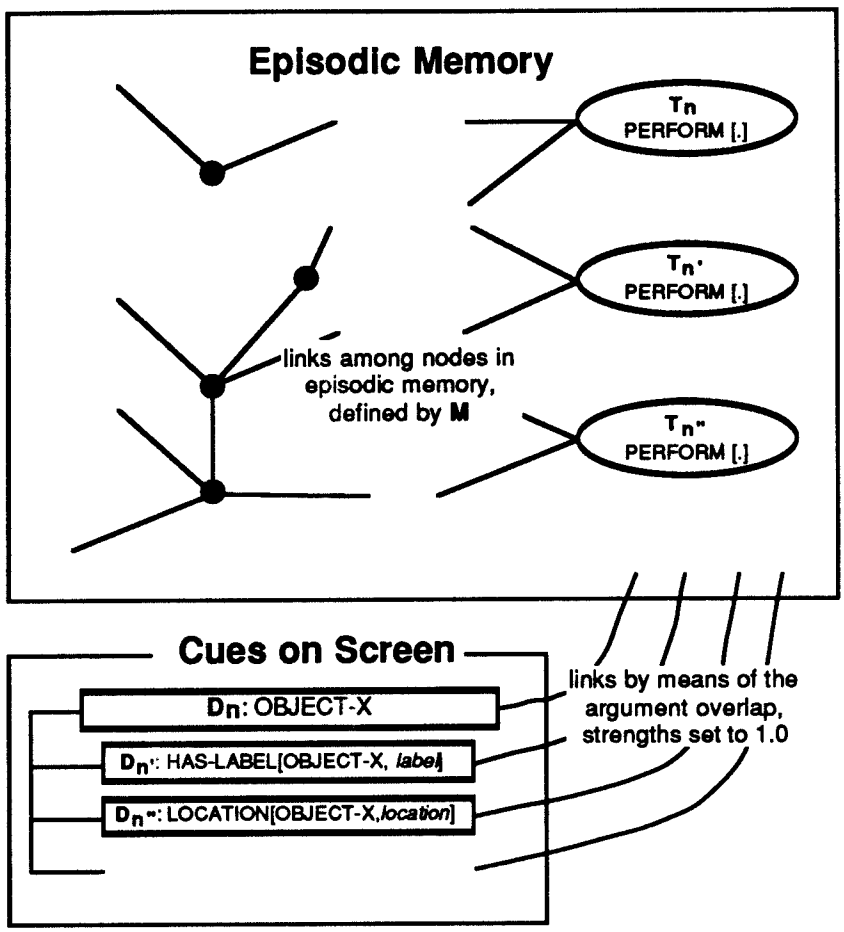


Figure 2 – Retrieval of task goals, PERFORM [ACTION, OBJECT, SP external screen representations.

4.5 Goal Formation: Retrieval of Goal Schemata Instances

After processing the instructions, the model retrieves a task goal from episodic memory using the application displays as retrieval cues. Task goals shown in Table 1are stored in the episodic memory. The following is the initial sequence of displays generated during the execution of the first subtask using Cricket Graph I. Each display was represented as a collection of screen objects. Each screen object is represented as a few propositions. See Kitajima and Polson (1995) for details.

The displays are:

1) Beginning of Task: The desk top with two icons, "Cricket Graph" and "Example Data"
2) After Launch of Cricket Graph: A display with menu items from Cricket Graph, "Data" and "Graph," and labels from spreadsheet, "Serial Position" and "Observed"
3) The Variable Selection Dialog Box: A display with two pairs of "Serial Position" and "Observed"

Figure 2 describes schematically how the retrieval process was simulated. The episodic memory network represents the outcome of comprehension of the input text, including generated nodes representing potential task goals. When a display appears, the display objects serve as retrieval cues for the task goals. The display cues are represented as a set of propositions that convey their perceptual information, such as label, highlighting, and so forth. For example, display (1) is represented as seven propositions:

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]

When the arguments of the display propositions overlap with those in episodic memory, they are connected by a unit strength link. 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 every iteration cycle. The other nodes, $D_{n'}$, ..., are initially set to 1.0. The nodes in episodic memory are set to their self-strengths as their initial values. Table 2 shows the activation values each task goal proposition (i.e., PERFORM [ACTION, OBJECT, SPEC]) obtained as the result of integration measured by a unit of 0.0001.

Table 2 – Activation values of potential task goals after cued retrieval by screen objects. (unit 0.0001)

	display (1)	display (2)	display (3)
prop. label	Beginning of Task	After Launch of Cricket Graph	The Variable Selection Dialog Box
TASK-10	19	0	0
TASK-20	0	0	1
TASK-30	8	3	0
DO-IT-31	51	0	0
TASK-40	3	39	14
TASK-60	0	1	4
TASK-61	0	4	21
TASK-62	0	4	21
TASK-70	0	19	7

In display (1), DO-IT-31 was activated most highly. It defines a task goal: PERFORM [$, $, SPEC [LABEL [EXAMPLE-DATA], DOCUMENT]]. As the result, the action selection process would presumably point at "Example-Data" document icon and double-click it. The arguments in this task goal, EXAMPLE-DATA, overlaps completely with the label on the document icon. In addition, the supplementary argument DOCUMENT in the task goal reinforces the link between the task goal and the correct screen object. Observe that a proposition defining the correct screen object, D-202: ISA [OBJECT-20, DOCUMENT], overlaps with the task goal. The argument DOCUMENT had significant effect on the resultant activation pattern. It enabled the model to focus activation on the correct screen object. Unless DOCUMENT were specified in DO-IT-31, a competing task goal TASK-10 would have been activated equally highly as DO-IT-31. The argument CRICKET-GRAPH in TASK-10 overlaps completely with the label of the application icon.

A Comprehension-Based Model of Exploration

MUNEO KITAJIMA

National Institute of Bioscience and Human-Technology

1-1 Higashi Tsukuba Ibaraki 305, JAPAN
Tel: +81 (298) 54-6730
E-mail: kitajima@nibh.go.jp

PETER G. POLSON

Institute of Cognitive Science University of Colorado

Boulder, CO 80309-0345, USA
Tel: +1 (303) 492-5622
E-mail: ppolson@psych.colorado.edu

and how the results of integration of the current network is carried over to the process for the next sentence[6].

4.4.1 Construction and Integration of the Network

We start with Sentence 1. As described in Section 4.3.2, the text base representing Sentence 1 consists of 9 propositions representing the original text and 3 propositions generated by applying *TASK Schema*. These propositions shown in Figure 1 are connected by shared arguments. For example, P13 and P14 are linked by the shared argument APPLICATION. We assign 1.0 as the link strength. In Figure 1, the nodes representing the original text are depicted in oval, and those generated by *TASK Schema* are in rectangle. The links are represented by solid lines.

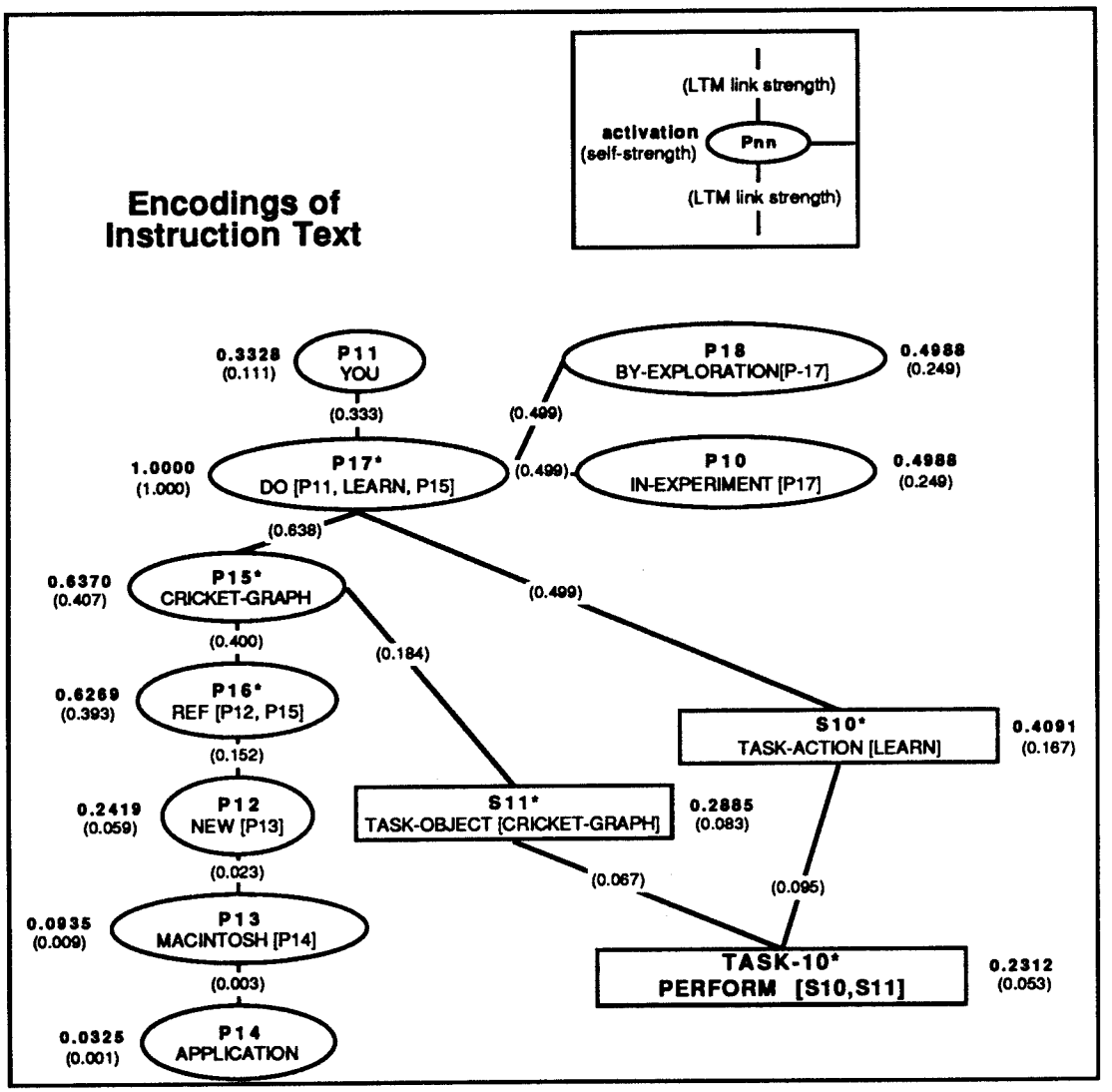


Figure 1 – A network representation for the first sentence.

---

[6] The simulation of processes of network construction, network integration, generation of episodic memory trace, and retrieval of a task goal was carried out by using a computer program developed by Mross and Roberts (1992).

The network, shown in Figure 1, is then integrated. The integration process is simulated by first giving a unit activation value to each of the textual nodes, P10 through P18, and then spreading activation iteratively in the network until the activation pattern settles. In each iteration the program scales the activation values so that the most highly activated node becomes 1.0. The numbers in bold face next to the nodes in Figure 1 show the activation values after the 11th iteration.

4.4.2 Forming Episodic Memory

When the network is integrated, the results are transferred to episodic memory. As episodic memory is empty when Sentence 1 has been integrated, a copy of nodes in the working memory shown in Figure 1 defines the episodic memory. Self strength of a node in the episodic memory is the square of the activation value of its corresponding node in working memory (see Section 2.3), thus, for example, the self strength of the node in the episodic memory corresponding to TASK-10 becomes $0.2312^2 = 0.05345$. The link strength between nodes in the episodic memory is defined by multiplying the activation values of the corresponding nodes. Figure 1 shows the values of self strengths and link strengths in parentheses.

4.4.3 Maintaining Coherence

After storing the result of comprehension of Sentence 1, and before starting processing of Sentence 2, the model carries over a few nodes in working memory for maintaining coherence. This is the requirement of the construction–integration model. There are two factors to be considered (Kintsch, 1988). One concerns the establishment of textual coherence; the other concerns taking advantage of the understanding of the task instruction that has been achieved so far. In the simulation, to fulfill the first the program maintains three most highly activated nodes for the next processing cycle. And for the second, the program carries over a single set of propositions that originated from a task goal formation schema. According to this rule, the program carries over P17, P15, and P16, and TASK-10, S10, and S11, to the cycle for Sentence 2.

4.4.4 Results of Comprehending Instructions

Six sentences in Section 4.2 were processed by repeating the above processes: representing the original sentence by propositions, applying task-domain schemata, and, if conditions were met, appropriate task-goal formation schemata added propositions to the current text base. The elaborated text base was then linked and integrated. The results of integration were incrementally stored in the episodic memory. The carry over rule was applied before initiating the above processes for the next sentence. Table 1shows task goal propositions that the task goal formation schemata generated by comprehending the example instructions.

Table 1 –Task goals generated by comprehending the example instructions.

Sentence	prop. 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]

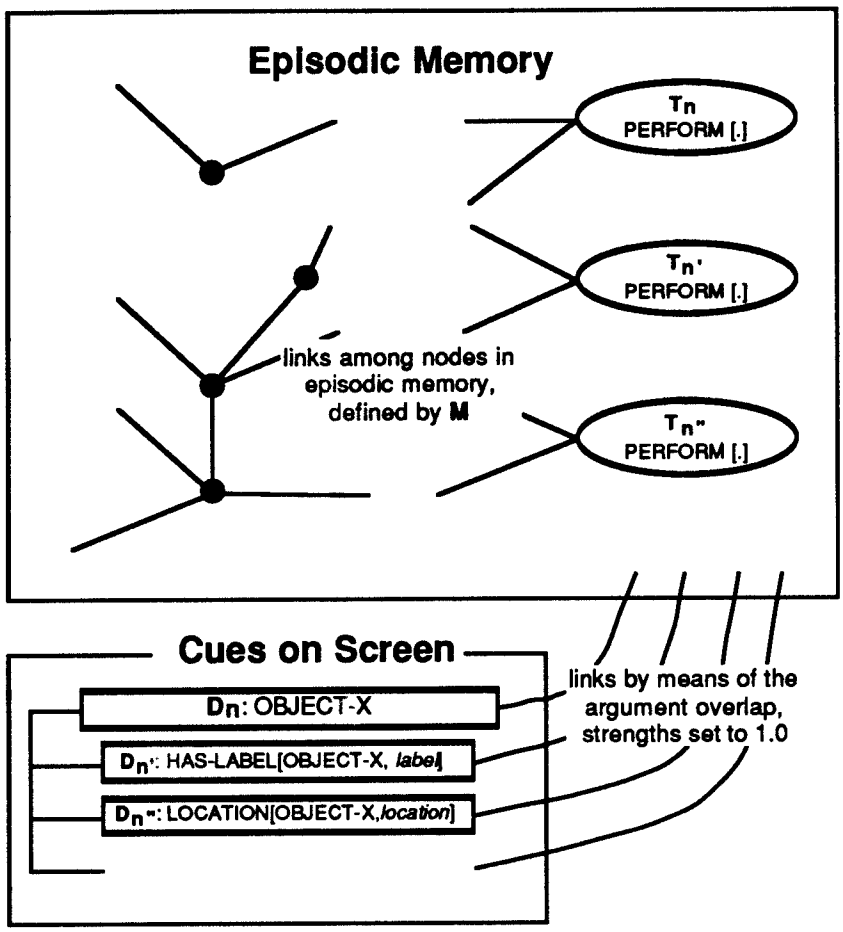


Figure 2 – Retrieval of task goals, PERFORM [ACTION, OBJECT, SP external screen representations.

4.5 Goal Formation: Retrieval of Goal Schemata Instances

After processing the instructions, the model retrieves a task goal from episodic memory using the application displays as retrieval cues. Task goals shown in Table 1are stored in the episodic memory. The following is the initial sequence of displays generated during the execution of the first subtask using Cricket Graph I. Each display was represented as a collection of screen objects. Each screen object is represented as a few propositions. See Kitajima and Polson (1995) for details.

The displays are:

1) Beginning of Task: The desk top with two icons, "Cricket Graph" and "Example Data"
2) After Launch of Cricket Graph: A display with menu items from Cricket Graph, "Data" and "Graph," and labels from spreadsheet, "Serial Position" and "Observed"
3) The Variable Selection Dialog Box: A display with two pairs of "Serial Position" and "Observed"

Figure 2 describes schematically how the retrieval process was simulated. The episodic memory network represents the outcome of comprehension of the input text, including generated nodes representing potential task goals. When a display appears, the display objects serve as retrieval cues for the task goals. The display cues are represented as a set of propositions that convey their perceptual information, such as label, highlighting, and so forth. For example, display (1) is represented as seven propositions:

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]

When the arguments of the display propositions overlap with those in episodic memory, they are connected by a unit strength link. 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 every iteration cycle. The other nodes, $D_{n'}$, ..., are initially set to 1.0. The nodes in episodic memory are set to their self-strengths as their initial values. Table 2 shows the activation values each task goal proposition (i.e., PERFORM [ACTION, OBJECT, SPEC]) obtained as the result of integration measured by a unit of 0.0001.

Table 2 – Activation values of potential task goals after cued retrieval by screen objects. (unit 0.0001)

	display (1)	display (2)	display (3)
prop. label	Beginning of Task	After Launch of Cricket Graph	The Variable Selection Dialog Box
TASK-10	19	0	0
TASK-20	0	0	1
TASK-30	8	3	0
DO-IT-31	51	0	0
TASK-40	3	39	14
TASK-60	0	1	4
TASK-61	0	4	21
TASK-62	0	4	21
TASK-70	0	19	7

In display (1), DO-IT-31 was activated most highly. It defines a task goal: PERFORM [$, $, SPEC [LABEL [EXAMPLE-DATA], DOCUMENT]]. As the result, the action selection process would presumably point at "Example-Data" document icon and double-click it. The arguments in this task goal, EXAMPLE-DATA, overlaps completely with the label on the document icon. In addition, the supplementary argument DOCUMENT in the task goal reinforces the link between the task goal and the correct screen object. Observe that a proposition defining the correct screen object, D-202: ISA [OBJECT-20, DOCUMENT], overlaps with the task goal. The argument DOCUMENT had significant effect on the resultant activation pattern. It enabled the model to focus activation on the correct screen object. Unless DOCUMENT were specified in DO-IT-31, a competing task goal TASK-10 would have been activated equally highly as DO-IT-31. The argument CRICKET-GRAPH in TASK-10 overlaps completely with the label of the application icon.

In display (2), TASK-40, PERFORM [CREATE GRAPH], was activated most highly. This task goal is specific enough for the action-selection process to correctly attend to the GRAPH menu item. In display (3), TASK-61 and TASK-62 were equally activated most highly, to generate appropriate task-goals, PERFORM [PLOT, OBSERVED, Y-AXIS] and PERFORM [PLOT, SERIAL-POSITION, X-AXIS], respectively. These goals do work for selecting correct actions in the given display. See Kitajima (1996) for more detail.

In summary, in this section we described a simulation in which a user comprehends an instruction text and stores it as episodic long-term memory trace. The result of comprehension is tested against three kinds of display cues that mimic situations on which Franzke (1995) experimented and showed that appropriate task goals were retrieved that would lead to correct actions.

5. Evaluation of the LICAI Model

We evaluate the LICAI model using data from Franzke (1994, 1995). The LICAI model only makes coarse grain predictions about the behavior of Franzke's participants. The instructions and the schemata assumed by the model may or may not enable them to generate the correct task goal. If participants construct and retrieve the correct task goal, the action planning will generate the correct action sequence. However, the LICAI model does not describe the search behavior that occurs if the task goal construction process fails (Rieman et al., in press). In fact, Rieman and colleagues found that people will do some exploration of the interface even when the instruction enables them to construct a correct task goal and they take the correct actions within 15 to 30 seconds.

5.1 Task Descriptions, Object Labels, and Number of Screen Objects

Franzke (1995) presented an analysis of her dissertation (Franzke, 1994) in which she collapsed interfaces and tasks. She focused on the effects of the relationships between task descriptions in the instructions and object labels and number of screen objects that were possible targets for an action as a function of practice. We did not simulate all of the tasks and the interfaces used in Franzke's experiments. Our goal is to show that the LICAI model can provide a qualitative account of Franzke's (1995) figures 5, 6, 7, and 8.

Franzke partitioned the possible relationships between a task description and the label of the correct screen object into four categories. The first was a perfect match between the label and the task descriptions. The second was that the labels were synonyms of the task description. The third was that an inference was required to link the two. The fourth was that there was 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.

5.1.1 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 on the first attempt at a task was well under 30 seconds for both match and synonym categories. Participants used overlap between task descriptions contained in the instructions with labels on menus, buttons, and other interface objects during exploration. The degree of success of the label following strategy depended on the quality of the label and on the number of competing screen objects.

Franzke (1995, Figure 7) found an interaction between number of objects (2–10) on the screen and quality of the label match (good, 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 independent of the number of competing screen objects. There was little or no effect of practice for good labels.

These results are consistent with the LICAI model. If instructions contained a description of either an action or an object that matched a screen object label, those labels were preserved when the propositional representation of the instructions was mapped into a task goal. The links between the task goal and the correct screen object can mediate performance of the correct action, even when there are a large number of competing screen objects if the matching label is unique.

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

5.1.2 Direct Manipulation

Franzke (1995) found that participants had trouble discovering direct manipulation actions where no label links the correct screen object and the task description. Examples included double-clicking on a title or axis label to edit it, clicking on an object in the tool bar, and drag and drop. 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 enable them to infer that an object could be edited by double-clicking on it. Thus, participants could not perform such tasks even if they generated the correct task goals from the instructions. The LICAI predicts such failures because of the missing direct links between the task goal inferred from the instructions and the correct screen object.

5.2 Franzke's (1994) Instruction Texts

Franzke 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 in our simulation described in Section 4.2. Such texts would generate multiple task goals.

The texts describing the edits to be performed on the default graph were short and telegraphic. The comprehension problem here was to make the inference necessary to generate a comprehensible task description and task goals that overlap with screen object labels. We first discuss the comprehension of these telegraphic task descriptions.

5.2.1 Task-Domain Schemata

The critical step in performing the first subtask, creation of the default graph, is interaction with a dialog box containing two scrolling lists for the label of variables to be placed on the X- and Y-axis. 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" using the task-domain schemata described in Section 4.3.1.1.

Terwilliger and Polson (1996) found clear evidence that participants also make such transformations. They measured the time experienced Macintosh users who have never used a graphing problem took to interact with two forms of the variable selection dialog box. There were two versions of task instructions. XY instructions read "create a graph with Serial Position on the

X-axis and Observed on the Y-axis." FN instructions read "create a graph of Observed as a function of Serial Position." There were two versions of the dialog box with two scrolling lists for variable selection. 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 (1996) found task took less time with the XY version of the dialog box for both XY and FN version of the instructions. In addition, they recorded think-aloud protocols while participants did the task. Many participants verbalized the transformation of FN to XY. These results strongly support the concept of task-domain schemata. Furthermore, they support the primary assumption of the label following strategy that the label on the interface must match user descriptions of their tasks.

To perform the task "change the legend text to Geneva, 9, bold" in Cricket Graph I, participants had to first double-click on the legend, which opened a dialog box. Most participants had trouble discovering this initial action and had to be given hints. Once the dialog box was open, participants had no trouble completing the task. The dialog box contained three scrolling lists labeled font, size, and style. We described in Section 4.3.1.2 how the model generates the three task goals necessary to interact with the three scrolling lists. The model can perform each subtask specified by the original text because the *Text Attributes Schema* generates task goals that link directly to a scrolling list title and to the relevant item in the scrolling list in the edit dialog box.

5.2.2 Long Instructions and Multiple Task Goals

Franzke's (1994) instructions for the initial subtask of creating the default graph were a more detailed version of the instructions used in our simulation. They included general information about the experiment as well as details about which variables were to be placed on X- and Y-axes. Even participants in the Cricket Graph I and III conditions had some difficulty with these initial steps in the task. Participant had to return to the instructions to obtain critical items of information like variable names.

This is surprising because the relevant parts of the instructions used terms that provided a perfect match for the label following strategy. As the LICAI model predicts, however, these instructions would generate multiple task goals. Retrieving the goals is a brittle process that is heavily dependent on the concrete display representation.

6. Models of Exploration

6.1 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 problem solution. An early example of this class of models 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 General Problem Solver (Ernst & Newell, 1969).

The LICAI model 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 done in the comprehension component. Kintsch (1988) argues that children's difficulty with arithmetic word problems is not in performing the basic arithmetic operations but are linguistic difficulties. Hudson (1983) showed that linguistic factors can have a powerful effect on problem solving success. Students who have difficulties with word problems can be shown to have accurate manipulation skills, either arithmetic or algebraic.

Kitajima and Polson's (1995) action planning model requires explicit task goals to generate correct actions. The form of useful goals are strongly constrained by the surface features of the interface like labels in dialog boxes and on menus. We assume that skilled users of an application have specialized comprehension knowledge directly analogous to Kintsch and Greeno's arithmetic schemata. These schemata enable users to construct the required, specific goal representation from text descriptions of their task. The more general claim that we are making is that 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.

6.2 Search-Based Models and the LICAI Model

The construction–integration theory was originally developed to account for reading, a form of 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. The theory has components that would enable it to exhibit search behavior: goals, a mechanism to choose between alternative actions, and the ability to react to the consequences of a selected action.

LICAI starts in comprehension mode reading the instructions and storing alternative task goals in memory. It then switches to action planning mode and attempts to execute the correct actions by using cues generated by successive displays to retrieve task goals from memory. LICAI cannot interleave comprehension and problem solving. The underlying construction–integration architecture does not support the control structures necessary to pass control back and forth between action planning and comprehension modes based on the current state of either comprehension or action planning processes.

In this section we review alternative theories of exploration that do have such control structures. We focus on IDXL, a SOAR model developed by Rieman et al. (in press). IDXL simulates learning by exploration of the Cricket Graph task simulated by the LICAI model, and it integrates much recent research on learning by exploration (Howes, 1994; Howes & Young, in press; Rieman, 1994; Rieman et al., 1994).

There are three important differences between LICAI and IDXL. The first 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) did a detailed analysis of the actual exploratory behavior of Franzke and Rieman's (1993) participants who learned Cricket Graph I. Rieman 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., in press). 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 operators include scanning, pointing, dropping pull down menus, releasing on a menu item, and the like. The comprehension processes operate on the currently attended-to screen object and compare its label to a currently focused task goal. The output of the comprehend operators are exact match, recognize a synonym to the task word, recall, analogy, ask for instruction, and envision consequences of an action.

The LICAI action planning processes model the same behavior at a much 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 currently focused task goal and the correct screen object label, and the user discovers the correct action after a significant amount of search. LICAI would simply fail to generate the correct action sequence.

The second difference is that IDXL interleaves comprehension and action planning. Both are forms of progressive deepening search mechanisms, and both action planning and comprehension operators are ordered by cost. Scanning actions like moving the mouse cursor and pulling down a menu have low cost. Releasing on an incorrect menu item can be costly. Comprehension operators are applied to screen objects put in the focus of attention by the scanning operators. Initially, low cost comprehension operators are applied like a test for an exact match to the currently focused task goal. Failure causes IDXL to continue scanning. When the model's focus of attention returns to the same screen object, results of the previous comprehension operators are recalled and lead to the 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 which is a multipart and hierarchical representation in the form of subject–verb–object format. IDXL assumes a shifting internal focus of attention, though it has not been modeled yet, to define a currently focused task goal (Rieman et al., in press).

It is clear that further developments of LICAI must focus on extending the underlying architecture with control mechanisms that enable interleaving comprehension and search. We must develop a principled account of the search behavior described by Rieman (1994) and simulated by IDXL. However, the resulting models would be different from IDXL. IDXL's comprehension operators elaborate the user's representation of objects on the interface with a fixed representation of currently focused task goal. These operators are similar to the elaboration phase in the Kitajima and Polson (1995) action planning model. An extension of LICAI would be search of alternative interpretations of the instructions that would 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 the simulation reported in this paper. In both cases, the comprehension processes generate multiple task goals. Our interpretation of much of the search behavior observed by Franzke and Rieman (1993) is that Franzke's participants are searching possible interpretations of the instructions to find useful task goals. LICAI simulates a participant who completely processes all of the instructions. However, Hayes and Simon (1974) observed that participants have a strong tendency to skim instructions. They attempt to solve the problem without all of the necessary information. They are forced to return to the instructions, often several times, to acquire the necessary knowledge. Thus, we argue that part of the search behavior described by Franzke and Rieman is caused by incomplete and/or multiple interpretations of the

instructions.

7. Discussion

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

This section describes fundamental results that come from the LICAI model and their implications for exploration in HCI.

7.1 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. There are two kinds of schemata defined in terms of their content. The task-domain schemata elaborate the original statement of the task description. Examples include *Plot*, *Put*, and *Text Attributes Schema*. These schemata describe the user's specialized knowledge of task-domain, and they are independent of the application interface. The goal-formation schemata (*TASK* and *DO-IT*) that generate goals used by the action planning process. The LICAI model claims that instruction reading is a strategic process that uses these schemata. The *Plot* and *Put Schemata* are supported by Terwilliger and Polson's (1996) results.

7.2 Label Following

The links in the network between arguments in propositions representing the goals and propositions representing the labels on screen objects (e.g., menu items and button labels) represent label following. As Kitajima's (1996) simulation showed, the action planning component is quite rigid. Label following is a requirement of the construction–integration architecture for successful action planning. Skilled action planning is driven by representations of goals that are strongly constrained by the superficial details of the application interface and the interaction conventions of the host operating system.

The LICAI model 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 defining task goals are not necessarily present in the original instruction texts. The original instruction texts can be elaborated by task-domain schemata such as *Text Attributes Schema*. Thus, the LICAI model predicts the label following behavior observed in new users of applications by Franzke (1995), Terwilliger and Polson (1996), and other investigators.

7.3 Multiple Task Goals

Another fundamental result dictated by the underlying architecture is that users studying instructions or an example will generate multiple task goals. Recall that the construction process is

Abstract

This paper describes LICAI, a model that simulates peforming tasks by exploration where the tasks are given to the user in the form of written exercises that contain no information about the correct action sequences. LICAI's comprehension processes and the action planning processes are based on Kintsch's construction–integration theory for text comprehension. The model comprehends the instructions and generates goals which are then stored in memory. The action planning process is controled by goals retrieved from memory cued by displays generated by the application.

The model assumes that the instruction comprehension is a strategic process; instruction texts must be elaborated using specialized strategies that guide generation of goals. Representations of goals that lead to correct actions are restricted by the construction–integration architecture. The model predicts that successful exploration requires perfect matching of goal representation and the label on the correct object. The model is evaluated by comparing predictions with the results from a laboratory study. Finally, implications to designing instruction materials and interface displays that facilitate exploration are discussed.

bottom up; there is no control process that forces 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 an 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.

7.4 Implications for Learning by Exploration

Our results have important implications for the development of training materials (Carroll, 1990) for interfaces that support learning by exploration (Wharton et al., 1994). We describe how a user's background knowledge constrains the content of instructional materials, and we show the mutual dependencies between interfaces that support learning by exploration and instructions that have as their intention to facilitate this form of learning.

7.4.1 The Minimalist Instruction Paradigm

Carroll (1990) summarizes an influential research program on the design and evaluation of training materials for application programs that led to the development of the Minimalist Instruction paradigm. The development of the Minimalist Instruction paradigm was stimulated by the then-surprising result that carefully designed, detailed training and reference materials for early 1980s versions of word-processors were unusable (e.g., Mack, Lewis, & Carroll, 1983).

However, the LICAI model predicts that comprehending and following instructions is an difficult task, 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 the LICAI model. Thus, instructions for new users must be detailed or incorporate extensive pretraining, like the Guide Tour[7] for the Macintosh interface, to explain how to use the mouse, how to select items from menus, and the like. Note that all of this knowledge is incorporated in the action planning component of the LICAI model.

Carroll's (1990) emphasis on active users and individuals' desires to learn by exploration are well founded (Rieman, 1994). However, the LICAI model shows that following written instructions is a complex task, analogous to doing arithmetic and algebra word problems. Successful instruction following requires the appropriate schema necessary to extract the information from the text required to guide generation of the correct action sequence. Thus, for the new user, following even the most carefully prepared instructions is a difficult and error-prone process.

Carroll's (1990) solution to the usability problems demonstrated by Mack et al. (1983) and other researchers was to develop 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 attempting to provide detailed step-by-step instructions. This approach also supports error recognition and recovery.

The LICAI model can be used to develop explicit design guidelines for the content of minimalist instructional material. Note that the minimalist paradigm and the LICAI model have a common

---

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

focus on the users' tasks. Task goals, representing what the user wants to achieve, drive the action planning process. Explicit, brief statements of relevant goals that can be understood by the user are a critical ingredient in supporting learning by exploration. Observe that many of the irrelevant task goals generated during the comprehension of the example instructions given in Section 4.2 were caused by attempts to provide some motivation for the user's task. Carroll explicitly recommends deletion of such irrelevant material.

However, the LICAI model makes clear the kinds of constraints that must be understood in following Carroll's (1990) design heuristic of minimizing the amount of written material contained in task instructions. Brief instructions, as recommended by Carroll, solve the problem of having to deal with multiple task goals. However, we also showed that comprehending terse instructions, like those used by Franzke (1995), require specialized background knowledge about the task and the interface.

The task description, "change the legend text to Geneva 9 bold," cannot be understood by someone who has no experience with a modern word-processor. Effectively minimizing the amount of written material requires careful attention to the action and display knowledge and schemata assumed in the target user population. We conjecture that a minimalist version of a complete instruction manual for a modern word-processor would have to assume that users have well developed versions of the *TASK* and *DO-IT Schemata* described in this paper. These schemata represent significant amounts of background knowledge (i.e., at least 6 months 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.

7.4.2 Cognitive Walkthroughs

Cognitive walkthroughs are a method for evaluating interfaces for ease of learning by exploration. The walkthrough is organized like structured walkthroughs widely used in the software development community (Yourdon, 1989). The cognitive walkthrough is based on a theory of exploratory learning (Polson et al., 1992) that is a predecessor to the LICAI model presented in this paper.

The cognitive walkthrough (Wharton et al., 1994) evaluates the effectiveness of the label following strategy and characterizes the background knowledge necessary to infer correct actions. 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 out some serious 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, the interface guides the user step-by-step through the procedure. The instruction for each step in a task are brief and contain information necessary for the user to select the next correct 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.

In many instructional situations, users are given a paragraph or two of instructions similar in form

and content to the example used in our simulations described in Section 4.2. Recall that the LICAI model generates multiple task goals when processing such instructions. Cues provided by the interface and/or the user must control the retrieval processes that generate the proper sequence of task goals. This retrieval process is not robust. We did other simulations. Minor changes in the details of the wording led to the retrieval of incorrect task goals.

7.5 Label Following and Minimalist Instructions

The LICAI model presented in this paper makes clear that there is an 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 LICAI model predicts that successful minimalist instructions must guide users to form task goals containing terms that overlap with the labels of the correct screen objects. Such task goals enable the action planning to select the correct action or action sequence without any previous instruction or detailed step-by-step guidance on what to do next. If the interface does not support label following, then the document designer will be forced to use step-by-step instructions. But Carroll (1990) and numerous other investigators have shown that people are reluctant, if not unwilling, to read and follow such step-by-step instructions. Furthermore, following step-by-step instructions is a difficult and error prone task.

8. 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 we have been able to show that it gives us important insights into the difficulties that individuals have in comprehending detailed technical documentation and provides further 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 for now 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 has forced us to focus on comprehension and the processes of goal formation. What we have shown in this paper is the kinds of background knowledge necessary to generate the highly constrained goals that are required by our model of action planning. The LICAI model is closely related to successful models of word arithmetic problem solving (Kintsch, 1988). The major thrust of both sets of models is that following instructions to carry out some procedure is a very difficult task and comprehension of instructions requires specific, specialized background knowledge.

9. Acknowledgments

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. Richard Young provided us with a detailed review of a much longer, early version of this paper. We thank him for helping us clarify many issues and shortening the paper. John Rieman, Clayton Lewis, and Walter Kintsch also made import contributions.

10. 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 & 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.

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

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

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, CO.

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). Natural training wheels: learning and transfer between two versions of a computer application. *Proceedings of Vienna Conference on Human Computer Interaction 93.* Sept. 20–22, 1993. Vienna, Austria.

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

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, & 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. (In press). Learning consistent, interactive, and meaningful task-action mappings: A computational model. *Cognitive Science.*

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

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

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

Kitajima, M., & Polson, P.G. (1996). A comprehension-based model of exploration. *Proceedings of CHI'96 Conference on Human Factors in Computing Systems* (pp. 000–000). New York: ACM.

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.

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. (in press). A field study of exploratory learning strategies. *ACM Transactions on Computer–Human Interaction*.

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. *Proceedings of CHI'94 Conference on Human Factors in Computing Systems* (pp. 438–444). New York: ACM.

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

Terwilliger, R.B., & Polson, P.G. (1996). Task elaboration or label following: an empirical study of representation in human-computer interaction. *Proceedings of CHI'96 conference on human factors in computing systems* (pp. 000–000). 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. (1994). The cognitive walkthrough

method: a practitioner's guide. In J. Nielsen & 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.

1. Introduction

The major focus of recent research on skill acquisition in human–computer interaction has been on learning by exploration (Carroll, 1990; Howes, 1994; Rieman, 1994, in press; Rieman, Young, & Howes, in press). Experienced users of a given environment (e.g., Macintosh or Windows95) learn new applications or extend their knowledge of applications they already use by task-oriented exploration. Formal training is rarely available, and there are usability problems with training and reference documentation (e.g., Compeau, Olfman, Sein, & Webster, 1995). Most users prefer to acquire new skills by exploration; they perform new tasks relevant to their work (Carroll, 1990).

A majority of current models of exploration are based on problem solving architectures like SOAR (Newell, 1990; Rieman et al., in press) 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., dropping a menu) in attempts to discover a correct action sequence.

Previous models of the label following strategy have taken goals as given and then described the resulting search behavior (e.g., Rieman et al., in press). However, models of exploration should go a step further and define the processes that generate the goals that determine exploratory behavior. Normally, tasks are given to users through various forms: narratives, written instruction texts, help messages, graphics images, or combinations of these forms. Whatever forms are 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.

The LICAI[1] model described in this paper focuses on goal formation. The model is based on the construction–integration architecture originally developed to account for comprehension and memory for texts. This architecture has been extended to account for action planning (Mannes & Kintsch, 1991), skilled display-based, human–computer interaction (Kitajima & Polson, 1995), and word problem solving (Kintsch, 1988). The LICAI model simulates an experimental paradigm used by Franzke (1994, 1995) in which experienced Macintosh users are introduced to a new application by performing a series of exercises described in written instructions by exploration. Comprehending instructions to form effective goals for exercises is critical for successful performance in Franzke's paradigm. Our approach is complementary to that of Rieman et al. (in press). The LICAI model accounts for Franzke's (1994) data by attributing users' likelihood of success on a given exercise to the difficulty of formulating the correct goal required by the model's action planning process.

This paper is an extended version of Kitajima and Polson (1996) and includes new simulation results. These results show that the comprehension-based goal formation processes can generate the goals required by Kitajima and Polson's (1995) action planning model.

---

[1] LICAI is acronym for the LInked model of Comprehension-based Action planning and Instruction taking. When LICAI is pronounced like "lick eye" or "Lee CHI (χ)", the pronunciation represents two-kanji character Japanese word, 理解, which means comprehension.

1. Introduction

The major focus of recent research on skill acquisition in human–computer interaction has been on learning by exploration (Carroll, 1990; Howes, 1994; Rieman, 1994, in press; Rieman, Young, & Howes, in press). Experienced users of a given environment (e.g., Macintosh or Windows95) learn new applications or extend their knowledge of applications they already use by task-oriented exploration. Formal training is rarely available, and there are usability problems with training and reference documentation (e.g., Compeau, Olfman, Sein, & Webster, 1995). Most users prefer to acquire new skills by exploration; they perform new tasks relevant to their work (Carroll, 1990).

A majority of current models of exploration are based on problem solving architectures like SOAR (Newell, 1990; Rieman et al., in press) 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., dropping a menu) in attempts to discover a correct action sequence.

Previous models of the label following strategy have taken goals as given and then described the resulting search behavior (e.g., Rieman et al., in press). However, models of exploration should go a step further and define the processes that generate the goals that determine exploratory behavior. Normally, tasks are given to users through various forms: narratives, written instruction texts, help messages, graphics images, or combinations of these forms. Whatever forms are 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.

The LICAI[1] model described in this paper focuses on goal formation. The model is based on the construction–integration architecture originally developed to account for comprehension and memory for texts. This architecture has been extended to account for action planning (Mannes & Kintsch, 1991), skilled display-based, human–computer interaction (Kitajima & Polson, 1995), and word problem solving (Kintsch, 1988). The LICAI model simulates an experimental paradigm used by Franzke (1994, 1995) in which experienced Macintosh users are introduced to a new application by performing a series of exercises described in written instructions by exploration. Comprehending instructions to form effective goals for exercises is critical for successful performance in Franzke's paradigm. Our approach is complementary to that of Rieman et al. (in press). The LICAI model accounts for Franzke's (1994) data by attributing users' likelihood of success on a given exercise to the difficulty of formulating the correct goal required by the model's action planning process.

This paper is an extended version of Kitajima and Polson (1996) and includes new simulation results. These results show that the comprehension-based goal formation processes can generate the goals required by Kitajima and Polson's (1995) action planning model.

---

[1] LICAI is acronym for the LInked model of Comprehension-based Action planning and Instruction taking. When LICAI is pronounced like "lick eye" or "Lee CHI (χ)", the pronunciation represents two-kanji character Japanese word, 理解, which means comprehension.

1.2 Outline of the Paper

We begin by describing the construction–integration theory of text comprehension. Next, we summarize research on models of word problem solving developed by Kintsch and collaborators. The LICAI model is an extension of Kintsch and Greeno's (1985) theory of arithmetic word problem solving. We then describe Kintsch and Welsch's (1991) model of memory for text. Kitajima and Polson's (1995) action planning model of skilled, display-based human–computer interaction is presented in the next section.

We then describe our comprehension-based model of goal formation that maps an initial description of a task onto the specific goals required by Kitajima and Polson's (1995) action generation process and report the results of a simulation experiment evaluating the model. The LICAI model stores multiple possible task goals in episodic memory and uses cues from displays generated by the application to retrieve the task goals that guide action planning. We evaluate the LICAI model using Franzke's (1995) results. We then review recent models of exploration in human–computer interaction in light of our results. Finally, we summarize the implications of our results for practice. These results have importation implications for the design of interfaces and training materials that support exploration (Carroll, 1990; Wharton et al. 1994).

2. Construction–Integration Theory of Word Problem Solving and Memory for Text

2.1 The Construction –Integration Framework

Kintsch (1988) proposed a model of comprehension that combines elements of symbolic and connectionist models of cognitive processes. Text comprehension is a cyclic process where readers process a sentence, or a major constituent of a long sentence, during a single cycle; reading a text involves a sequence of such cycles (Kintsch & van Dijk, 1978).

The construction–integration cycle is a two-phase process. In the first phase, a network of propositions is created that contains possible alternative meanings of the current sentence or fragment. This *construction process* generates an associative network whose nodes are propositions representing the input text, elaborations of the input text retrieved from long-term memory, and propositions carried over from the last cycle. The elaborations come from two sources. The first is propositions retrieved from long-term memory by a stochastic, associative retrieval process (Raaijmakers & Shiffrin, 1981). The second are inferences generated by schemata triggered by propositions in the original input text. Construction is a bottom-up process that is not guided by context. Thus, at the end of the construction process the model has multiple possible meanings for the input text. The *integration process*, the second phase, selects an interpretation of the input sentence consistent with the current context and the reader's goals. The integration process is connectionist in nature and uses a spreading activation mechanism. The most highly activated nodes in the network represent the reader's interpretation.

2.2 Kintsch's (1988) Model of Word Problem Solving

The model of goal formation processes in the LICAI model is taken directly from Kintsch's (1988) version of Kintsch and Greeno's (1985) model of word arithmetic problem solving and related

work by Kintsch and associates (Cummins, Kintsch, Reusser, & Weimer, 1988; Delarosa, 1986; Fletcher, 1985). Kintsch has incorporated assumptions about representation and strategic processes originally developed by van Dijk and Kintsch (1983) into the construction–integration theory. These assumptions are central to Kintsch and Greeno's (1985) model of word arithmetic problem solving.

Van Dijk and Kintsch (1983) assumed that text comprehension generates multiple representations, the most important of which are the *text base* and the *situation model*. The text base is the semantic representation of a text describing a problem or instructions to perform a task. The situation model is a representation of the problem or task generated by the comprehension processes. The other key assumption made by van Dijk and Kintsch (1983) is that reading is a strategic process. Different kinds of text comprehension processes are involved in reading a narrative text. These *strategies* generate the inferences that are required to construct a situation model. The knowledge used by the strategies is represented as *comprehension schemata*.

In Kintsch and Greeno's (1985) model of word arithmetic problem solving, the arithmetic strategies take the text that describes the word problem and transform it into a situation model (i.e., *problem model*) representing the problem as sets of objects and their interrelationships. The problem model is operated on by appropriate problem solving mechanisms (arithmetic, algebra, or action planning) to generate a solution to the problem described in the text.

Kintsch (1988) incorporated the comprehension schemata needed to solve word problems into the construction–integration model by assuming that the arithmetic strategies operate during the construction phase. For example, an arithmetic schema is used to identify noun phrases as sets. Propositions are added to the network specifying alternative hypotheses about the role of each set in the problem (part-set, whole-set). Representations of all possible problem models consistent with the various roles are incorporated into the network. The alternative interpretations 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 text. The integration process then selects the most highly activated problem model. This problem model is an explicit arithmetic problem. When the problem model is solved by appropriate arithmetic procedures, the result is the answer to the word problem.

The critical difference between the Kintsch and Greeno (1985) and Kintsch (1988) versions of the arithmetic word problem models is that the control processes in the Kintsch and Greeno model generate a single problem model from the text. Kintsch (1988) argued that incorporating such "smart" rules is not a viable model of comprehension because such rules turn out to be too brittle. The Kintsch (1988) model of arithmetic word problem solving and the model of goal formation processes incorporated in the LICAI model generate multiple problem models.

2.3 Memory for Text

The LICAI model simulates a user who reads a short paragraph of task instructions and then attempts to perform the task by exploration. The model assumes that multiple goals, generated during reading the instructions, are stored in episodic memory along with the text base for the instructions. The goals are retrieved from memory by cues from displays generated by the application. The retrieved goals control the action planning process generating the exploratory behavior.

The LICAI model uses Kintsch and Welsch's (1991) model of memory for text. Their model processes each sentence during a single construction–integration cycle. After each cycle, the active