A Quantitative Model of the Learning and Performance of Text Editing Knowledge

Peter G. Polson
University of Colorado
Boulder, CO 80309

David E. Kieras
University of Michigan
Ann Arbor, MI 48109

Abstract

A model of manuscript editing, implemented as a simulation program, is described in this paper. The model provides an excellent, quantitative description of learning, transfer, and performance data from two experiments on text editing methods. Implications of the underlying theory for the design process are briefly discussed.

Introduction

This paper describes a model of a user's representation of the knowledge necessary to perform manuscript editing. We derive and evaluate quantitative predictions from a simulation model based on a theory of human-computer interaction (Kieras and Polson, in press) and on Card, Moran, and Newell's (1980, 1983) analysis of manuscript editing tasks. We show that the model can predict learning, transfer, and execution times from experiments manipulating training orders and practice.

A Quantitative Theory

The Kieras and Polson (in press) theory represents a system or device as a generalized transition network (Kieras and Polson,1983) and user's knowledge of operating procedures as a production system. Their top level goal is to develop a theory that can make quantitative predictions of ease of learning, transfer, execution times, and ease of use.

This goal is achieved by formally describing the knowledge required to operate a device. Cognitive complexity is defined as the content, structure, and amount of the knowledge required to operate a device. The characteristics of this knowledge determine ease of learning and ease of use for the system. Informally, user friendliness is simply the converse of cognitive complexity.

Kieras and Polson (in press) assume that the knowledge can be partitioned into job situation and how-to-do-it knowledge. Job situation knowledge is a user's knowledge of tasks the system performs, the contexts in which tasks are performed, and how tasks are interrelated. How-to-do-it knowledge is knowledge of the actual operating procedures for a system and of methods used to perform tasks.

A user's job situation knowledge and how-to-do-it knowledge can be characterized in terms of the GOMS model developed by Card and Moran and Newell (1980, 1983). The GOMS model assumes that knowledge is partitioned into the following components: goals, operations, methods, and selection rules.

Mapping From GOMS to Productions

Job situation knowledge is described by job goals and job selection rules. Job goals represent the user's knowledge of the task that a system can perform. Job selection rules specify the circumstances under which these various tasks should be performed. A user's how-to-do-it knowledge is a collection of methods. Associated with each method is one or more selection rules defining the context of goals and particular constraints that make that method appropriate for accomplishing a specific goal. Operations are the mental operations and user actions necessary to accomplish a method. Job goals and job selection rules and the four components of a user's how-to-do-it knowledge, goals, methods, selection rules and operations, are represented in the production system formalism described in the next two sections.

Production Systems

Production systems are one of the more important formalisms that have been used to describe various cognitive processes including problem solving (e.g., Newell and Simon, 1972; Karat, 1982), text comprehension (Kieras, 1982), learning, and other cognitive processes (Anderson, 1982, 1983). The particular model described in this paper is the most elementary form of production system in that there are no conflict resolution rules, and very simple kinds of pattern matching are used in evaluating conditions.

From Human Factors in Computing Systems Proceedings. Edited by Lorraine Borman and Bill Curtis, April 14-18, 1985, San Francisco, Special Issue of the Sigchi Bulletin.

A production system is made up of two components: a collection of production rules and a working memory. The working memory contains goals and representations of inputs from the environment and other information. The productions are a collection of condition-action pairs of the form:

IF (condition) THEN (action).

The conditions specify a series of patterns that are matched against the current contents of working memory and the environment. Each pattern defines goals, particular pieces of information in working memory, or information on a CRT screen or in a manuscript. If a pattern matches, its associated production fires, and the action of the production is executed. The actions manipulate information contained in working memory or carry out operations that change the external environment.

A model of a user's performance of a task or a method is a program written as a collection of productions that when executed generates the correct sequence of user actions and simulates the hypothesized set of cognitive operations required to carry out the task. The program executes by having the system alternate back and forth through recognize and act modes. The conditions of all productions are matched against the contents of working memory and the environment during the recognize mode. The system then transitions to the act mode in which the action components of all productions whose conditions match, fire and are executed. Actions carried out modify the contents of working memory, and possibly the external environment, enabling a new production or set of productions to fire.

Description of Notation

The notation used in our simulation is shown in Figure 1. Labels, identifiers, and function names may be of arbitrary lengths and contain hyphens. The parentheses shown in Figure 1 are part of the notation. Each production has a label which is not functional but can be used for tracing for execution and debugging. The list of conditions is prefaced by an IF and an AND. The AND specifies that the production will fire if all of the conditions in the list are true. Each of the conditions is a test for the presence of a pattern specifying information in working memory or in the environment. Individual elements of a pattern are separated by one or more spaces and may be either constants or variables; variables are denoted by the prefixed *. The symbol ??? is a place holder and it will always match a corresponding constant without binding the constant to any variable. Each type of test shown in Figure 1 tests for different kinds of information. TEST-GOAL and TEST-NOTE test for the presence of patterns in working memory that represent goals and notes respectively. The condition NOT is true if the specified condition is false.

A GENERIC PRODUCTION

```
(label
    (IF (AND (TEST-GOAL description of goal)
             (TEST-NOTE description of note)
             (TEST-SCREEN description information on screen)
             (TEST-MSS description of editing task)
             (NOT (condition)) )
    THEN
        ((ADD-GOAL description of goal)
         (DELETE-GOAL description of goal)
         (ADD-NOTE description of note)
         (DELETE-NOTE description of note)
         (DO-KEYSTROKE depress single key or control)
         (DO-TYPE-IN enter string stored in working memory)
         (LOOK-MSS find string in manuscript))  )
```

Figure 1. A generic production showing the possible conditions and actions.

The action is a sequence of operations that modifies the contents of working memory and generates simulated user actions that are passed off to the device simulation. These user actions cause the device simulation to generate the appropriate responses which can in turn change the CRT screen or other information that the production system attends to. Modifications of working memory by adding and deleting goals and notes enable new productions to fire during the next cycle. ADD-GOAL, ADD-NOTE, DELETE-GOAL, and DELETE-NOTE add or delete their respective patterns from working memory. DO-KEYSTROKE simulates the activation of a single function key and this information is passed to the device simulation. LOOK-MSS causes the simulation to scan its representation of the marked-up manuscript seeking a matching string.

The primary limitation of the theory is that no attempt is made to represent more fundamental cognitive processes like memory retrieval or reading comprehension. Kieras and Polson (in press) assume that they could formalize a theory of human-computer interaction without dealing with more basic cognitive processes which would dramatically increase the complexity of the simulation while providing little additional information about interactions between human and system. They argue that more basic cognitive processes have been explored in detail and are in principle well understood.

The Model and its Evaluation

The simulation program, a production system, for the manuscript editing task was constructed from the following components. The core is the

top level control structure that causes the model to search for the next edit to be performed and that halts the process when all edits have been completed. The cursor positioning routines describe the system's knowledge for manipulating the arrow keys. The select range component specifies the range for each edit using a single character find function described later. Finally, there are separate segments for each text editing method: INSERT, GENERAL DELETE, three specialized deletion methods for characters, words, and sentences, COPY, MOVE, and TRANSPOSE. The simulation system, implemented on a Xerox 1108 in INTERLISP, executes the same series of edits carried out by subjects given descriptions of each editing task. Predictions are derived from static characteristics of various components of the simulation program and the program's execution time behavior.

Simplifying assumptions are made in order to derive strong predictions from the theory. First, all productions are homogeneous in that they take an equal amount of time to learn or execute. This assumption is not necessary or even well motivated in many circumstances but it dramatically simplifies evaluation of a model. Second, linear relationships are assumed between parameters of the simulation model and learning and performance measures.

Given the above assumptions, and the production system architecture described previously, the following performance predictions can be derived from the theory. Execution time (productivity) is a linear function of the number of recognize-act cycles executed during the simulated execution of an editing task. Learning and transfer assumptions are closely interrelated. It is assumed that the time to learn a procedure is a function of the number of "new" productions in its description. The first method learned by a subject involves all new productions and thus time to learn is a function of the number of productions in its description.

A common elements theory of transfer is assumed. If a similar production is learned during the acquisition of another method, it is assumed that this production will be incorporated into the representation of a new method at no cost in time to learn. Thus, time to learn a given method is a function of the number of "new" productions, that is the number of productions unique to the new method.

Learning and transfer predictions were evaluated by showing that the mean time to learn a given method can be predicted by the number of new productions when we systematically manipulated training order. Performance predictions were evaluated using data from a long-term practice study.

Experimental Evidence

The theory's predictions were evaluated with data from two experiments. The first study manipulated training orders of the editing methods (INSERT, DELETE, COPY, MOVE, and TRANSPOSE) and instructions describing methods for DELETE. A second study, using a limited number of subjects, employed the same training procedure, and then subjects edit manuscripts for eight additional sessions.

Method

The experimental environment was two CRTs driven by Fortran programs that communicated with each other. The first program implemented a simplified screen editor. Cursor positioning was done using cursor keys; there was no find function. After the cursor had been positioned at the starting location of an edit, the operation to be performed was indicated by depressing a function key.

DELETE, MOVE, COPY, and TRANSPOSE require that the range of the operation be specified. During a select range operation, a single character entered into the system causes cursor to move to the matching character in the text and all intervening text to be highlighted. The subject can press an arbitrary sequence of character keys and cursor keys to specify a range. Range selection is terminated by pressing the ENTER key. MOVE and COPY use the cursor keys to specify the target location. TRANSPOSE involves the specification of two ranges using the method described above. The cursor keys are used to position from the end of the first range to the beginning of the second. After completing an edit, the subject can undo the edit by pressing the REJECT key or indicate that the edit is correct by pressing the ACCEPT key.

The second part of the environment was a computer assisted instruction package (CAI) which implemented the training procedure, presented all instructional material, and provided feedback on errors. On depression of the ACCEPT key, the editor transmitted a description of the resulting edit to the CAI package where it was evaluated. On errors, appropriate feedback was provided by the CAI package which had the logic necessary to evaluate what kind of error had been made on a given edit, e.g., typing in the incorrect text for an insert.

The experiment involved two phases. First, subjects read instructions on the structure of the editing task and how to use the cursor positioning keys. They then learned five text editing methods. For each method, subjects were given detailed instructions on how to perform the method including information about how to select the range. Subjects then practiced the method ny editing a two page manuscript with four edits on each page. If a subject made an error, appropriate feedback was given immediately after depression of the ACCEPT key, and the subject then reviewed the instructions for the method. Subjects were required to redo all edits if they made at least one error. The learning criterion was one error free repetition of all 8 edits. Subjects immediately went on to the instructions for the next method after having successfully completed the practice task for the current method.

There were six experimental conditions with 15 subjects in each condition. Subjects were recruited from the community via a newspaper ad

and were paid $15 for participation in the experiment. The experimental conditions were defined by the factorial combination of delete instructions and training orders. INSERT was learned first by subjects in all 3 training orders. The orders for the remaining four methods were 1) DELETE, COPY-MOVE, TRANSPOSE, 2) TRANSPOSE, DELETE, COPY-MOVE, and 3) COPY-MOVE, TRANSPOSE, DELETE. MOVE always followed COPY.

The instruction manipulation involved giving subjects different descriptions of DELETE. GENERAL DELETE described the select range process in terms of the use of the cursor keys and the single character find. This was the same description of the select range process included with the instructions for COPY, MOVE, AND TRANSPOSE. SPECIFIC DELETE was described as a set of methods for deleting characters, words followed by a space, and sentences. Thus, deleting a word was described as positioning the cursor at the beginning of the word, hitting the space, and then the enter key.

Results and Discussion

The data from the first experiment were used to evaluate predictions concerning learning and transfer. The average times to reach criterion on each block of practice edits was calculated; this time included the reading times for feedback screens and review of instructions. The 30 means are the observed values in Figure 2.

The number of "new" productions was calculated in the following fashion. Subjects learned the overall structure of the editing task, the core, and methods for using the cursor keys during the acquisition of INSERT, the first method learned by all six groups. Subjects acquired the general select range procedure during their first exposure to GENERAL DELETE, COPY, or TRANSPOSE. It was assumed that productions describing the core, the methods for using the cursor keys, and the general select range procedures transferred perfectly to any method for which they were appropriate during the latter parts of training. Finally, there were common parts to all methods that were learned during acquisition of INSERT. COPY and MOVE were almost identical except for four unique productions. TRANSPOSE shared important components including the general select range with GENERAL DELETE, COPY, and MOVE.

The predicted values in Figure 2 were calculated by entering the number of new productions for each mean as a predictor. The number of new productions accounted for 83% of the variance with a slope of .54 min. per production and an intercept of 9.02 minutes. Polson and Kieras (1983) and Kieras and Bovair (1983) have found very similar results. In particular, all of these studies give approximately the same figure for the amount of time necessary to learn a new production, 30 seconds.

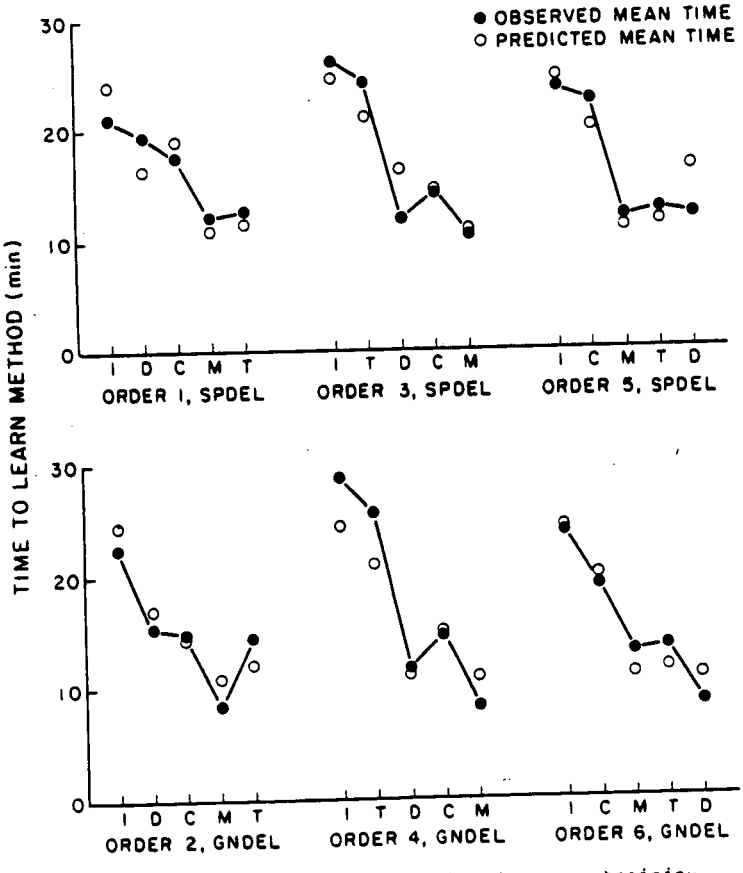

Figure 2. Observed and predicted mean training times for each method and training condition.

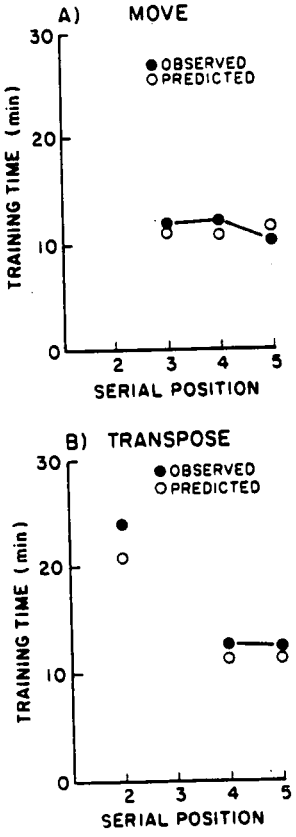

Figure 3. Observed and predicted mean training times for move(A) and transpose(B) as a function of serial position in training order.

The theory also makes predictions about the difficulty of a method as a function of serial position in a training order. Because MOVE always follows COPY, the theory predicts that the training time for MOVE will be constant across serial position because the additional training on productions common to COPY, MOVE, and other editing methods as no effect. The observed and predicted mean training times for MOVE averaged across DELETE instructions plotted as a function of serial position are shown in Figure 3A. The theory predicts that the serial position function for TRANSPOSE will be a step function with equal training times for serial positions 4 and 5, and these results are shown in Figure 3B. Examination of Figures 2, 3A, and 3B shows excellent correspondence between theory and data.

We conclude that these results provided strong support for the learning and transfer assumptions of the theory. The number of new productions is a good predictor of training times in very different experimental situations; see the previously referred to papers. We do not have a detailed process explanation of the constancy of the learning parameter. However, such results suggest that with respect to the learning process our homogeneity assumption is an excellent first order approximation.

Performance Predictions

The second experiment involved 8 subjects who were paid for their participation in the experiment. Subjects were first trained using the methods described above. On the next day, they edited a single manuscript containing 70 edits. There were 10 edits on each page, 2 of each kind. The correctness of each edit was checked by the CAI package after depression of the accept key. Subjects were required to do incorrect edits over again. The mean time to perform each correct edit was calculated from the lapse time between depression of the last cursor control key that positioned the cursor at the beginning of the range to the depression of the ACCEPT key. These times were then averaged over all subjects using similar select range methods. The above procedure was repeated for an additional seven days with different manuscripts, but the long term practice effects will not be discussed here.

The data from INSERT was excluded because it is dominated by typing time and therefore provides little insight into the processes assumed by the theory. One subject used the cursor keys to specify the range; this individual was dropped from the analysis. The remaining seven subjects selected the range of an operation using the end-character-method or a simple variant of it on approximately 80 percent of the edits. The end-character-method highlights the range by entering the last character in the range, repeatedly if necessary. The data from these subjects were used to test the performance predictions of the theory.

Predicted keystroke sequences and other statistics used as predictors were generated by the simulation. The predictors were calculated from simulation runs in which the simulation program edited the same text used in the experiment. The predictors included totals from the last cursor control keystroke before the function key stroke to depression of the ACCEPT key for the number of recognize-act cycles (CYCLES), the number of times the simulation looked at the manuscript, the number of keystrokes, and various measures of working memory load including the number of goals and notes entered into working memory (WM-IN).

The mean editing times for the last 48 edits were fit using multiple regression techniques with CYCLES and WM-IN as predictors. The first eight edits were dropped from the analysis because they were highly variable due to lack of practice. Dropping them lead to a large improvement in the percentage of variance explained with little or no change in the observed means or values of the regression coefficients. The observed and predicted times as a function of editing method averaged over the 48 edits are shown in Figure 4.

The coefficient for CYCLES is .17 sec. and .56 sec for WM-IN; the intercept is not significantly different from 0. The percentage of variance explained is .72. CYCLES was selected as a predictor on the basis of assumptions discussed in a preceding paragraph. Several different working memory statistics were evaluated as potential predictors including peak and average working memory load, but WM-IN is the superior predictor suggesting that entry of information into working memory rather than maintenance is a resource and time consuming operation. Number of predicted keystrokes did not improve the fit of the model because this predictor is very highly correlated with CYCLES.

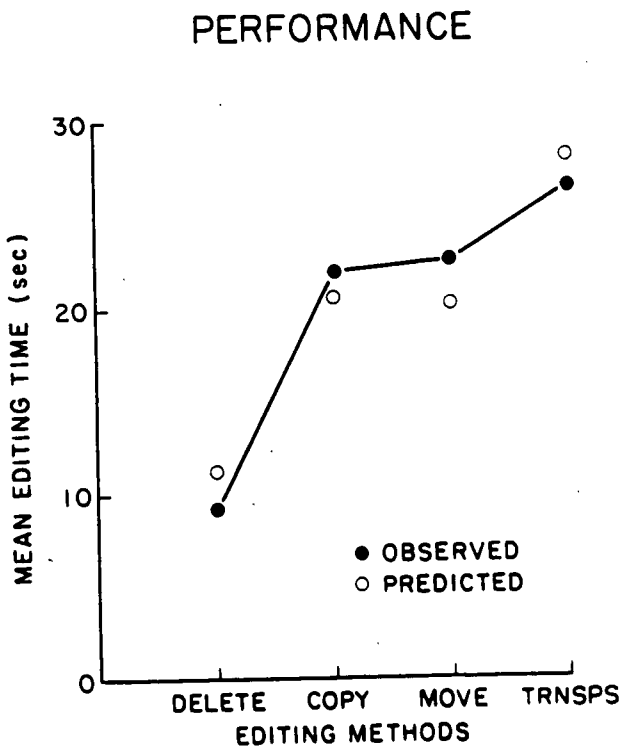

Figure 4. Observed and predicted mean execution times as a function of editing method.

Work in progress is attempting to model practice effects under the assumption that improvements in performance can be explained in terms of modifications of the production system as a function of practice. We assume that productions that fire in an invariant order are combined into a single production, the process of composition (Anderson, 1983), and that unnecessary checks of prompts, messages, and entries are eliminated from expert methods leading to methods that take fewer CYCLES and thus less time.

Summary

What we have been able to show is that the Kieras and Polson (in press) formalism can provide an adequate quantitative account of learning, transfer, and performance for a manuscript editing task. Polson and Kieras (1983) and Kieras and Bovair (1983) have shown that the learning predictions of the theory are also supported in very different contexts. We take these results to provide strong support for the basic processing and representational assumptions incorporated in the theory.

The theory has important applied as well as research implications. It can be used as a tool, along with fast prototyping methods, to evaluate the user interface for various software products at all stage of the design cycle. Methods that are inefficient in terms of execution time, difficult to learn because of large numbers of productions, or because inconsistency across methods reduces transfer can be detected using the simulation methodology and corrected at the design stage. Thus the theory, can be used as a design tool to quantify the effects of design changes and to compare alternative designs.

The theory can also be used to more precisely define design tradeoffs. There is a large amount of informal discussion in the literature as well as over various networks concerning the merits of various kinds of editors. These discussions are dominated by testimonials and personal opinion rather than any sort of scientific analysis and deal with discussions of EMACS versus other editor designs.

The difference between EMACS and other screen editors can be precisely specified using the production system formalism. EMACS has a large number of simple methods, typically a single keystroke, and thus methods are represented as single productions. However, EMACS has a large number of selection rules, at least one for each method. Thus the tradeoff is between the complexity of methods versus the number of selection rules.

In conclusion, we think it is necessary that the study of various issues in human-computer interaction be conducted at a far more abstract level than they typically are today and that theories be developed that make quantitative predictions for various parameters of user performance. Such theories are necessary for the development of a principled, design technology for user interfaces.

References

Anderson, J.R. Acquisition of cognitive skill. Psychological Review, 1982, 89, 369-406.

Anderson, J.R. The Architecture of Cognition. Cambridge, MS: Harvard University Press, 1983.

Card, S.K., Moran, T.P. & Newell, A. Computer text editing: An information-processing analysis of a routine cognitive skill. Cognitive Psychology, 1980, 12, 32-74.

Card, S.K., Moran, T.P. & Newell, A. The psychology of human-computer interaction. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1983.

Karat, J. A model of Problem Solving with Incomplete Constraint Knowledge. Cognitive Psychology, 1982, 14, 538-559.

Kieras, D.E. A model of reader strategy for abstracting main ideas from simple technical prose. Tex, 1982, 2, 47-82.

Kieras, D.E. & Polson, P.G. A generalized transition network representation of interactive systems. Proceedings of the CHI 1983 Conference on Human Factors in Computing. Boston, December, 1983.

Kieras, D.E., & Polson, P.G. An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies, in Press.

Kieras, D.E. & Bovair, S. The acquisition of procedures from text. Technical Report, Department of Psychology, University of Arizona, in preparation.

Newell, A., & Simon, H.A. Human problem solving. Englewood Cliffs, New Jersey: Prentice-Hall, 1972.

Polson, P.G. & Kieras, D.E. A formal description of users' knowledge of how to operate a device and user complexity. Behavior Research Methods, Instruments, & Computers, 1984, 16, 249-255

Acknowledgements

This paper is the product of a collaborative project conducted by the authors at the University of Michigan and the University of Colorado with support from the International Business Machines Corporation. The opinions and conclusions contained within this research are those of the authors and not necessarily those of IBM. Requests for reprints should be sent to Peter G. Polson, Department of Psychology, Campus Box 345, University of Colorado, Boulder, CO 80309.

The Association for Computing Machinery, Inc.
11 West 42nd Street
New York, NY 10036

ISBN 0-89791-149-0

Sample Citation Information
Greene, Graham. The Heart of the Matter.
In *Proc. CHI '85 Human Factors in Computing Systems*
(San Francisco, April 14-18, 1985), ACM, New York, pp. 1-10.

Additional copies may be ordered prepaid from:

ACM Order Department
P.O. Box 64145
Baltimore, MD 21264

Price:
Members: $15
All others: $20

ACM Order Number: 608850

Orders may be charged 24 hours a day, seven days a week, by calling toll free nationwide 1-800-526-0359 Ext. 75; in New Jersey 1-800-932-0878 Ext. 75.