Knowledge Required for Understanding Task-Oriented Instructions

Muneo Kitajima, National Institute of Bioscience and Human-Technology
kitajima@nibh.go.jp

Peter G. Polson, Institute of Cognitive Science University of Colorado
ppolson@psych.colorado.edu

Abstract

*When they encounter problems with a novel or infrequently performed task, experienced users often complete their work by referring to manuals and trying task-oriented exploration. However, the contents of the documentation can vary from step-by-step instructions for a single task to generic descriptions of a family of related procedures. This paper describes the **LI**nked model of **C**omprehension-based **A**ction planning and **I**nstruction taking (LICAI) that simulates the processes involved in comprehending and using realistic task-oriented instructions. The simulations describe the necessary relationships among the content of the instructions, the goal, and the interface that enable users to accomplish their tasks.*

1 Introduction

This paper presents a theoretical analysis of the knowledge required to make effective use of documentation to accomplish novel tasks with a familiar application or any task using a new application. We assume an individual is an experienced user of the host operating system (e.g., Macintosh or Windows 95) and that he or she has the necessary background knowledge of the application domain.

The example task used in our analysis is 'Hide the Legend' of a data graph. The user is preparing a graph using an application like Cricket Graph III, Delta Graph Pro, or EXCEL. She has created the graph and is editing it to achieve specific design goals. During the editing process, she formulates the goal 'Hide the Legend.' She has never done this task before.

Rieman[7] conducted a diary study that investigated how people handle such situations in ordinary job settings. The participants in his study recorded their problems and how they dealt with them for one week. Rieman reported that the typical way to handle this situation was to get hints and then try to perform the task by exploration. The sources of hints were diverse: paper documentation, on-line help, or more experienced colleagues. Another option for users is to first attempt to perform the task by exploration. However, Rieman found that participants were less successful with this approach.

We selected the Hide Legend task as an example task because Rodriguez [8] found that both experienced and novice participants cannot perform this task using Cricket Graph III without getting additional information. Thus, this task is a good model for the problem solution episodes studied by Rieman [7].

The analysis presented in this paper assumes the following: a user has formulated a goal, "perform 'hide legend'", instructions on how to perform tasks are in the manual, and the user has located the description for the task in the documentation.

In an informal survey of the documentation for various graphics applications, we discovered that descriptions of the action sequence necessary to perform a task come in many forms. The goal of this paper is to carry out analyses of the processes involved and knowledge required to comprehend these different forms of instructions and perform the task.

The **LI**nked model of **C**omprehension-based **A**ction planning and **I**nstruction taking (LICAI[1]) [5] was used in these analyses. This model simulates the processes involved in comprehending instructions and generating the actions required to perform a task. The processes are implemented in Mathematica programs.

2 Different Forms of Instructions

Our analysis of instruction types is based on LICAI's assumptions of the requirements for successful action planning. The model assumes that the representation of a large format display can contain over a hundred screen objects (e.g., icons, menus, application objects like titles and legends, the contents of individual cells in a spreadsheet). The action is a description of one of the admissible actions on the interface (e.g., type, click, press and hold). A user must extract appropriate object–action descriptions from the instructional text and/or generate a correctly ordered sequence of these descriptions.

[1] When LICAI is pronounced [li kai], the pronunciation represents a two-kanji Japanese word, 理解, meaning 'comprehension.'

Successfully performing a novel task with the aid of instructions requires them to be translated into a representation of the correct action sequence. First, object descriptions must be recognized and translated into the specifications that uniquely identify corresponding screen objects. Second, action descriptions (e.g., turn off, choose) must be translated into possible actions on the interface.

These mappings are not trivial. For example, users must understand the vocabulary of the application domain (e.g., title, legend, x-axis label) and identify corresponding screen objects. Another complication is that a single sentence may contain multiple object–action pairs that must be properly sequenced.

We identified four cases that define the processes involved in following procedural instructions. They vary on two dimensions:

(1) the level of detail ranging from step-by-step instructions to texts that guide exploration without specifying the action sequence;
(2) the level of difficulty involved in selecting and sequencing multiple object–action representations.

Examples are shown in Figure 1.

Some instructional texts describe how to do a single task. Case I represents simple, detailed step-by-step instructions. Each step is described in a single sentence that specifies a screen object and one action on that object.

Case II represents complex, detailed step-by-step instructions. Multiple steps are described in a single sentence, and individual steps may refer to objects that will not appear on the screen until previous steps have been correctly executed. For example, 'Choose Show Graph Items… from the Options menu' is the second and third steps of the Case I instructions. Note that the first phrase, 'Choose Show Graph Items…,' specifies an action on an object that is not yet on the screen.

Case III is a text that gives step-by-step instructions for a generic procedure from a family of related tasks. The text describes how to show or hide graph components including the graph title, the legend, and so forth. The LICAI model generates a version of the Case I instructions from this text to specify the correct action sequence.

Case IV is a text that describes a related collection of tasks on changing the legend. However, the information given does not include complete step-by-step instructions for any task. These instructions support learning by exploration. Kitajima and Polson [4][5] have analyzed the conditions in which users can successfully perform a task by exploration.

The focus of this paper is to show how the LICAI model can extract the steps required to perform the Hide Legend task for Cases I, II, and III, and to discuss qualitative differences in their performance.

Case I

Step1.Select the graph.
Step2.Pull down the Options menu.
Step3.Select Show Graph Items… menu item.
Step4.Click the check box labeled 'Legend.'
Step5.Click on the button labeled 'OK.'

Case II

Step1.Select the graph.
Step2.Choose Show Graph Items from the Options menu.
Step3.Turn off the Legend check box.
Step4.Click OK.

Case III

Step1.Select a graph object.
Step2.Choose Show Graph Items from the Options menu.
Step3.Turn on check boxes for graph components you want to display, or turn off check boxes for components you want to hide.
Step4.Click OK to turn to the active graph window.

Case IV

Using Legends in Your Chart

A Legend is a combination of text and graphics used to associate additional data to the chart. Depending on the type of chart you have plotted, some charts may appear with or without a legend. Row and column labels in the Data page determine the legend labels. The "Legend…" command allows you to add to or remove chart legends. You can also reverse the direction of the data within the legend and also change the legend for any overlay charts.

Figure 1. Four versions of instructions for the Hide Legend task. Steps for Case III are adapted from *CA-Cricket Graph III User Guide for Macintosh*. Text for Case IV is adapted from *DELTAGRAPH USER'S GUIDE 4.0*.

3 The LICAI Model

The LICAI model was developed originally by Kitajima and Polson [5] to account for a variant of Case IV. In the following sections, we first describe the cognitive architecture on which the LICAI model is built, and then explains very briefly about the model. See Kitajima and Polson [4][5] for thorough description.

3.1 The Construction–Integration Architecture

LICAI is a comprehension-based model of instruction following and exploration. The cognitive processes specified in LICAI are implemented using the construction–integration (C-I) architecture developed by Kintsch [1][2], which has been applied successfully to model cognitive processes involved in text comprehension [1], word problem solving [1], and action planning [3][6].

In the *construction* phase, a C-I cycle generates a connectionist network that represents alternative meanings of a sentence or alternative actions that can be performed on a given step in a computer-based task, and the knowledge necessary to select among the alternatives. The *integration* phase uses spreading activation to implement a constraint satisfaction process that selects a contextually appropriate alternative consistent with the users' goals. The nodes in the network are propositions. Links in the network are established by common arguments of propositions; when two nodes share a common argument, they are connected. The constraint satisfaction process is controlled by the pattern of interconnections.

3.2 Comprehending Hints

Kitajima and Polson [5] describe in detail the processes in LICAI that simulate comprehension of hints like "Pull down the Options menu." This hint specifies actions to be performed on an object on the screen. LICAI transforms the propositional representation of the hint into a representation that controls the action planning process, a *do-it goal*. LICAI requires that a successful do-it goal should have direct links to the representation of the correct object on the screen. This means that a hint like "Pull down the Application menu" would not be as effective as the hint like "Pull down the Options menu" because the former requires users of additional knowledge to associate "the Application menu" with the icon that appears on the top-right corner of the Macintosh desktop.

3.2.1 Goal Formation. Representations of do-it goals determine the effectiveness of hints. This section describes in detail how LICAI generates the do-it goals from the instructions.

LICAI assumes that the user processes each instruction sentence-by-sentence. The instruction comprehension process takes a propositional representation of each sentence and generates goals of the form,

PERFORM [*ACTION*, *OBJECT*] or
PERFORM [$, *OBJECT*]

where $ is a place holder for an *ACTION* that will have to be inferred by the action planning process. (Note that variables are represented in capital italics, like *OBJECT*.)

LICAI assumes that the instruction comprehension process is analogous to Kintsch's [1] model for solving word problems, which describes reading as a strategic process [9] and assumes that strategies generate inferences required to guide problem solving. The knowledge used by the strategies is represented as comprehension schemata. The instruction comprehension process in LICAI takes a semantic representation of the instructions as input and elaborates this representation with inferences generated by comprehension schemata to construct goals.

LICAI incorporates three kinds of comprehension schemata. *Global instruction reading* schemata represent the top-level strategy used by a reader to process text. 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. *Goal formation* schemata transform DO propositions into propositions that represent goals.

The simulations described in this paper focus on the processes that operate after the instructional text has been transformed into a propositional representation, i.e. the text base, and the global instruction reading schema has been applied to transform all propositions of the form, *VERB* [*OBJECT*] into the form DO [YOU, *VERB*, *OBJECT*].

While comprehending step-by-step instructions, LICAI applies *do-it schemata* to generate do-it goals by elaborating propositions of the form DO [YOU, *VERB*, *OBJECT*] in the original instruction text. Do-it schemata elaborate propositions to perform a specific action on a screen object. When *VERB* in propositions of the form DO [YOU, *VERB*, *OBJECT*] is a DEVICE-ACTION like select, choose, pull down, click, drag, or release, the do-it schema elaborates such propositions into propositions representing do-it goals.

The do-it schema has the following form:

<u>*Do-It Schema I:*</u>

```
IF (DO [YOU, VERB, OBJECT] &
  ISA [VERB, DEVICE-ACTION]) ⟹
 PERFORM [VERB, OBJECT, a list of specifications]
```

For example, in Case I, the description for the second step 'Pull down the Options menu' is represented as a set of propositions:

```
DO [YOU, PULL DOWN, $]
ISA [$, MENU]
HAS-LABEL [$, Options]
```

$ is a place holder that indicates a screen object yet to be identified. The first proposition triggers application of Do-It Schema I, generating the following do-it goal:

```
PERFORM [PULL DOWN, $,
    ISA [$, MENU], HAS-LABEL [$, Options]]
```

Sometimes 'pull down' 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:

Do-It Schema II:

```
IF (HAS-LABEL [OBJECT, LABEL]) ⎝
  PERFORM [$, OBJECT,
    HAS-LABEL [OBJECT, LABEL],
    a list of specifications]
```

For example, Step 2 in Case I specifies the label of a screen object Options to be acted on. Thus, Do-It Schema II can be applied to generate the following do-it goal:

```
PERFORM [PULL DOWN, $,
    HAS-LABEL [$, Options], ISA [$, MENU]]
```

4 Simulation: Following Task-Oriented Instructions

We describe how the LICAI model simulates users who follow task-oriented instructions that are given in various forms aiming at supporting the hide legend task. We focus on the representation of do-it goals that LICAI generates and discuss their appropriateness for controlling LICAI's action planning process.

4.1 Simulation of Case I

In Case I, each sentence describes a single step, the interface action is described using the most common verb, and the target of the action is specified completely.

The do-it goal for Step 1 is generated by applying Do-It Schema I:

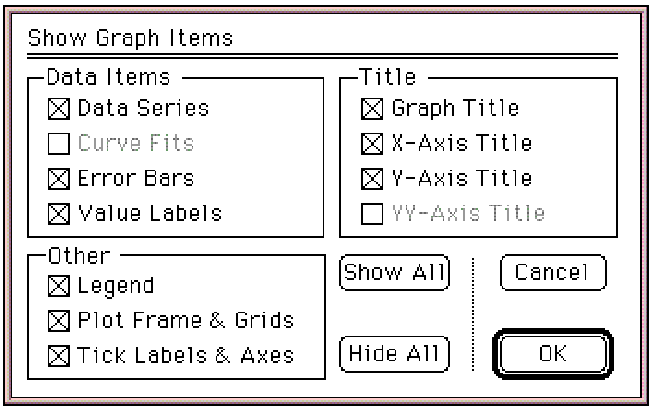
```
PERFORM [SELECT, $, ISA [$, GRAPH]]            Step 1
```

Steps 2 through 5 specify the labels of the screen objects to be acted on. Thus, Do-It Schema II can be applied to generate the following do-it goals for each step:

```
PERFORM [PULL DOWN, $,
    HAS-LABEL [$, Options], ISA [$, MENU]]     Step 2

PERFORM [CHOOSE, $,
    HAS-LABEL [$, Show Graph Items…],
    ISA [$, MENU-ITEM]]                        Step 3

PERFORM [CLICK, $,
    HAS-LABEL [$, Legend],
    ISA [$, CHECK-BOX]                         Step 4

PERFORM [CLICK, $,
    HAS-LABEL [$, OK], ISA [$, BUTTON]]        Step 5
```

In the action planning process, each do-it goal is performed on the following displays: the graph on the screen (Step 1), the Options menu item (Step 2), Show Graph Items… on the pulled down menu (Step 3), and the dialog box shown by Figure 2 (Steps 4 and 5). The representations of each do-it goal are linked to the correct screen object and action, and the action planning process will attend to the correct screen object and select the correct object–action pair. However, following instructions like 'Pull down the Options menu' (Step 2) involves significant inferences. For example, 'pull down' must be mapped onto the action 'press and hold' the mouse button on the required screen object. The screen object must be pointed at by the mouse cursor. This condition is satisfied by the action planning process.

Thus, following instructions in Case I is predicted by LICAI to be straightforward if users have the necessary background knowledge to generate these do-it goals, and if they correctly alternate sentence-by-sentence instruction reading and action planning.



Show Graph Items

Data Items
- ☒ Data Series
- ☐ Curve Fits
- ☒ Error Bars
- ☒ Value Labels

Title
- ☒ Graph Title
- ☒ X-Axis Title
- ☒ Y-Axis Title
- ☐ YY-Axis Title

Other
- ☒ Legend
- ☒ Plot Frame & Grids
- ☒ Tick Labels & Axes

Figure 2. The dialog box for 'Show Graph Items.'

4.2 Simulation of Case II

In Case II, steps 2 and 3 in Case I are combined, the two object–action pairs are out of order, and the verb 'pull down' is missing from the surface structure of the sentence. However, Case II can be reduced to Case I when appropriate task-domain schemata are available: Step 2 requires a schema that transforms 'choose *X* from *Y*-menu' into 'pull down *Y*-menu' followed by 'select *X*'; Step 3 requires a comprehension schema to equate 'turn off *Z* check box' with 'point and click the marked check box labeled by *Z*.' Rodriguez [8] showed that his participants had no difficulty in following Steps 2 and 3 of Case II suggesting the existence of the required knowledge.

4.3 Simulation of Case III

The key difference between the Case III text and the Cases I and II texts is that Case III is not written for a single task. This text describes a class of tasks that have the common effect on a class of application objects. In our particular example, the effects are 'hide' or 'show.' The objects to be effected are 'graph items.'

The instruction comprehension process requires additional knowledge about the graph domain to make the do-it schemata applicable. For example, the text for

Step 1, 'Select a graph object,' requires understanding what the phrase 'graph object' refers to. A do-it goal might have been generated by Do-It I without such an understanding. But this do-it goal would not be useful because the necessary link to the correct screen object does not exist.

The *CA-Cricket Graph III Users Guide* for Macintosh lists possible graph objects or graph components:

1) graph-title,
2) graph-frame,
3) plot-frame,
4) legend,
5) plot-element,
6) horizontal-tick-marks,
7) horizontal-axis-tick-labels,
8) horizontal-axis,
9) vertical-tick-marks,
10) vertical-axis-tick-labels,
11) vertical-axis-title,
12) vertical-axis.

We assume that this knowledge is stored in long-term memory and is retrieved when Step 1 of Case III is elaborated. The same knowledge is required to comprehend the term 'graph components' in Step 3. This elaboration process is controlled by a probabilistic memory retrieval process [1][3].

We describe a simulation of the instruction comprehension process of Case III by showing how the task-domain knowledge shown above is incorporated into the network in the construction phase and how the retrieved knowledge is used to generate do-it goals. The propositional representation of Step 1 of Case III is as follows:

DO [YOU, SELECT, $],
ISA [$, GRAPH-OBJECT].

The probabilistic memory retrieval process uses links between the text propositions and proposition stored in long-term memory. The argument GRAPH-OBJECT links to

MADE-UP-OF[GRAPH-OBJECT,GRAPH-COMPONENTS],

which in turn links to each of the 12 specific descriptions of the graph components, for example,

ISA [GRAPH-TITLE, GRAPH-COMPONENTS],
…
ISA [LEGEND, GRAPH-COMPONENTS],
…

The text base elaborated by retrieving relevant knowledge can include the inferences shown above. This elaborated text base is then augmented by Do-It Schema I. LICAI can generate up to 12 do-it goals, one for each graph component. Each proposition describing a component retrieved from long-term memory combined with DO [YOU, SELECT, $] through the link ISA [$, GRAPH-OBJECT] via MADE-UP-OF [GRAPH-OBJECT, GRAPH-COMPONENTS] to generate a do-it goal specifying selection (single clicking) of that graph component. Thus, the model could generate from 1 to 12 do-it goals.

However, the critical inference for the Hide Legend task, ISA [LEGEND, GRAPH-COMPONENTS], is more likely to be retrieved from long-term memory because of its link to the task goal, PERFORM [HIDE, LEGEND]. Thus, it is likely that the correct do-it goal given below will be generated during the instruction comprehension process:

```
PERFORM [SELECT, $,
    ISA [$, GRAPH-OBJECT],
    MADE-UP-OF [GRAPH-OBJECT,
         GRAPH-COMPONENTS],
    ISA [LEGEND GRAPH-COMPONENTS]]
```

When multiple do-it goals are generated, LICAI must the goal-selection process that selects one of these as the goal for action planning process when the initial task display appears. In this process, the task goal, PERFORM [HIDE, LEGEND], also plays a major role, providing additional links to the do-it goal.

5 Discussion

LICAI enables us to define a continuum of texts describing procedures ranging from simple step-by-step instructions (Case I) to instructions that support exploration (Case IV). In each case, LICAI describes constraints on the vocabulary used in the text, the details of the interface, and necessary background knowledge that must be satisfied to support successful execution of a task.

Case I instructions are robust if the text satisfies some obvious constraints. Readers must have the background knowledge necessary to transform action descriptions (e.g., pull down, choose) into possible actions on the interface (e.g., click, press and hold). It is reasonable to assume that this is the kind of knowledge acquired when the user is learning basic interaction skills. Readers must also be able to identify the screen objects described in the text. They must know the vocabulary used to describe various categories of screen objects like menu, menu items, and the like as well as understand the vocabulary of the application domain (e.g., title, legend, x-axis label). Labels on screen objects and information on the identity of objects retrieved form the necessary links between the text representation and the representations of objects and actions.

However, Case I makes strong assumptions about the coordination of the reading and action planning processes. The user is assumed to read one sentence, perform the described action on the specified object, and then continue reading the next sentence. In addition, a user could have trouble if they read several sentences and then try to perform the task. The sequence of do-it goals would have to be retrieved in the correct order and remembered correctly.

Case II instructions are also robust if readers have the necessary background knowledge, although they must understand a wider range of vocabulary describing actions and objects. However, the major complication is dealing with sentences that describe multiple steps. Specialized task-domain schema must infer missing elements in the text and the multiple do-it goals must be stored in working memory and retrieved in the correct order by the application displays.

Case III is obviously the most interesting from both the point of view of the model as well as usability considerations.

The instructions for Case III have a phrase in Step 1, ‘graph object,’ and a phrase in Step 3, ‘graph components,’ that effectively function as variables that must be bound to the correct object, the legend. The bottom-up nature of the construction process causes LICAI to consider multiple bindings for these variables. The model selects the do-it goal with correct binding because of the presence of the task-goal PERFORM [HIDE, LEGEND].

Kitajima and Polson [4][5] did an analysis of Case IV. They found that the user’s task goal must link to the screen object that must be acted on to successfully perform the task. Thus, if the interface had a Hide menu that presented a Legend menu item, step-by-step instructions would be unnecessary. The links to the screen object labels ‘Hide’ and ‘Legend’ would enable action planning process to perform the task because of the links to the task goal PERFORM [HIDE, LEGEND].

6 Conclusions

We performed analyses of three versions of instructions for the Hide Legend task using the LICAI model. In each case, we described the relationship that must hold between the user’s representations of the instructions, the display, and the action required to perform the task. Step-by-step instructions generate a sequence of do-it goals, each of which described an object on the screen and action. The most basic requirement is that the user must understand the vocabulary used to describe the object and actions in order to map them onto interface actions and descriptions of objects on the screen.

A basic prediction of the LICAI model is that Case III instructions require specialized task-domain knowledge to bind generic terms to specific items that are related to the current task. This causes the instruction comprehension process to generate multiple interpretations of the instruction text and that the task goal is critical in selecting the correct do-it goal.

7 References

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

[2] Kintsch, W. (1998). Comprehension: *A Paradigm for Cognition*. Cambridge: Cambridge University Press.

[3] Kitajima, M., and 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.

[4] Kitajima, M., and 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.

[5] Kitajima, M., and Polson, P. G. (1997). A comprehension-based model of exploration. *Human-Computer Interaction*, **12**, 345-389.

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

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

[8] Rodriguez, M. (1997). A detailed analysis of exploratory behavior of new users of a graphics application. Technical Report in preparation. Institute of Cognitive Science. University of Colorado, Boulder.

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