A Task-Analytic Approach to the Automated Design of Graphic Presentations

STEPHEN M. CASNER
University of Pittsburgh

BOZ is an automated graphic design and presentation tool that designs graphics based on an analysis of the task for which a graphic is intended to support. When designing a graphic, BOZ aims to optimize two ways in which graphics help expedite human performance of information-processing tasks: (1) allowing users to substitute simple perceptual inferences in place of more demanding logical inferences, and (2) streamlining users' search for needed information. BOZ analyzes a logical description of a task to be performed by a human user and designs a provably equivalent perceptual task by substituting perceptual inferences in place of logical inferences in the task description. BOZ then designs and renders an accompanying graphic that encodes and structures data such that performance of each perceptual inference is supported and visual search is minimized. BOZ produces a graphic along with a perceptual procedure describing how to use the graphic to complete the task. A key feature of BOZ's approach is that it is able to design different presentations of the same information customized to the requirements of different tasks. BOZ is used to design graphic presentations of airline schedule information to support five different airline reservation tasks. Reaction time studies done with real users for one task and graphic show that the BOZ-designed graphic significantly reduces users' performance time to the task. Regression analyses link the observed efficiency savings to BOZ's two key design principles: perceptual inference substitutions and pruning of visual search.

Categories and Subject Descriptors: D.2.2 **[Software Engineering]**: Tools and Techniques—*user interfaces*; H.1.2 **[Models and Principles]**: User/Machine Systems—*human information processing*; I.2.1 **[Artificial Intelligence]**: Applications and Expert Systems; I.3.6 **[Computer Graphics]**: Methodology and Techniques—*ergonomics*

General Terms: Algorithms, Design, Human Factors, Theory

Additional Key Words and Phrases: Automated design, graphic design, graphic user interface, task analysis, visual languages

1. THE COGNITIVE UTILITY OF GRAPHIC PRESENTATIONS

Studies of people using graphics for the purpose of solving problems or performing information-processing tasks find little evidence to support the claim that graphics are superior to other types of presentations such as

This work was supported by the Office of Naval Research, University Research Initiative, under Contract N00014-86-K-0678, and is based on parts of the author's Ph.D. dissertation, Intelligent Systems Program, University of Pittsburgh.
Author's address: S. M. Casner, NASA Ames Research Center, Mail Stop 262-4, Moffetfield, Calif., 94035-1000. e-mail: Internet: casner@eos.arc.nasa.gov

tables, lists, outlines, or text [19]. This is in striking contrast to the popular belief that graphics are the preferred medium for information presentation. These studies do not reject our intuitions about the usefulness of graphics but rather find that the utility of any information presentation is a function of the *task* that the presentation is being used to support. Graphics appear to succeed in practice when they have been designed to directly support a specific task, the success arising out of a judicious combination of task to be performed and particular graphic used. Generalizations made about the observed usefulness of a graphic for one task are highly inappropriate since using the same graphic for different tasks often causes the usefulness of the graphic to disappear. These results further suggest that graphic design principles that do not take into account the nature of the task to be supported (e.g., "line graphs are best for continuous data") are too underspecified to be useful in general. That is, empirical studies have shown that line graphs are supportive of some tasks that manipulate continuous data and are detrimental to the performance of others. The implication is that effective graphic design should begin with the task that a graphic is intended to support and be focused on finding those parts of the task, if any, that might be performed more efficiently within the context of a graphic.

Psychological studies concerned with how and why graphics are useful demonstrate two task-specific ways in which graphics can help expedite human performance of information-processing tasks [24, 34].

(1) *Computation.* Graphics sometimes allow users to substitute quick perceptual inferences in place of more difficult logical inferences. Perceptual inferences such as distance and size determinations, spatial coincidence judgments, and color comparisons, allow users to obtain the same information as more demanding logical inferences such as mental arithmetic or numerical comparisons.

(2) *Search.* Graphics sometimes reduce users' search for needed information by grouping related information in a single spatial locality, and by employing encoding techniques such as color, shading, and spatial arrangement that support preattentive and sometimes parallel visual search.

The following examples present combinations of tasks and graphics to illustrate the task-specific advantages that graphics can offer. The examples demonstrate two important consequences of the task-dependent usefulness of graphics. First, *different presentations of the same information best support different tasks.* That is, there is no such thing as the "most effective" way to display a data set. Graphic designs fall in and out of usefulness depending on the task a user wishes to perform using a data set. Consequently, there seems to be no way of designing an effective presentation without first considering the task for which the presentation will be used. Second, *what makes a presentation interesting are the efficient perceptual procedures that users can perform using the presentation to quickly arrive at a desired result.* In the examples below, the graphic designed for each task greatly simplifies the task by reducing the amount of cognitive work (computation and search)

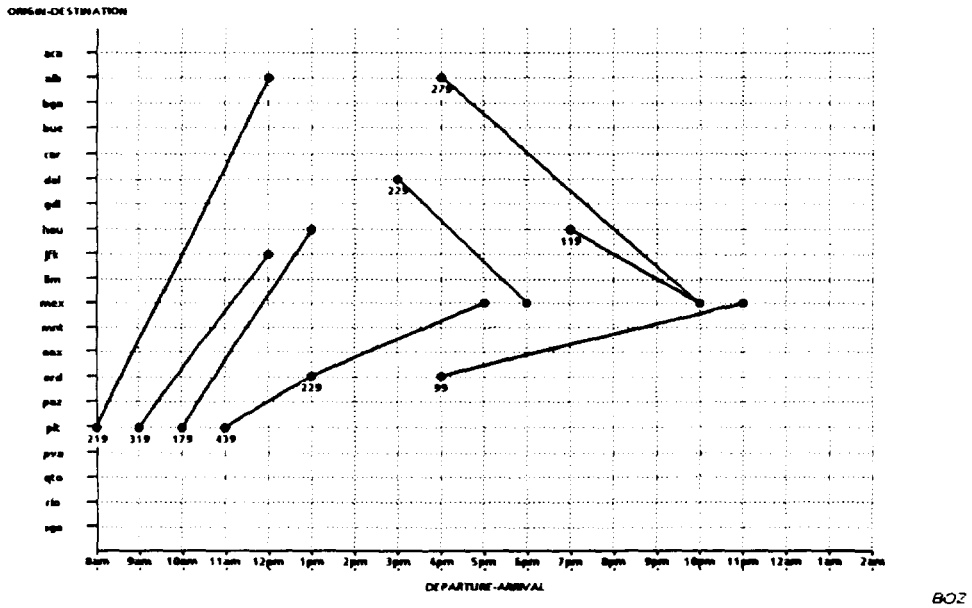


Fig. 1. Graphic for specific layover task.

the user must perform. To fully appreciate the task-specific advantages of each graphic, try performing each different task with each different graphic. The examples all pertain to the problem of using airline schedule information to locate flights obeying certain criteria. Each different task imposes different demands on the user, requiring the user to approach the same set of information in different ways. The graphic presentation for each task was designed by BOZ, the automated tool described throughout the paper.

(1) *Schedule Specific Layover.* Suppose you must travel from Pittsburgh to Mexico City but have to keep an afternoon business appointment in Albuquerque. Using the graphic in Figure 1 you can quickly locate your appointment time along the horizontal axis and locate Albuquerque (ALB) along the vertical axis. Running your fingers up and across the graphic to find the intersection of these two positions, you can now look for two flights that fall the closest to the right and to the left of the point of intersection. Note that this graphic allows you to immediately access exactly those flights that optimally satisfy your city and time constraints, reducing your search to a minimum. This graphic also lets you quickly figure your total "downtime" in the airport by simply judging the horizontal distance between the inside ends of each flight, and to figure your total travel time by judging the distance between the outer ends of each flight.

(2) *Find Cheapest Flight.* Suppose you wanted to find the cheapest flight going from Chicago to Mexico City. Using the graphic in Figure 2 you can

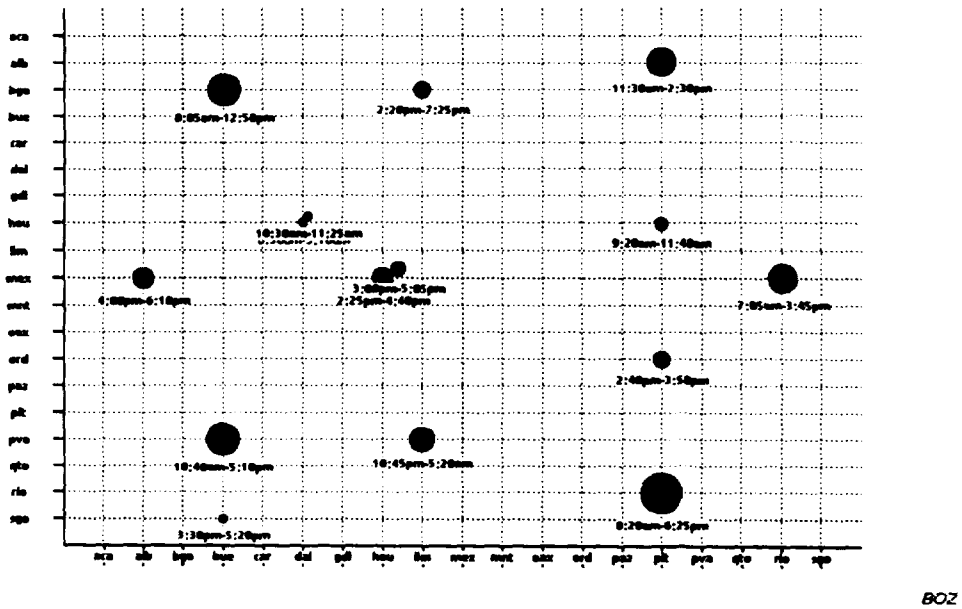


Fig. 2. Graphic for find cheapest flight task.

locate Chicago (ORD) and Mexico (MEX) on the horizontal and vertical axes, find the intersection of these positions, and choose the smallest circle present at that location since the size of the circle representing each flight encodes the airfare.

(3) *Find Most Direct Route.* Suppose you wanted to find the most direct route between two cities, minimizing the number of takeoffs and landings. Using the graphic in Figure 3 you can find your origin and destination cities around the perimeter of the circle and perform a simple connect the dots task, finding the shortest path between the two points. To be sure that the flights connect you must compare the departure and arrival times of each leg of the flight.

(4) *Look Up Flight Information.* Suppose you just wanted to know the time or cost of a flight. The tabular presentation in Figure 4 lets you locate your flight by city in the leftmost column and read off the required information in the corresponding row. In this case the best graphic is no graphic! If your task is to simply read the information, no encoding scheme is likely to prove superior to text.

Overview. The research described in this paper explores an approach to the design of graphic presentations based on an analysis of the tasks for which they are intended to support. The design approach is implemented in an automated graphic design and presentation tool called BOZ. BOZ designs graphic presentations that are customized to the requirements of a stated task. The core idea behind BOZ can be summarized as follows: *since the*

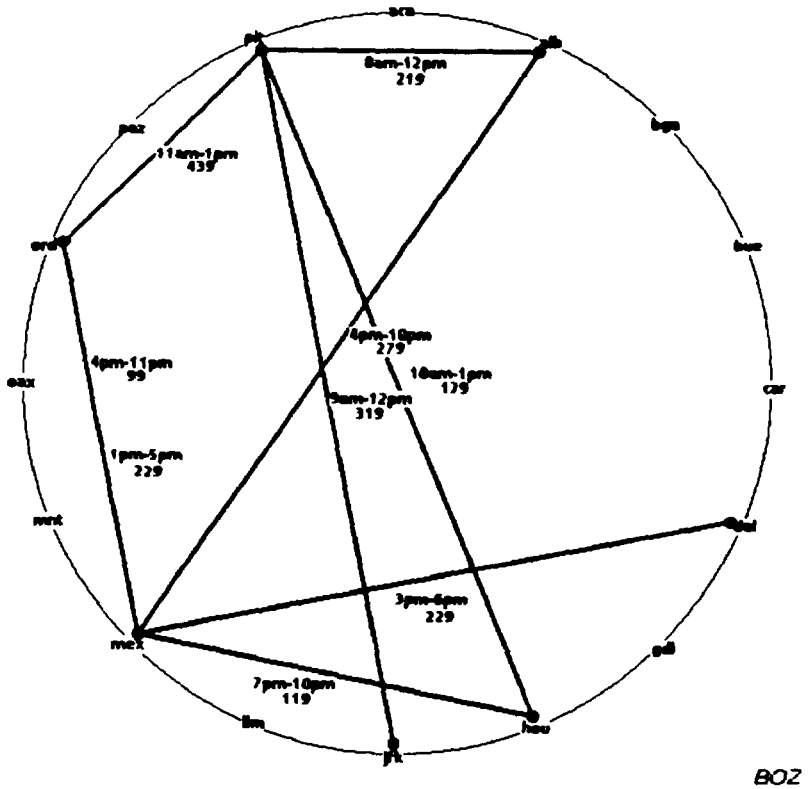


Fig. 3. Graphic for find most direct route task.

ORIGIN	DESTINATION	AVAILABILITY	COST	DEPARTURE	ARRIVAL
pit	hou	ok	179	10am	1pm
pit	alb	ok	219	8am	12pm
pit	jfk	ok	319	9am	12pm
pit	ord	ok	439	11am	1pm
ord	mex	full	99	4pm	11pm
ord	mex	full	229	1pm	5pm
hou	mex	ok	119	7pm	10pm
alb	mex	ok	279	4pm	10pm
dal	mex	ok	229	3pm	6pm

Fig. 4. Graphic for look up information task.

usefulness of a graphic presentation is a function of the task that the graphic will be used to support, graphic design should focus on designing efficient perceptual procedures to be performed by human users. Decisions made about how to encode and structure information in an accompanying graphic should be based primarily on supporting efficient and accurate performance of the

perceptual procedure. The enabling step in the task-analytic approach is to capture the notion of a perceptual procedure performed by human users within the context of a graphic presentation using the same formal framework used to describe abstract computational processes, allowing design decisions to follow formal criteria.

Section 2 reviews previous work related to the problem of designing graphic presentations. Section 3 overviews the five main components of BOZ and introduces a running example used throughout the paper to describe BOZ's approach. Section 4 describes a language used to create logical descriptions of user tasks that are submitted to BOZ as input. Section 5 shows how alternative perceptual procedures can be derived from a logical description of a user task by substituting perceptual inferences in place of logical inferences when the logical and perceptual inferences can be shown to yield the same result. Section 6 shows how an analysis of the relationships between task inferences can be used to determine how information can be structured within a graphic such that visual search is minimized when users perform a perceptual procedure. Section 7 describes three criteria used by BOZ when selecting a single perceptual procedure and graphic design that best supports performance of the user's task. Section 8 describes an automated rendering component that transforms logical facts to graphical facts and displays them on the screen. Section 9 analyzes the cognitive advantages of a perceptual procedure and graphic designed by BOZ. The analysis produces a set of specific hypotheses about the potential task performance efficiencies of the BOZ-designed procedure and graphic. Finally, Section 10 reviews an experiment reported in Casner and Larkin [8] in which participants performed a BOZ-designed perceptual procedure using a series of accompanying BOZ-designed graphics. Results show significant decreases in users' performance times and suggest that users obtained the efficiency savings through the hypothesized perceptual inference substitutions and visual search reductions.

2. PREVIOUS WORK

The following surveys theoretical and empirical work concerned with the problem of designing effective graphics.

2.1 Graphic Design Practices

Tufte [32, 33], Bertin [2], Cleveland [10], and Schmid [28] present many inviting examples of graphics and describe general graphic design practices. These practices help designers to make use of graphic techniques that have been observed to be useful and to avoid bad practices known to make graphics ambiguous, confusing, or generally less usable. There are two basic limitations of these works. First, since the works do not well articulate the reasons *why* graphics succeed in practice, the graphic design practices provide little prescriptive information about how to design a graphic from scratch. Stating that a graph "reveals" a data set in one case tells us little about how to go about designing a graphic that reveals other data sets. Second, the graphic design practices focus mainly on the information to be presented in a graphic and include less concern for the tasks for which the

graphics are designed to support. This approach offers little guidance when designing different graphics to support different tasks.

2.2 Automated Graphic Presentation Systems

APT [26] is an automated graphic presentation tool that designs static, 2D presentations of relational information. A significant contribution of APT was to formally characterize something to which many previous investigators informally alluded. Graphic presentations can be expressed as sentences in formal graphical languages that have the same precise syntax and semantics as propositional formalisms. The advantage of having a formalism for graphic presentations is that it provides a set of criteria for deciding the role of each visible sign or symbol placed in a graphic and improves the integrity of a graphic presentation by using formal methods for transforming logical facts to graphical facts. APT's style of analysis for formal graphical languages has been used by nearly every graphic presentation tool designed after APT, including the one described in this paper. A second contribution of APT is that, unlike proposals for graphic design practices, APT designs graphics with a minimum amount of intervention on the part of the designer; that is, APT embodies a genuinely *prescriptive* theory of how to design a graphic. However, APT's design algorithm is based on an analysis of the information to be presented and does not explicitly consider the task for which a graphic is to be used. This prevents APT from directly exploiting the task-related advantages of graphics and from creating different presentations of the same information to support different tasks.

SAGE [27] is a hybrid, text, and graphics presentation system that generates explanations of changes that occur in quantitative modeling systems such as project modeling and financial spreadsheets. Graphic presentations are designed by SAGE in response to information queries made by the user. Through an analysis of user queries, SAGE's design of graphic presentations is sensitive to the goals of the user, taking an important step toward exploiting the task-related advantages of graphics. SAGE presently contains only a small set of primitive problem-solving operators and is not able to fashion presentations to support complex information-processing tasks involving combinations of many primitive operators.

AIPS [36] accepts descriptions of information encoded using the KL-ONE [3] knowledge representation language. AIPS matches the KL-ONE descriptions against a set of predefined presentation formats and chooses that format that best matches the characteristics of the data. AIPS is not able to design novel graphic presentations.

BHARAT [16] accepts descriptions of data sets and chooses display formats to present relations between the data. The presentation format chosen is determined by the characteristics of the data: line charts are used for continuous data, pie charts for proportional data, and bar charts for all others. Like AIPS, BHARAT chooses among existing presentations rather than designing new ones.

VIEW [15] creates graphic presentations of information about ships maintained in a naval database. VIEW's knowledge base contains information

about particular users, a set of tasks for the domain and a set of predefined KL-ONE descriptions of possible presentation formats. By matching users' identities, tasks, and queries against the presentation format descriptions, VIEW is able to present different graphics in different contexts. As with AIPS and BHARAT, VIEW does not design its presentation formats.

APEX [14] creates graphical explanations of actions performed with physical devices in a three-dimensional world. Explanations are created by presenting sequences of static images depicting the individual steps in an action. APEX uses hierarchical descriptions of the objects that can appear in an explanation where each level in the hierarchy contains more detailed features of the object. A second mechanism allows APEX to determine how much detail is needed at each step and to display only that information. Since the objects that appear in explanations are highly domain specific, they must be hand created prior to using APEX.

2.3 Cognitive Research on the Utility of Graphic Presentations

Larkin and Simon's [24] studied the utility of graphics from a cognitive science perspective. Larkin and Simon built detailed cognitive simulations of human task performance with each simulation performing a task using equivalent logical and graphical representations of a set of data. Larkin and Simon's analysis yielded two ways in which graphic presentation-based procedures could be performed more efficiently by humans: (1) by allowing users to substitute quick perceptual inferences for more demanding logical inferences, and (2) by reducing search for needed information.

Several more recent studies have investigated other cognitive utilities of graphic presentations. Hegarty and Just [17] studied the use of diagrams in understanding complex machines. Fallside [13] investigated how learners make use of animated diagrams similar to those produced by Feiner's APEX program when understanding complex machines. Koedinger and Anderson [23] examined the role of diagrams in geometry learning and instruction. Many other cognitive advantages of graphics have been proposed. Graphics appear to help improve recall for presented data, reduce short-term memory loads during problem solving, provide information about the state of a problem solution, help users organize their knowledge about a problem, and increase motivation or user satisfaction. A future articulation of how these advantages relate to specific graphic design features may allow them to be incorporated in a prescriptive theory of graphic presentation design.

3. BOZ: AUTOMATED TASK-ANALYTIC DESIGN OF GRAPHIC PRESENTATIONS

The following describes a technique for designing graphic presentations of relational information based on an analysis of the task that a graphic is intended to support. The graphic design technique is articulated in an automated graphic design and presentation tool called BOZ. BOZ focuses on designing perceptual procedures, to be performed by human users, that allow users to accomplish a stated goal more efficiently than they would be able to without the benefit of a graphic presentation. BOZ's task-analytic approach uses the following five components.

(1) A *logical task description language* allows the user of BOZ to describe the information-processing task that she or he wishes to design a graphic to support. This language is used to enumerate the individual problem-solving steps (called logical operators) that are required for a user to complete a task without the benefit of any information presentation. Logical task descriptions must be prepared by hand and submitted to BOZ as input. The task description language is also used to describe the informtion manipulated by a task.

(2) A *perceptual operator substitution* component considers each operator in a logical task description looking for ways to substitute perceptual operators in place of logical operators when the operators can be shown to produce the same output given the same input. BOZ contains a catalog of perceptual operators describing problem-solving steps performed within the context of a graphic. Perceptual operator substitution is the mechanism used to streamline the inferencing done by the human user when performing a task. Several perceptual operators typically qualify as substitutes for each logical operator, yielding a set of possible perceptual procedures.

(3) A *perceptual data structuring* component examines the information manipulated by each logical operator and determines how information shared by several operators should be collected together to form complex graphical objects, and how unrelated information can be partitioned into distinct presentations. Perceptual data structuring is one mechanism used to minimize the amount of time the user spends searching for information in a graphic. The perceptual data structuring component determines the optimal grouping and distribution of information within a graphic. It does not determine how the information is to be perceptually encoded in the graphic.

(4) A *perceptual operator selection* component chooses a single perceptual operator to substitute each logical operator in a task description. The first criteria for perceptual operator selection is how efficiently and accurately each perceptual operator is likely to be performed by human users. Selecting each particular perceptual operator also decides the way that the information manipulated by that operator must be perceptually encoded in a graphic. A second criteria for operator selection is choosing a complete set of perceptual operators that results in a set of graphical encodings that can be combined according to the specification produced by the perceptual data structuring component. The perceptual operator selection component yields a detailed description of a single perceptual procedure and an accompanying graphic design that supports the performance of the perceptual procedure.

(5) A *rendering* component translates logical facts into graphical facts and displays them on the computer screen following the graphic design produced by BOZ. Graphics produced by the rendering component support two-way interactions that allow changes made in the internally stored logical facts to be automatically reflected in the graphic presentation and direct manipulations of the graphical objects in the presentation to be reflected in the internal set of logical facts.

The next five sections describe the components of BOZ in detail. To illustrate how BOZ works, a running example is developed throughout the discussion. In the example, a graphic presentation is designed to support one additional airline reservation task:

> *Find a pair of connecting flights that travel from Pittsburgh to Mexico City. You are free to choose any intermediate city as long as the layover in that city is no more than 4 hours. Both flights that you choose must be available. The combined cost of the flights cannot exceed* $500. *Find an available seat on each flight.*

This task differs from the previous four reservation tasks in that the user must now be concerned with obeying both time and cost constraints. We will see that the requirements of this task lead BOZ to produce a graphic presentation completely different from those designed for the four previous reservation tasks.

4. LOGICAL TASK DESCRIPTION LANGUAGE

The first component of BOZ provides a means of describing the information-processing activities that a graphic is intended to support. Task descriptions must presently be prepared in advance and submitted to BOZ as input.[1] BOZ's task description language contains two basic components: (1) a notation for describing logical procedures, and (2) a notation for expressing logical facts manipulated by a logical procedure.

Logical procedure definitions are similar to programs in conventional programming languages such as Pascal. Every logical procedure contains three parts: (1) a set of domain set definitions; (2) a set of logical operator definitions; and (3) a main body. *Domain sets* are the information types that define the universe of discourse for a task. Domain sets consist of a name, a type of information, and a (possibly infinite) set of data values of that type. Three types of domain sets are allowed by BOZ: *quantitative*, *nominal*, and *ordinal* [30]. A *logical operator* (LOP) is composed of an operator name, a list of arguments taken as input to the operator, and a single relation that the operator computes. Logical operators occur in two forms. A *search operator* uses one of the three meta-commands: ASK, TELL, and RETRACT to query, assert, and remove logical facts from a simple database of logical facts. Logical facts describe object-attribute-value relations and take the form: (⟨attribute⟩ ⟨object⟩ ⟨value⟩), where ⟨attribute⟩ names some property of ⟨object⟩, and ⟨value⟩ is the property value. The arguments ⟨object⟩ and ⟨value⟩ in a search operator may be instantiated or uninstantiated. *Uninstantiated arguments* (i.e., variables that have not yet been assigned a value) are capitalized. *Instantiated arguments* are variables that were previously uninstantiated but have since been assigned a value. Instantiated arguments appear in lower case. A *computation operator* describes computations performed on a

[1]Section 11 discusses prospects for automating the process by which users communicate their tasks and goals to BOZ.

set of arguments using one of a set of predefined arithmetic or logical predicates: +, −, *, /, AND, OR, and NOT. Note that only instantiated arguments may appear in a computation operator.

The following are two logical operators in the airline reservation procedure. The two operators determine the departure time of an airline flight (search) and the layover between two flights (computation), respectively:

```
(LOP determineDeparture (⟨flight⟩ ⟨DEPARTURE⟩)
  (ASK (Departure ⟨flight⟩ ⟨DEPARTURE⟩)))
(LOP computeLayover (⟨departure⟩ ⟨arrival⟩ ⟨LAYOVER⟩
  (− ⟨departure⟩ ⟨arrival⟩ ⟨LAYOVER⟩))
```

The keyword LOP is used to denote a logical operator. The lists (⟨flight⟩ ⟨DEPARTURE⟩) and (⟨departure⟩ ⟨arrival⟩ ⟨LAYOVER⟩) are the sets of arguments that the two operators receive as input. The ASK predicate states that a list of facts should be checked to see if the relation that follows can be shown to be true, namely, if there exists a fact expressing the departure time of the flight. The relation (− ⟨departure⟩ ⟨arrival⟩) specifies that the predefined subtraction predicate is to be computed given the values ⟨departure⟩ and ⟨arrival⟩ and the variable ⟨LAYOVER⟩ is to be instantiated with the result.

The *main body* of a logical procedure is an ordered sequence of calls to the set of defined logical operators. To express control information, the main body of a logical procedure may additionally contain any of the following control constructs: **while-do**, **for**, **repeat-until**, and **if-then**.

To illustrate how logical procedures are described using the task language, Figure 5 shows a complete procedure description for the airline reservation task.[2] The procedure in Figure 5 assumes that there are no direct flights from Pittsburgh to Mexico City.

Logical facts are used to describe relational information manipulated by a logical procedure. Logical facts state relations between values drawn from one or more domain sets. The airline reservation procedure manipulates information from the domain sets specified in the top portion of Figure 5. Figure 6 shows logical facts that describe a set of three airline flights.

5. PERCEPTUAL OPERATOR SUBSTITUTION

Perceptual operator substitution is the graphic design technique used to insure that a graphic presentation best exploits the first task-specific advantage of graphics: users can substitute efficiently performed perceptual inferences in place of more demanding logical inferences. The perceptual operator substitution component considers each logical operator appearing in a logical procedure looking for ways of substituting perceptual operators in place of logical operators when the logical and perceptual operators can be shown to be equivalent. The perceptual operator substitution component produces a set

[2]Note that there exist many variations of the procedure shown in Figure 5. The user might alternatively start by searching for flights that terminate in Mexico City and work backwards, or start by choosing an intermediate city and then searching for two flights that arrive from Pittsburgh and leave for Mexico City.

```
(PROCEDURE
 (let ((found nil))
  (while (and found (findFlightWithOrigin FLIGHT 'pit)) do
   (if (available? flight 'T) then
    (findDestination flight LAYOVERCITY)
    (determineArrival flight ARRIVAL)
    (while (and found (findFlightWithOrigin CONNECTING layovercity)) do
       (if (available? connecting 'T) then
        (findDestination flight FINALDESTINATION)
        (if (landsInDestinationCity? finaldestination 'mex) then
         (determineDeparture connecting DEPARTURE)
         (computeLayover departure arrival LAYOVER)
         (if (and (connecting? departure arrival)
                  (layoverLessThanX? layover '4)) then
          (determineCost flight COST1)
          (determineCost connecting COST2)
          (addCosts cost1 cost2 TOTAL)
          (if (costLessThanX? total '500) then
           (repeat
            (findSeat flight SEAT1)
           until (emptySeat? seat1 'T))
           (findSeatNumber seat1 SEATNUM1)
           (repeat
            (findSeat connecting SEAT2)
           until (emptySeat? seat2 'T))
           (findSeatNumber seat2 SEATNUM2)
           (if (and seat1 seat2)
            (setq found t)]

(DOMAINSETS
 (flight NOMINAL 50)
 (origin NOMINAL (pit hou dal ord alb mex gdl qto paz bga))
 (destination NOMINAL (pit hou dal ord alb mex gdl qto paz bga))
 (departure QUANTITATIVE 1440)
 (arrival QUANTITATIVE 1440)
 (layover (departure arrival))
 (cost QUANTITATIVE 1000)
 (availability NOMINAL (ok full))
 (seat NOMINAL 144)
 (seatnumber ORDINAL (1A 1B 1C 1D 1E 1F ... 24A 24B 24C 24D 24E 24F))

(OPERATORS
 (LOP findFlightWithOrigin (<FLIGHT> <origin>)
  (ASK (Origin <FLIGHT> <origin>)))
 (LOP findDestination (<flight> <DESTINATION>)
  (ASK (Destination <flight> <DESTINTATION>)))
 (LOP landsInDestinationCity? (<destination> <destination>)
  (= <destination> <destination>)))
 (LOP available? (<flight> <availability>)
  (ASK (Availability <flight> <availability>)))
 (LOP determineDeparture (<flight> <DEPARTURE>)
  (ASK (Departure <flight> <DEPARTURE>)))
 (LOP determineArrival (<flight> <ARRIVAL>)
  (ASK (Arrival <flight> <ARRIVAL>)))
 (LOP computeLayover (<departure> <arrival> <LAYOVER>)
  (- <departure> <arrival> <LAYOVER>))
 (LOP connecting? (<departure> <arrival>)
  (> <departure> <arrival>))
 (LOP layoverLessThanX? (<layover> <layover>)
  (< <layover> <layover>))
 (LOP determineCost (<flight> <COST>)
  (ASK (Cost <flight> <COST>)))
 (LOP addCosts (<cost> <cost> <COST>)
  (+ <cost> <cost> <COST>))
 (LOP costLessThanX? (<cost> <cost>)
  (< <cost> <cost>))
 (LOP findSeat (<flight> <SEAT>)
  (ASK (Seat <flight> <SEAT>)))
 (LOP emptySeat? (<seat> <availability>)
  (ASK (Availability <seat> <availability>)))
 (LOP findSeatNumber (<seat> <SEATNUMBER>)
  (ASK (Seatnumber <seat> <SEATNUMBER>))))
```

Fig. 5. Logical airline reservation procedure.

```
(origin flight117 pit)          (cost flight117 179)
(origin flight738 pit)          (cost flight738 219)
(origin flight839 pit)          (cost flight839 319)
(destination flight117 hou)     (departure flight117 10:00)
(destination flight738 alb)     (departure flight738 8:00)
(destination flight839 jfk)     (departure flight839 9:15)
(availability flight117 ok)     (arrival flight117 12:50)
(availability flight738 ok)     (arrival flight738 12:00)
(availability flight839 ok)     (arrival flight839 12:05)
```

Fig. 6. Logical facts for airline reservation tasks.

of perceptual operators that can potentially serve as substitutes for each logical operator. Decisions about which particular perceptual operator to substitute for each logical operator are subject to further design criteria described in Section 7.

Perceptual operator substitution relies on two important components: (1) a *catalog of perceptual operators* that describes information-processing activities that occur within the context of a graphic presentation; and (2) a *substitution algorithm* that considers each logical operator in a task and searches the catalog of perceptual operators for those perceptual operators that compute the same function as the logical operator. Since there are often several perceptual operators that qualify as substitutes for a logical operator, the perceptual operator substitution component produces a set of possible perceptual procedures equivalent to the logical procedure.

5.1 A Catalog of Perceptual Operators

Perceptual operators (*POPs*), analogous to logical operators, characterize information-processing activities performed within the context of a graphic presentation and whose performance depends on the use of a graphic presentation. Perceptual operators describe perceptual inferences or visual search performed using graphically expressed information. For example, judging the distance between two objects in a graphic and locating an object having a particular color are examples of perceptual operators.

Perceptual operators are organized around a set of *primitive graphical languages* available to the designer of a graphic presentation [26]. Primitive graphical languages comprise the designer's resources for encoding information graphically. The set of primitive graphical languages used by BOZ are shown in Table I. The parenthesized numbers in Table I indicate an upper limit on the number of distinct values that can be practically encoded in a single graphic presentation using each primitive graphical language. Associated with each of the primitive graphical languages is a set of perceptual operators that are admitted when the designer of a graphic presentation elects to use one or more of the primitive languages in a graphic. For example, if we elect to use the Horizontal Position language we admit a family of perceptual operators (POPs) such as determining the horizontal position of a graphical object, comparing two or more horizontal positions, and finding the midpoint of an interval defined by two horizontal positions.

Table I. Primitive Graphical Languages

Horizontal Position (100)	Color (12)
Vertical Position (100)	Labels (∞)
Height (50)	Line Thickness (3)
Width (50)	Line Dashing (2)
Line Length (50)	Shape (5)
Area (10)	Visibility (2)
Shading (4)	Tabular (∞)
Connectivity (8)	

Table II. Perceptual Operators (POPs)

Horizontal Position	**Shading**
`determine-horz-pos`	`determine-shade`
`search-object-at-horz-pos`	`search-object-with-shade`
`search-any-horz-pos-object`	`search-object-and-shade`
`verify-object-at-horz-pos`	`verify-object-and-shade`
`horz-coincidence?`	`darker?`
`left-of?`	`lighter?`
`right-of?`	`same-shade?`
`horz-forward-projection`	
`horz-backward-projection`	
`determine-horz-distance`	

Table II shows the set of perceptual operators admitted by the Horizontal Position and Shading primitive graphical languages. It is interesting to compare the perceptual operators associated with each primitive graphical language. This exercise helps make explicit the task-specific usefulness of each graphical encoding technique. Note the difference in the number of computation operators supported by the Horizontal Position and Shading primitive graphical languages. For instance, human users can easily determine the difference between two horizontal positions but are not generally able to determine the difference between two shades.

Equivalence Classes for Perceptual Operators. Every perceptual operator computes a function over relational information. An *equivalence class* of perceptual operators is a set of operators that can be shown to compute the same function over relational information. Table III describes eleven equivalence classes used to categorize the perceptual operators in the catalog. Every perceptual operator in the catalog is classified under exactly one operator equivalence class. Table IV shows the perceptual operators associated with

Table III. Perceptual Operator Equivalence Classes

SEARCH OPERATORS
search: Given a graphical property, search a set of objects for an object having that graphical property.
lookup: Given an object, determine a specific property of that object.
search and lookup: Search for any object and determine a specified property of that object.
verify: Given an object and property, verify that the object has that property.
COMPUTATION OPERATORS
equal: Do two objects have the same graphical property?
less than: Does one object have a lower valued graphical property than another?
greater than: Does one object have a lower valued graphical property than another?
plus: Compute the sum of two graphical properties.
difference: Compute the difference of two graphical properties.
times: Compute the product of two graphical properties.
quotient: Compute the quotient of two graphical properties.

Table IV. Members of Two Perceptual Operator Equivalence Classes

search	**subtraction**
search-object-at-horz-pos	determine-horz-distance
search-object-at-vert-pos	determine-vert-distance
search-object-with-height	determine-height-difference
search-object-with-width	determine-width-difference
search-line-with-length	determine-difference-in-line-length
search-object-with-area	determine-area-difference
search-connected-object	subtract-labels
search-object-with-shading	determine-slope-difference
search-object-with-color	subtract-table-entries
search-object-with-label	
search-line-with-thickness	
search-line-with-dashing	
search-line-with-slope	
search-object-with-shape	
search-visible-object	
search-entry-in-table	

the **search** and **subtraction** equivalence classes. Perceptual operators are formalized using the same notation used for logical operators. For example, the search-object-with-shade search operator and determine-horz-distance computation operator are defined as follows:

(POP search-object-with-shade (⟨OBJECT⟩ ⟨shade⟩)
    (ASK (Shading ⟨OBJECT⟩ ⟨shade⟩)))

(POP determine-horz-distance (⟨horzpos1⟩ ⟨horzpos2⟩ ⟨DISTANCE⟩)
    (DIFFERENCE ⟨horzpos1⟩ ⟨horzpos2⟩ ⟨DISTANCE⟩))

The search-object-with-shade operator searches for an object in a graphic having a shade equal to the value of ⟨shade⟩ and instantiates the variable ⟨OBJECT⟩ with the result. The determine-horz-distance operator determines the distance between two objects located at ⟨horzpos1⟩ and ⟨horzpos2⟩ in a graphic and instantiates the variable ⟨DISTANCE⟩ with the result.

5.2 Substituting Operators

The formal characterization of logical and perceptual operators using equivalent notations allows BOZ to design perceptual tasks that allow users to accomplish the same results as a logical task submitted to BOZ as input. The ultimate goal of BOZ is to arrive at that perceptual task that is equivalent to the original logical task, and that is most easily and efficiently performed by human users. To design a perceptual task, BOZ considers each logical operator in a task description and searches the catalog of perceptual operators attempting to locate those POPs that can be shown to compute the same function as a LOP. Insisting on operator equivalence insures that whatever perceptual procedure is followed in place of a corresponding logical procedure, it is guaranteed that the user will obtain the same results if the perceptual procedure is performed correctly.

A perceptual operator qualifies as a substitute for a logical operator if and only if the logical operator can be categorized in the same equivalence class (see Table III) as the perceptual operator. Categorization is determined by attempting to match an LOP to an arbitrary member of an equivalence class. If an LOP is successfully matched to a member of an equivalence class, all perceptual operators in that class initially qualify as substitutions for the LOP. For example, given the computeLayover logical operator in the airline reservation procedure:

```
(POP computeLayover (⟨departure⟩ ⟨arrival⟩ ⟨LAYOVER⟩)
    ( – ⟨departure⟩ ⟨arrival⟩ ⟨LAYOVER⟩))
```

BOZ attempts to classify the LOP into each of the equivalence classes given in Table III until a class is found or the set of classes is exhausted. The computeLayover LOP can be successfully categorized into the **subtraction** class of search operators. One member of the **subtraction** class is the determine-horz-distance operator associated with the Horizontal Position language and shown above. Since both operators compute the subtraction function and we can map the arguments of the LOP onto those of the POP, any graphic that represents departure and arrival times as objects positioned along a horizontal axis will always allow the user to perform the determine-horz-distance operator and obtain the same answer produced by the computeLayover operator.

Figure 7 shows the classifications for the logical operators in the airline reservation task. Each operator in the task can be categorized into a single equivalence class. Consequently, a set of perceptual operators initially qualify as substitutions for each logical operator in the task. For example, the list of perceptual operators associated with the **search** equivalence class are proposed as substitutions for the findFlightWithOrigin and findSeat operators.

```
findFlightWithOrigin (search)
findDestination (lookup)
landsInDestinationCity? (verify)
available? (verify)
determineDeparture (lookup)
determineArrival (lookup)
computeLayover (subtraction)
connecting? (greaterthan)
layoverLessThanX? (lessthan)
determineCost (lookup)
addCosts (addition)
costLessThanX? (lessthan)
findSeat (search)
emptySeat (verify)
findSeatNumber (lookup)
```

Fig. 7. Operator classifications for airline reservation task.

Similarly, the operators of the **subtraction** class match the description of the computeLayover operator.

It is important to note that BOZ has not yet decided *which* perceptual operator to choose in each case. Decisions about which perceptual operators to match with each logical operator are subject to further constraints computed by the perceptual data structuring and perceptual operator selection components described in Sections 6 and 7. What BOZ has produced at this stage is a space of perceptual procedures that may be selected according to additional design criteria.

6. PERCEPTUAL DATA STRUCTURING

The perceptual data structuring component is the design technique used to implement the second type of cognitive advantage of graphic presentations: graphics sometimes allow users to spend less time searching for needed information. The perceptual data structuring component examines the information required to perform each logical operator in a task. Two types of analyses are performed using this information. First, by noting the domain sets that each logical operator manipulates, BOZ determines what information should appear in a graphic designed to support a task. Second, by analyzing the relationships between the operators in a task in terms of the domain sets of information they manipulate, BOZ determines: (a) how information shared by several operators should be collected in the same spatial locality and graphically encoded using the same primitive graphical language and (b) how information not shared among operators can be partitioned into distinct presentations. The perceptual data structuring component produces a *perceptual data structure specification* that outlines the presentations that will be used to support the task, the information that should appear in each presentation, and how the information is to be grouped within each presentation. The perceptual data structuring component does *not* decide how information is to be graphically encoded in the presentation. These decisions are made by the perceptual operator selection component (Section 7).

[flight] x [origin] x [destination] x [availability] x [departure] x [arrival] x
[layover] x [cost] x [seat] x [seatnumber]

Fig. 8. Feature space for airline reservation task.

The remainder of the section describes a scheme that analyzes relationships between operators by representing each operator as a vector defined over the domain sets that the operator manipulates. Relationships between vectors are determined by identifying common domain sets occurring in vectors. A complete sketch of all relationships between vectors reveals how information is to be collected together into graphical objects and partitioned among presentations.

6.1 Operator Vectors

Recall that every task description is defined over a finite collection of domain sets. When taken together, all of the domain sets used by a task description form a feature space. A *feature space* is formally defined as the cross product of all domain sets spanned by a task description. Figure 8 shows an example of a feature space defined over the domain sets that pertain to the airline reservation task.

Each logical operator in a task description computes a relation over one or more domain sets in the feature space defined for that task. BOZ makes explicit the domain sets relevant to each logical operator in a task description using a construct called a *vector*. For every logical operator in a task, BOZ generates a corresponding vector that indicates the domain sets of information manipulated by that operator. There are two types of vectors that correspond to the two types of logical operators: search vectors and computation vectors. A *search vector* contains the names of the two domain sets from which the arguments of a search operator are drawn. The first element of a search vector is treated as the *object*, the second domain set as an *attribute* of the object. For example, the findFlightWithOrigin logical operator defines the vector: ⟨flight,origin⟩, where flight is the object, and origin is the attribute. A *computation vector* contains the names of all domain sets that appear as arguments to the computation operator. For example, the computeLayover operator defines the vector: ⟨departure,arrival,layover⟩ since it manipulates arguments drawn from these three domain sets. The complete set of vectors defined by the logical operators in the airline reservation task are shown in Figure 9.

6.2 Relationships Between Vectors

BOZ's next step is to examine the relationships between the vectors in terms of the domain sets they manipulate. There are four types of relationships that can hold between vectors. BOZ determines relationships between vectors

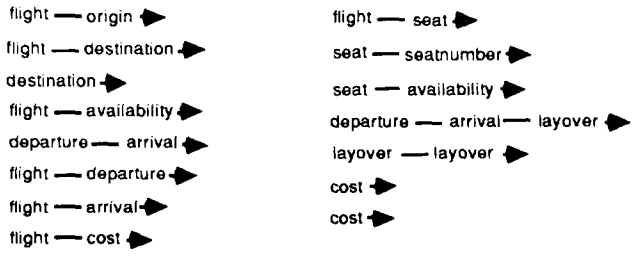


Fig. 9. Vectors for airline reservation task.

by applying an ordered set of four rules to the complete set of vectors for a task.

(1) *Conjoint.* Search vectors $s_1 = \langle o_1, a_1 \rangle$ and $s_2 = \langle o_2, a_2 \rangle$ are *conjoint* when they contain common objects, $o_1 = o_2$. For example, the vectors defined by the determineDeparture and determineArrival operators are conjoint. Conjoint vectors group together attributes that pertain to the same object. Consequently, these attributes should be encoded in a single graphical object in order to reduce eye movement over the presentation when searching for that object and its attributes.

(2) *Parallel.* Search vectors $s_1 = \langle o_1, a_1 \rangle$ and $s_2 = \langle o_2, a_2 \rangle$ are *parallel* when there exists a computation vector, $c = \langle a_1, a_2 \rangle$, that contains both a_1 and a_2. Parallel vectors indicate that some computation operator requires that two or more different objects (e.g., flight and seat) and their attributes be coordinated in order to draw a particular inference. Consequently, both objects should appear in the same graphic presentation and should be encoded using the same primitive graphical language. In the airline example, none of the vectors are parallel.

(3) *Orthogonal.* Search vector $s_2 = \langle o_2, a_2 \rangle$ is *orthogonal* to search vector $s_1 = \langle o_1, a_1 \rangle$ if the attribute of s_2 appears as the object in s_1, that is, $o_1 = a_2$. In the airline example, the search vectors corresponding to the seat object are orthogonal to the flight object vectors. Orthogonality indicates part-of relationships between objects. Domain sets manipulated by orthogonal vectors should appear in separate nested presentations. That is, the user should be able to view the part-of presentation by making an appropriate selection in the first presentation. Note that it is possible for two vectors to be orthogonal to each other.

(4) *Disjoint.* Search vectors s_1 and s_2 are *disjoint* when they share no common object or attribute. Disjoint vectors indicate that no computation operator requires that two or more objects be coordinated to draw an inference. Information relevant to disjoint vectors should appear in different presentations since the task does not require that the information be used together. None of the airline search vectors are disjoint.

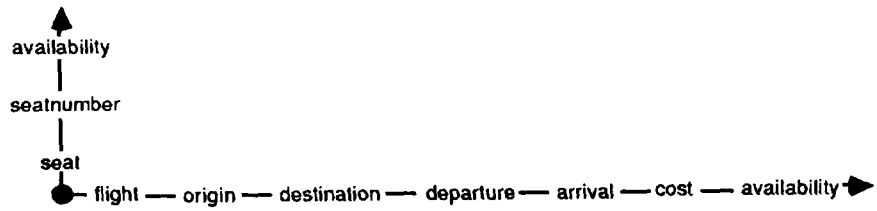


Fig. 10. Vector relationships for airline reservation task.

```
(NESTED (PRESENTATION1 (flight (Origin Destination Departure
                                Arrival Cost Availability))
        (PRESENTATION2 (seat (Seat Availability))))
```

Fig. 11. Initial perceptual data structure specification for airline reservation task.

Applying the vector relationship rules to the vectors in Figure 9 yields the arrangement shown in Figure 10. Since the vectors pertaining to origin, destination, departure, arrival, cost, and availability are conjoint, BOZ constrains them to be encoded in the same graphical object. Similarly, the vectors pertaining to seat number and availability are conjoint and thus form their own graphical object. The flight and seat vectors are not parallel since the task does not require the user to coordinate information about flights and seat (e.g., subtract seat number from arrival time). The seat vectors are orthogonal to the flight vectors indicating that the seat object is a part of the flight object. Hence, BOZ constrains the seat graphical objects to appear in a separate presentation that is nested inside each flight box.

BOZ uses the vector relationships shown in Figure 10 to specify how information will be grouped together into complex graphical objects and distributed among distinct graphic presentations. The initial perceptual data structure specification for the airline task is shown in Figure 11.

It is important to note that BOZ has not yet decided how facts about the origin, destination, departure, etc., of a flight are to be encoded in the graphic. That is, BOZ has not yet associated the names of primitive graphical languages with the predicate names appearing in the perceptual data structure specification. Which primitive graphical languages to associate with each predicate is determined by the perceptual operators selected to substitute the logical operators in the task. Note that information about flight numbers and layovers will not be encoded in any graphic. This has occurred since information about flight numbers is never used in the task and facts about layovers are produced as the results of the computeLayover computation operator.

7. PERCEPTUAL OPERATOR SELECTION

The perceptual operator selection component chooses a single perceptual operator to substitute each logical operator from the list of possibilities

generated by the perceptual operator substitution component. Selecting a single perceptual operator to substitute each logical operator accomplishes two things: (1) reduces the space of possible perceptual procedures to a single perceptual procedure judged to be the most effective, and (2) allows BOZ to design a single accompanying graphic that supports human performance of the selected perceptual procedure.

Three important issues constrain the selection of perceptual operators. First, since the goal is to arrive at a perceptual procedure that minimizes the effort required to correctly complete a task, for each logical operator BOZ seeks to choose that perceptual operator that is performed most efficiently and accurately by human users. A first criteria for operator selection involves estimating the relative performance efficiency and accuracy of the perceptual operators. Second, recall that each perceptual operator is associated with a primitive graphical language that must be used to graphically encode information manipulated by that operator. A second criteria when selecting operators is that the representational power of the primitive graphical language associated with a candidate perceptual operator is sufficient to encode the logical facts manipulated by the operator. Third, recall that the perceptual data structure specification produced by the perceptual data structuring component constrains some domain sets of information to be represented in a single graphical object or using the same primitive graphical language. A third criteria for operator selection is that the primitive graphical languages associated with the selected perceptual operators be combinable such that they result in coherent graphic presentations that agree with the perceptual data structure specification for the task.

7.1 Human Performance Rankings for Perceptual Operators

The most important criteria when selecting a perceptual operator is choosing that operator that allows the human user to obtain the results of the operator most efficiently and accurately. To determine which of a set of perceptual operators is likely to be the most performance effective, BOZ uses a two-tier ranking system that is a generalization of the approach used in Mackinlay's APT program [26]. The first tier ranks the equivalence classes for logical operators in order of their relative difficulty. For instance, arithmetic operators require more effort to perform than lookup operators. Consequently, arithmetic operators are always awarded the most efficient perceptual operators. The second tier ranks the perceptual operators within each perceptual operator equivalence class. For instance, determining the horizontal distance between two points on a scale is generally performed more efficiently and accurately than determining the difference between two areas. Similarly, searching for an object having a particular color is performed more efficiently than searching for an object having a particular shape. An important observation is that the operators associated with each different primitive graphical language best support different tasks. That is, it is impossible to say that any one primitive graphical language is the most effective way to encode data in a graphic. Primitive graphical languages too fall in and out of usefulness with respect to the task to be supported.

Table V. Ranking of Perceptual Operators and Equivalence Classes

A. Class Rankings:

1. plus, difference, quotient, times
2. search
3. less than, greater than
4. equal
5. search and lookup
6. lookup
7. verify

B. Operator Rankings:

SEARCH OPERATORS

search: {Visibility, HorzPos, VertPos, Shape, Connectivity, Shading, Height, Width, LineDashing, LineLength, LineThickness, Labels, Area}

lookup, verify, search and lookup: {Shading, Shape, Labels, Height, Width, LineDashing, LineThickness, Connectivity, HorzPos, VertPos, LineLength, Area, Visibility}

COMPUTATION OPERATORS

equal: {Labels, Shading, HorzPos, VertPos, Shape, LineDashing, Height, Width, LineThickness, LineLength, Connectivity, Visibility, Area}

lessthan, greaterthan: {Shading, HorzPos, VertPos, Height, Width, LineThickness, LineLength, Labels, Connectivity, Shape, LineDashing, Visibility, Area}

plus, times: {Height, Width, LineLength, LineThickness, HorzPos, VertPos, Labels, Connectivity, Shading, LineDashing, Shape, Area}

difference, quotient: {HorzPos, VertPos, Height, Width, LineLength, LineThickness, Labels, Connectivity, Area, Shading, Shape}

BOZ's perceptual operator rankings were generated using a combination of two methods: (1) theoretical predictions based on a more fine-grained consideration of each perceptual operator [5, 34], and (2) experimental observations of human perceptual task performance [11, 12, 20, 31, 34]. Table V shows the rankings for perceptual operator equivalence classes and the rankings for the perceptual operators in each class.

7.2 Expressiveness

The second criterion used during operator selection is that a selected perceptual operator must be associated with a primitive graphical language that is powerful enough to encode the logical facts manipulated by that operator. For example, even though the search-shaded-object is the most efficiently performed search operator, it cannot be selected to substitute the findFlightWithOrigin logical operator since the number of different cities exceeds the number of different shades. When a selected perceptual operator fails to meet the expressiveness needs of a logical operator it is disqualified and the next highest ranking perceptual operator is considered. The interested reader can consult Mackinlay [26] for a thorough analysis of primitive graphical language expressiveness. Like all other recent presentation systems, BOZ adopts Mackinlay's technique for deciding expressiveness.

7.3 Perceptual Operator Combinability

The third criterion for operator selection concerns the combinability of perceptual operators. Suppose we have selected the stack-heights perceptual operator to substitute the addCosts logical operator in the airline reservation

task and are currently selecting a perceptual operator to substitute the findFlightWithOrigin operator. Suppose that we are currently considering the determine-slope perceptual operator as a candidate selection. Recall that the perceptual data structuring component has indicated that the information relevant to these two operators should be encoded in the same graphical object. Every perceptual operator has associated with it a graphical presentation object (defined below) that is used to graphically encode the information manipulated by that object. For example, the graphical presentation object for the stack-heights and determine-slope operators are ⟨rectangle⟩ and ⟨line⟩, respectively. Note that the information relevant to the two operators cannot be encoded in the same graphical object. That is, it is meaningless to speak of the slope of a rectangle or the height of a line. Consequently, these two operators are not combinable and we must disqualify determine-slope as a candidate for selection. Now, suppose we move on and consider the read-label perceptual operator. Note that the two operators are indeed combinable. Even though the graphical presentation object for the read-label operator is ⟨label⟩ and the graphical presentation object for the stack-heights operator is ⟨rectangle⟩, the two graphical objects can be combined to form a labeled rectangle.

The next two sections describe how the set of graphical presentation objects and a set of graphical object composition rules are used by BOZ to decide combinability of perceptual operators.

7.3.1 *Graphical Presentation Objects.* Each primitive graphical language has a *graphical presentation object* associated with it, either: ⟨point⟩, ⟨line⟩, ⟨rectangle⟩, ⟨polygon⟩, or ⟨label⟩. The graphical presentation object for a primitive graphical language is that graphical object that supports the performance of the perceptual operators that are associated with that graphical language. For example, the graphical presentation object for the Height primitive graphical language is ⟨rectangle⟩. Note that only this object makes the perceptual operators associated with the Height language meaningful. That is, it would be impossible to determine the height of a point or a line since points and lines by definition have no height.[3] Table VI lists the graphical presentation objects associated with each of the primitive graphical languages. The first step in deciding perceptual operator combinability is to determine the graphical presentation object of a candidate perceptual operator.

7.3.2 *Composition Rules for Graphical Presentation Objects.* The second step in deciding operator combinability is to compare the graphical presentation object of the perceptual operator currently being considered with the presentation objects of all previously selected operators that appear in the same vector in the perceptual data structure specification. If the graphical presentation object matches those of the previously chosen operators then the

[3]In the case of the ⟨line⟩ object, it is important not to confuse the notion of height with that of line length for a vertically oriented line.

Table VI. Graphical Presentation Objects for the Primitive Graphical Languages

Horizontal Position = <point>	Shading = <polygon>
Vertical Position = <point>	Labels = <label>
Height = <rectangle>	Color = <point>
Width = <rectangle>	Line Thickness = <line>
Line Length = <line>	Line Dashing = <line>
Area = <polygon>	Shape = <polygon>
Connectivity = <line>	Visibility = <point>

new operator is combinable. If the presentation object does not match, BOZ attempts to show them combinable using the set of composition rules for graphical objects given in Table VII. Each composition rule describes how a set of individual presentation objects can be legally composed to form a single presentation object that inherits all of the graphical properties of the constituent objects. For any new perceptual operator and set of previously selected operators, the new operator is combinable if and only if a rule can be found that maps the set of presentation objects into another legal presentation object.

Applying the perceptual operator selection strategy to the set of possible perceptual operators obtained for the airline reservation procedure yields the perceptual procedure shown in Figure 12. For each logical operator, BOZ has selected the most efficient available perceptual operator as a substitute within the expressiveness and combinability constraints. It is interesting to note that the logical operator findFlightWithOrigin has been substituted by the search-object-with-label, a very low-ranking perceptual operator. This example illustrates how the expressiveness and combinability constrains operator selection. A more appealing substitution for findFlightWithOrigin would have been a perceptual operator in which the user locates a flight with a particular origin by searching a horizontal scale or by searching for an object having a particular shape. The reason that these two more attractive operators were not selected is because the higher-ranking computeLayover had already staked claim to the Horizontal Position operators and the Shape primitive graphical language unfortunately does not offer enough unique shapes to represent all ten different possible cities of origin. Consequently, due to these constraints imposed by the other competing operators, the findFlightWithOrigin operator was relegated to the more difficult perceptual task of searching for a labeled item.

Figure 13 shows the final perceptual data structure specification for the airline reservation task after perceptual operators have been selected. Note that each predicate appearing in the perceptual data structure specification has been associated with a single primitive graphical language. In each case the primitive graphical language chosen is precisely that language associated with the perceptual operators that have been selected to manipulate that type of information.

Table VII. Composition Rules for Graphical Presentation Objects

Object Composition Rules:	**Axis Composition Rules:**
RULE 1: <point> + <point> = <point>	RULE 12: <horz-axis> + <horz-axis> = <horz-axis>
RULE 2: <point> + <line> = <line>	RULE 13: <vert-axis> + <vert-axis> = <vert-axis>
RULE 3: <line> + <line> = <line>	RULE 14: <horz-axis> + <vert-axis> = <cart-axis>
RULE 4: <rectangle> + <point> = <rectangle>	
RULE 5: <rectangle> + <rectangle> = <rectangle>	
RULE 6: <polygon> + <point> = <polygon>	
RULE 7: <polygon> + <polygon> = <polygon>	
RULE 8: <label> + <label> = <label>	
RULE 9: <label> + <line> = <line>	
RULE 10: <label> + <rectangle> = <rectangle>	
RULE 11: <label> + <polygon> = <polygon>	

```
(let ((looking t))
 (while (and looking (search-object-with-label FLIGHT 'pit)) do
  (if (shaded? flight) then
   (read-label flight LAYOVERCITY)
   (determine-horz-pos flight ARRIVAL)
   (while (and looking (search-object-with-label CONNECTING layovercity)) do
      (if (shaded? connecting) then
       (read-label flight FINALDESTINATION)
       (if (same-labels? finaldestination 'mex) then
        (determine-horz-pos connecting DEPARTURE)
        (determine-horz-distance departure arrival LAYOVER)
        (if (and (right-of? departure arrival)
                 (left-of? layover '4)) then
         (determine-height flight COST1)
         (determine-height connecting COST2)
         (stack-heights cost1 cost2 TOTAL)
         (if (shorter? total '5) then
          (repeat
           (search-object-with-label flight SEAT1)
          until (shaded? seat1))
          (read-label seat1 SEATNUM1)
          (repeat
           (search-object-with-label connecting SEAT2)
          until (shaded? seat2))
          (read-label seat2 SEATNUM2)
          (if (and seat1 seat2)
           (setq looking nil)]
```

Fig. 12. Final perceptual airline reservation procedure.

```
(NESTED (PRESENTATION1 (flight ((Origin Labels) (Destination Labels)
                                (Departure HorzPos) (Arrival HorzPos)
                                (Cost Height) (Availability Shading))
                                <rectangle>)
        (PRESENTATION2 (seat ((Availability Shading)) <rectangle>)))
```

Fig. 13. Final perceptual data structure specification for airline reservation task.

7.4 Limitations of Automated Perceptual Operator Selection

It is important to note that there exists no algorithmic strategy that always chooses the most efficiently or accurately performed perceptual operators, including those based on experimental observations and detailed theoretical predictions. I am aware of no experimental result of people using graphics that has been successfully generalized across any one of the following: user [21], level of skill [17], practice [29], task [19], particular presentation used [25], age [9], culture [18], or even social situation [1]! Each of these factors have been shown to introduce variance strong enough to overturn the results of any particular experiment, making strong generalizations of these results inappropriate. What we can hope to achieve in an automated graphic design tool is a codified set of operational design principles that perform satisfactorily across interesting tasks and graphics.

Many task domains make use of standardized domain-specific graphic conventions that were originally designed using intuition or criteria other than cognitive efficiency. Without specific knowledge of a problem domain, an automated graphic design tool is unable to identify and select operators that correspond to existing graphic conventions and this is a second limitation of automated graph design. BOZ, like APT [26], allows the designer to intervene and manually select perceptual operators in order to support existing conventions.

The human performance efficiency of a perceptual operator may be sensitive to the particular data on which the operator is performed. For example, judging the distance between two points on scale that are aligned three units away from one another appears to be easier than if the points are aligned, say, 39 units apart. This phenomenon occurs because some input data allow the user to exploit more low-level perceptual capabilities such as *subitizing* [22]. A more reliable perceptual operator ranking system might be achieved by making the set of rankings functionally dependent on the set of logical facts to be displayed. That is, rather than using a fixed set of operator rankings, a different set of rankings would be computed for each different set of logical facts. The usefulness of such a scheme, along with its computational feasibility, is an open question.

BOZ's perceptual operator selection component proceeds under the assumption that the time to perform a perceptual operator is unaffected by any other perceptual operators that are performed before or after that operator. That is, the time to perform a perceptual operator is assumed to be context independent. There are two ways in which this assumption may be invalidated. First, while the questions of parallel perceptual operator performance remains open for debate, experimental observations suggest that human users are sometimes able to obtain the results of several perceptual operators in a time less than the sum of the observed performance times for each individual operator [34]. Furthermore, researchers have gained a partial understanding of which combinations of perceptual operators exhibit this property. Combinations of perceptual operators that achieve this property result in a greater savings in performance efficiency. Second, experimental observations of users in other task domains such as typing suggest that certain combinations of operators may sometimes result in a performance time that is greater than the sum of the individual performance times [35].

8. GRAPHIC PRESENTATION RENDERING

The rendering component translates logical facts to graphical facts and displays them on the computer screen. The form in which the graphical facts appear on the screen agrees precisely with the design described by the perceptual data structure specification along with a set of general rendering assumptions. The rendering component produces a fully rendered graphic presentation of the graphical facts. Graphic presentations produced by the rendering component support interactive manipulations of the graphical objects appearing in the presentation, allowing users to effect changes in the

```
(shape obj001 square)
(horzpos obj001 6)
(shading obj001 black)
```

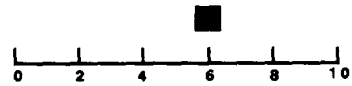


Fig. 14. Example graphical facts.

```
(origin flight117 pit)                    (label flight117 pit)
(origin flight239 hou)                    (label flight239 hou)
(destination flight117 hou)               (label flight117 hou)
(destination flight239 mex)               (label flight239 mex)
(departure flight117 10:00)              (horzpos flight117 10)
(departure flight239 15:00)              (horzpos flight239 15)
(arrival flight117 12:50)              (horzpos flight117 12.83)
(arrival flight239 17:15)              (horzpos flight239 17.25)
(cost flight117 179)                     (height flight117 1.79)
(cost flight239 239)                     (height flight239 2.39)
(availability flight117 ok)        (shading flight117 whiteshade)
(availability flight239 ok)        (shading flight239 whiteshade)
```

Fig. 15. Translated airline reservation facts.

internally stored logical facts by making changes to the graphical displayed facts.

8.1 Translating Logical Facts to Structured Graphical Facts

A prerequisite to graphically rendering arbitrary sets of logical facts on the computer screen is a notation for representing graphical facts that is equivalent to the notation used to express logical facts. To accomplish this, Mackinlay's formalism for expressing graphical facts is used and this has been shown to be equivalent to a logical representation of the same relational information [26]. Mackinlay's formulation allows logical facts to be expressed using each of the primitive graphical languages given in Table 1. Graphical facts expressed in a primitive graphical language take the following form: (PGL ⟨OBJECT⟩ ⟨VALUE⟩). For example, the facts in Figure 14 describe a square-shaped graphical object that is shaded black and positioned along a horizontal axis.

Figure 15 shows how logical facts (left side) about airline flights are translated to graphical facts (right side) when the mapping given in the perceptual data structure specification is applied. A final notation is needed for expressing collections of graphical facts whose structure agrees with the perceptual data structure specification for the task. A *structured graphical fact* corresponds to the "gestalt wholes" defined by the perceptual data structure specification. For example, the facts in Figure 16 show how the graphical facts (Figure 15) are structured according to the perceptual data structure specification for the airline task given in Figure 13. Each structured graphical fact in Figure 16 corresponds to a single flight.

```
(((labels flight117 pit)
  (labels flight117 hou)
  (horzpos flight117 10)
  (horzpos flight117 12.83)
  (height flight117 1.79)
  (shading flight117 whiteshade))
((labels flight239 hou)
  (labels flight239 mex)
  (horzpos flight239 15.00)
  (horzpos flight239 17.25)
  (height flight239 2.39)
  (shading flight239 whiteshade)))
```

Fig. 16. Structured graphical facts.

8.2 Rendering Graphical Facts

The rendering component automatically displays arbitrary sets of structured graphical facts on the computer screen. This is accomplished by considering each structured fact, determining the form in which it is to be presented by consulting the perceptual data structure specification, and rendering the image of the fact on the screen. The rendering component uses an object-oriented approach to rendering structured graphical facts. For every type of graphical presentation object (i.e., ⟨point⟩, ⟨line⟩, ⟨rectangle⟩, ⟨polygon⟩, and ⟨label⟩) there exists a corresponding display object that can be rendered on the screen. Display objects can inherit one or more of a set of display methods that render the graphical properties of a display object. Display methods are defined for each of the primitive graphical languages in Table I. For presentations that do not use horizontal and vertical position to encode information, a simple displacement scheme is used to avoid occlusion of display objects by other display objects. Scales and guidelines are automatically computed, drawn, and labeled using the DOMAINSETS field in the logical procedure description. Fonts have been chosen arbitrarily and standardized. Nested graphics are implemented by mouse-sensitive buttons that are always placed in the lower left corner in rectangles and polygons and immediately on top of points and lines. Display methods are automatically attached to the buttons that cause the nested graphic to be rendered when the button is selected.

Figure 17 shows a fully rendered set of graphical airline facts. As specified by the perceptual data structure specification, the presentation consists of a single type of graphical object (i.e., a flight box) that inherits four graphical properties (i.e., horizontal position, shading, height, and labels). Selecting the seats button for any flight causes the nested seating chart for that flight to be rendered. A rendered seating chart is shown in Figure 18.

The graphics generated by the rendering component support two-way interactions between sets of logical facts and their graphical images. In addition to being able to effect changes in a graphic presentation through manipulations of the internally stored logical facts, the graphical objects in the presentations can be manipulated by the user to effect changes in the set of stored logical facts. For example, the upper, leftmost flight box in the presentation in Figure 17 indicates that there is a flight from Pittsburgh to JFK Airport (New York) leaving at 11:30 am, with no available seats, costing

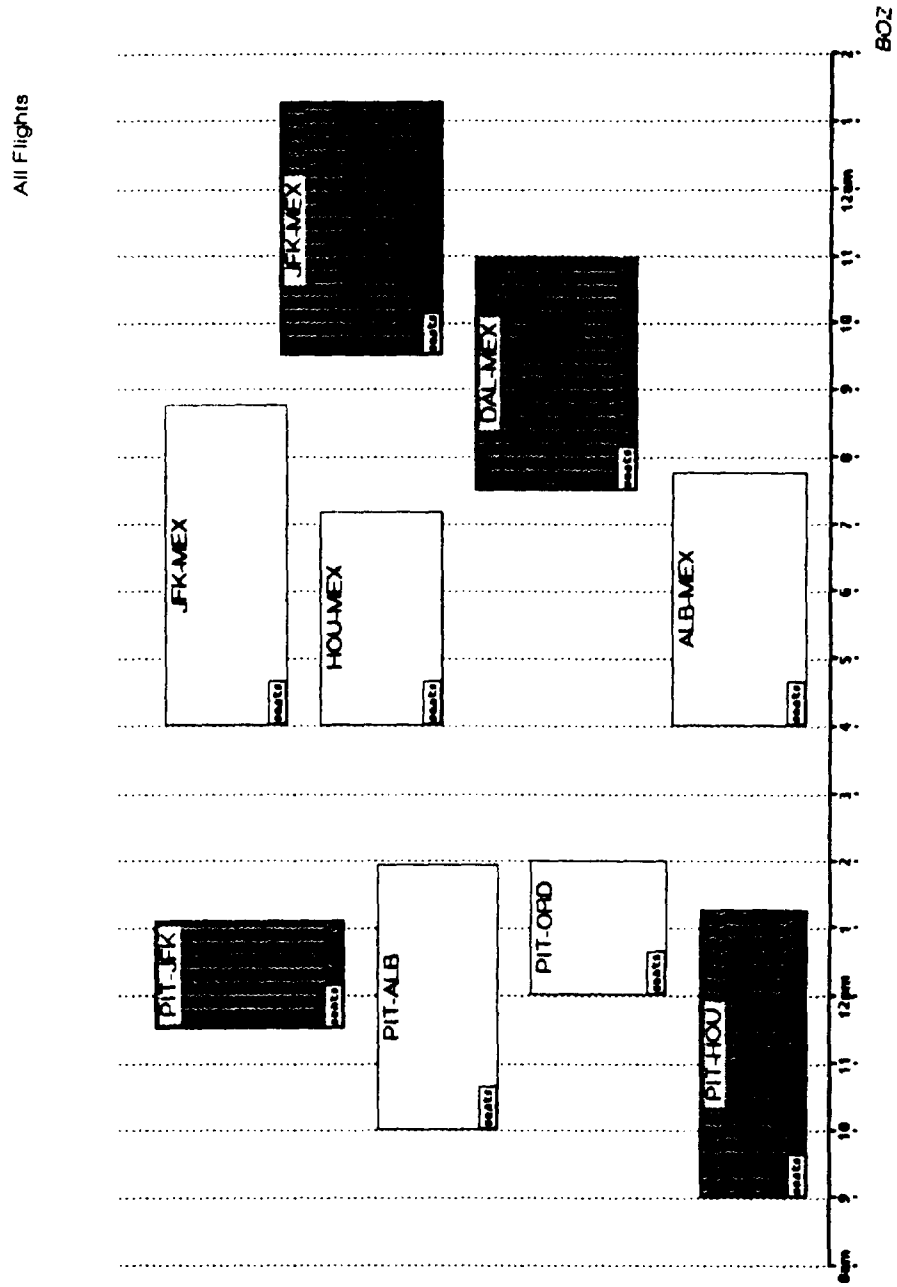


Fig. 17. Rendered airline schedule.

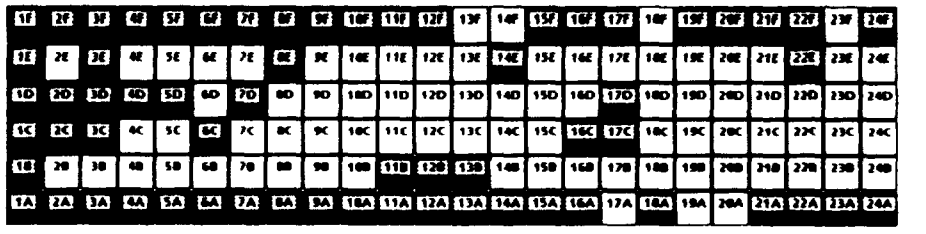


Fig. 18. Rendered seating chart schedule.

$400. The user may simultaneously change the graphical and internal representation of this information by simply mouse-selecting the flight box and moving it to a new location, changing its shading, or increasing or decreasing its height. Casner [6] generalizes this technique in a tool that allows users to create customized diagramming languages that can be attached to and used to manipulate internally stored data and knowledge representation structures.

8.3 Limitations of Automated Rendering

There are several important limitations of the automated rendering component. First, the rendering component is incapable of rendering presentations that make use of domain-specific conventions. For example, airline seating charts typically orient the aircraft vertically with lower numbered seats appearing at the top and higher number seats at the bottom. Since BOZ's rendering component has no knowledge of this convention, the seats are arranged in increasing order from left to right as are the hours along the time scale. Second, many presentations depict realistic information such as spatial arrangements and shapes that do not encode information vital to the task at hand but preserve many features of a real-world artifact in an artificial representation. For example, airline seating charts typically depict the aisle separating the two halves of the plane. Some seating charts also use chair-shaped icons to represent seats instead of the generic box-shape used in BOZ's presentation. BOZ of course has no knowledge of these conventions. Note that despite these two limitations it is still possible to locate any seat. What may be lost is a familiarity and practice that users may have already acquired using other conventions. Third, the art of font, typeface, and color selection falls beyond the scope of BOZ's current design model. Consequently, BOZ chooses types and colors arbitrarily. This limitation is in part due to the lack of theoretical account of how type and color directly affect perceptual task performance. As theory-based typographic and color selection principles become available, they too could be integrated into BOZ's task-analytic design approach.

9. COGNITIVE ANALYSIS OF THE AIRLINE SCHEDULE GRAPHIC

Table VIII summarizes the potential cognitive advantages of the airline schedule graphic. The advantages occur in two forms that agree precisely with the design goals of BOZ: (1) substituting efficient perceptual inferences

Table VIII. Predicted Cognitive Efficiencies of the Airline Graphic

Computation
Substitutes a distance judgement (`determine-horz-pos`) in place of subtracting numerically expressed departure and arrival times (`computeLayover`).
Substitutes a shade judgement (`shaded?`) for reading the words "ok" and "full" (`available?`).
Substitutes judging the combined heights of two flight boxes (`stack-heights`) for adding two numerically expressed costs (`addCosts`).
Search
Eliminates eye movements when looking up time, city, cost, and availability information since this information is represented in the same spatial locality (a single flight box).
Allows users to limit their search for connecting flights to only those flights that appear to the right of the originating flight.
Since shading can be processed pre-attentively, users may immediately exclude from their search any flight square that has no available seats.
Allows users to immediately rule out "tall" flights from their search since these are likely to violate the cost constraint.
When looking for an available seat, users can immediately eliminate shaded seats. Users can also restrict their search to window or aisle seats by following simple eye movement patterns.

in place of more demanding logical inferences, and (2) reducing the number of items considered when searching for needed information. The advantages are explained by comparing how the task is performed using both the graphic presentation and a tabular presentation of the same information.

It is important to note that the hypothesized advantages of any BOZ-designed graphic depend on the user understanding and being able to perform the perceptual procedure supported by that graphic. Whether or not real users can or actually do follow the designed perceptual procedure and the extent to which the predicted efficiency advantages are reflected in users' performance is an empirical question to which we now turn our attention.

10. USERS' PERFORMANCE WITH THE AIRLINE GRAPHIC

Casner and Larkin [8] describe an experimental study designed to determine the extent to which the hypothesized advantages of the airline graphic listed in Table VIII were reflected in users' performance. To better understand the contribution made by each hypothesized advantage, a sequence of four graphics was designed in which each successive graphic provided an additional opportunity to substitute a perceptual for logical operator and an additional opportunity to reduce search. The final presentation in the sequence contained all of the efficiency advantages listed in Table VIII (except those pertaining to the seating chart presentation). The four experimental graphics, herein called Graphics 1, 2, 3, and 4, are shown in Figure 19.

Response times were collected from 7 participants who performed the airline reservation task using the four graphics a total of 10 times each (40 trials total). Users completed the task after receiving an explanation of the conventions used in each graphic and one practice trial. The order in which the graphics were presented to the users was varied systematically to evenly distribute effects due to learning and practice.

The results shown in the graphic in Figure 20 indicate significant differences in response times between Graphics 1 and 2, and between Graphics 2 and 3, but not between Graphics 3 and 4. The data suggest that time scale encoding used in Graphic 2 (and also in Graphics 3 and 4) reduced the amount of time required to locate two connecting flights and to determine whether or not two flights obey the layover constraint. Allowing users to perform the perceptual operator of determining the shade of a flight box (Graphic 3) also resulted in a significant savings. The perceptual task of determining whether or not two flights obey the cost constraint by judging the heights of the flight squares did not result in any reliable savings over the task of adding the two numbers or in narrowing down the search space of flights to consider. An analysis of the standard deviations in response times suggests that users exhibited significantly more stable performance between Graphics 1, 2, and 3 in that order.

Our next step was to understand how users obtained the efficiency savings we observed. We ran a regression analysis on the number of times each operator must be performed using each graphic, the number of facts searched, and the observed users' response times. We obtained the best fitting models when each graphic was combined with the procedures that used all of the operator substitutions and search reductions that were applicable to that graphic. This suggests that for each graphic users took advantage of all of the operator substitutions and search reductions that were possible with that graphic. The beta-coefficients in the regression model yielded estimates on performance time for several of the individual perceptual and logical operators.

—The time required to fix the eye on each item in a graphic was uniformly about 330 ms for all four graphics.

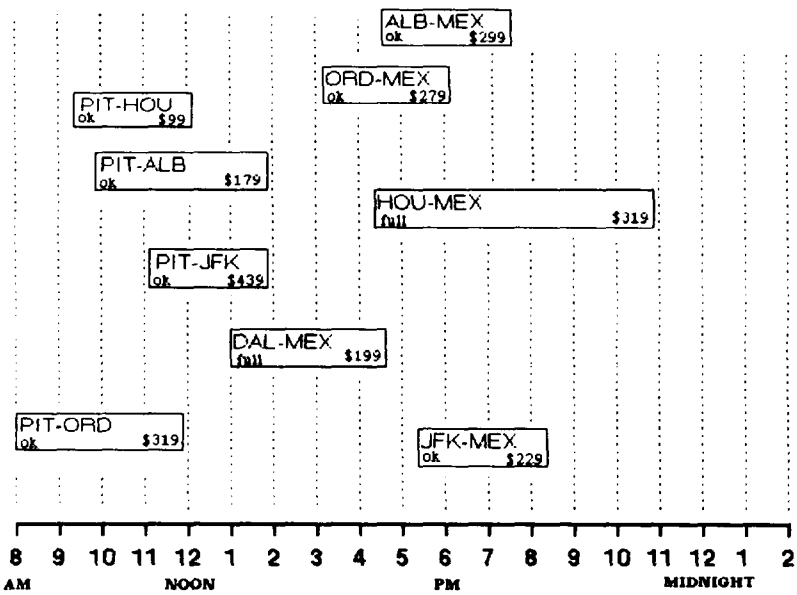
—Perceptually estimating layovers (determine-horz-distance) using Graphics 2, 3 and 4 proceeded about 2 s faster than subtracting the numerically expressed times (computeLayover).

—Judging the combined heights of two flight boxes (stack-heights) in Graphic 4 was negligibly 100 to 300 ms slower than adding the numerically expressed costs (addCosts).

The savings gained through substitution of perceptual for logical operators and use of search reductions match well with the global reductions observed in overall response times. Overall the results agree with users' comments after using all four graphics: Graphics 3 and 4 were the most effective. The interested reader can find details of the experimental design and methodology in Casner and Larkin [8].

Origin/ Destination	Availability	Price	Departure	Arrival
ALB-MEX	ok	$219	1:15pm	7:50pm
PIT-ORD	ok	$129	10:00am	12:00pm
PIT-HOU	full	$199	9:15am	2:05pm
ORD-DAL	full	$199	1:00pm	4:30pm
DAL-MEX	ok	$229	7:30pm	10:30pm
PIT-ALB	ok	$219	8:00am	11:50am
BAL-MEX	full	$279	3:50pm	9:50pm
JFK-MEX	ok	$229	3:00pm	6:00pm
PIT-JFK	full	$239	9:50am	12:30pm

(a)

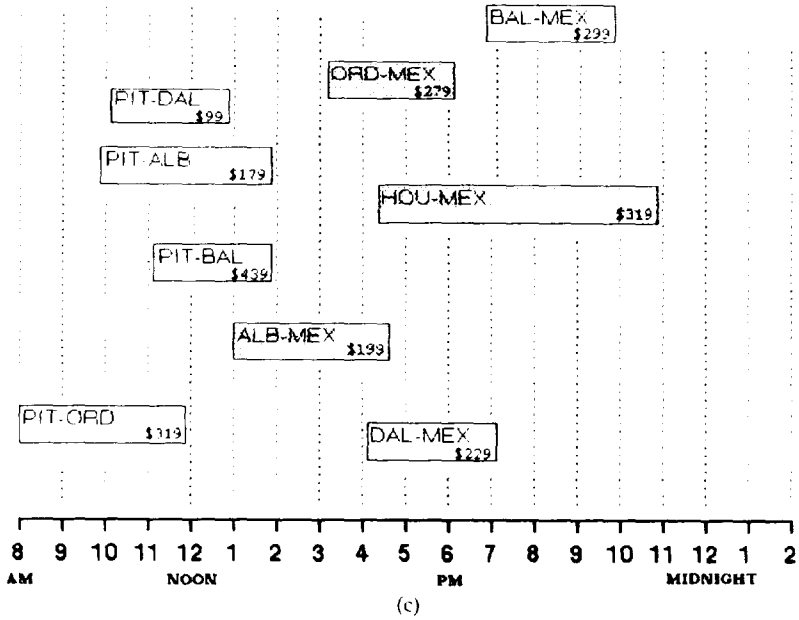


(b)

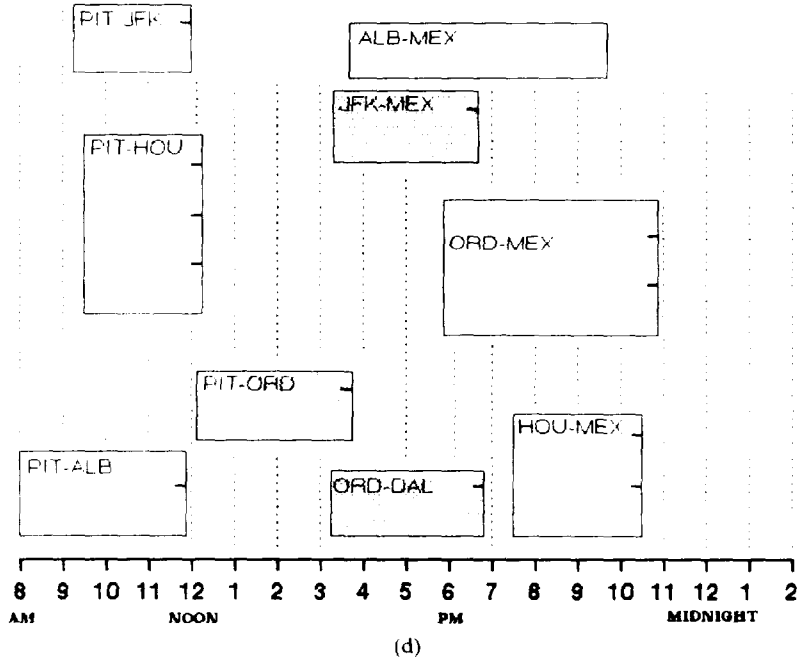
Fig. 19. Four experimental graphics for airline reservation. (a)Graphic. (b) Graphic 2. (c) Graphic 3. (d) Graphic 4.

11. GENERAL DISCUSSION

The research described above explores a task-analytic approach to the design of graphics in which graphic presentations are viewed as perceptually manipulated data structures that help streamline task performance in the same way that abstract data structures and their associated procedures help expedite abstract computational processes. The important distinction made in



(c)



(d)

Fig. 19—*Continued*

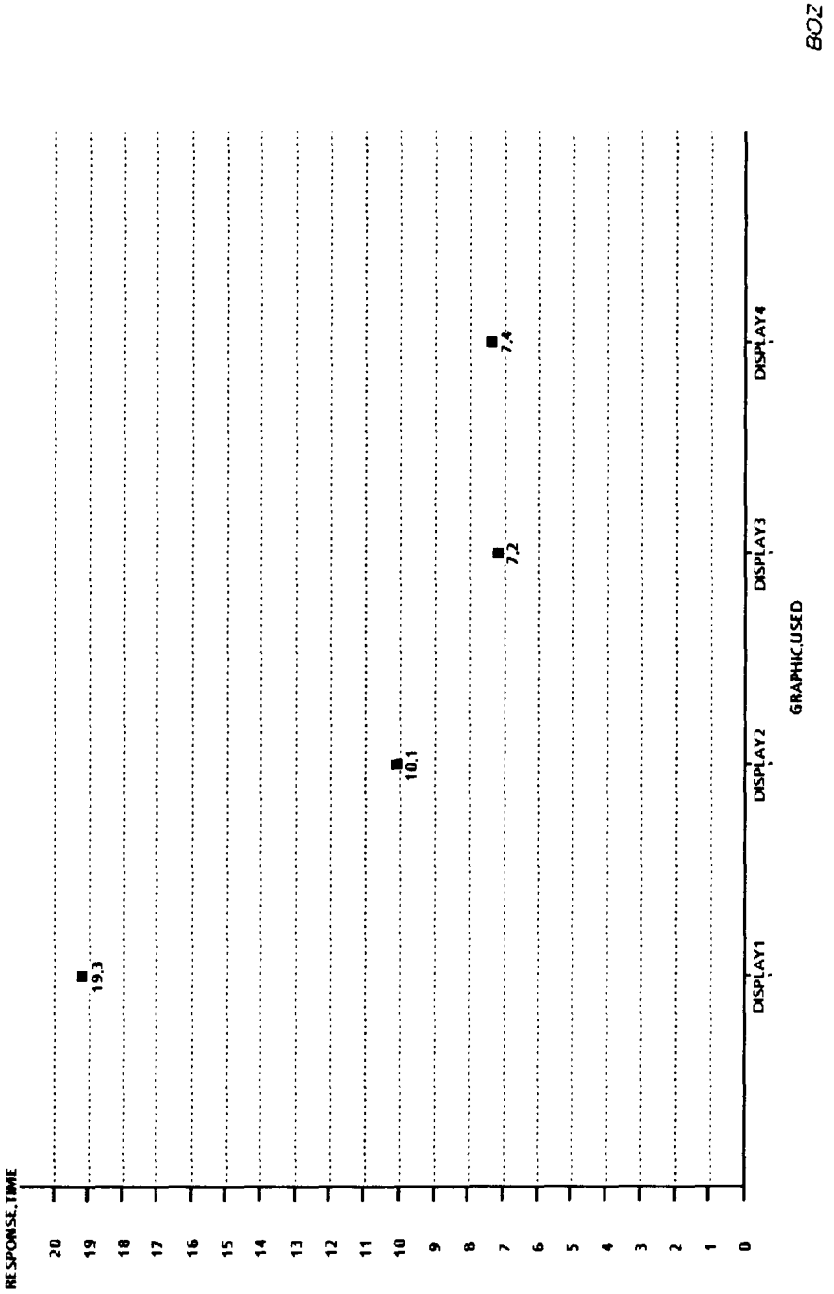


Fig. 20. Participants' mean response times for airline reservation task.

task-based graphic design is that the effective use of perceptual data structures, as with abstract data structures, depends on designing the right structure for a given task. That is, the utility of a presentation is linked to the nature of the task to be supported more than the characteristics of the information to be presented. Consequently, the design methodology embodied in BOZ proceeds by analyzing a user task and generating equivalent perceptual tasks that can be performed more efficiently by humans. The design of an accompanying graphic is targeted primarily at supporting efficient and accurate human performance of the perceptual task. The examples and experimental results show how the task-analytic approach can be successfully applied to designing effective perceptual tasks and presentations that provide two types of advantages: (1) allowing users to substitute quick perceptual inferences in place of more demanding logical inferences, and (2) reducing user's search for needed information.

Design of Effective Graphic Presentations. What are the real graphic design successes of BOZ? Has BOZ produced a graphic that no one else has designed before? Given its current set of perceptual operators and graphical presentation objects it is unlikely that BOZ will produce new graphic designs that differ radically from existing designs. That is, in the examples developed in this paper, no graphic seems to contain a shape or a configuration of shapes that strikes the reader as being completely novel, or having never appeared in a graphic used in one domain or another. This comes as little surprise as the study of novel graphical elements and their mathematical properties enjoys centuries of prior investigation. Rather, the contribution of the present version of BOZ lies in forming compositions of existing designs to arrive at presentations that gather the task-supportive features of several individual designs in order to arrive at "customized" presentations that support the operators in a target user task. It is important to note that the degree of customization achieved by each presentation is more than a simple collection of useful presentations implicated by a task description. Rather, BOZ is additionally able to assign degrees of importance to the individual operators in a task and use this information to make sacrifices in less important aspects of a task to facilitate efficient perceptual processing in more demanding aspects of the same task.

BOZ has thus far been used to design graphics for the five airline reservation tasks given above, a computer operator task, a class scheduling problem, and simple charts and graphs such as those found in the popular literature. The interested reader can consult Casner [7] for these examples along with the complete task descriptions from which the graphics were generated. Future work will attempt to apply BOZ to more challenging information-processing tasks and in a wider variety of task domains. These efforts should lead to a more sophisticated understanding of how the problems of task analysis and graphic design interact, as well as novel ways of extending BOZ's capabilities.

Learning Graphical Conventions and Procedures. An important aspect of the utility of a graphic not addressed by BOZ is the time required to

understand the procedures that must be followed to use that graphic successfully. There seem to be no inherent advantages of learning a perceptual procedure instead of a logical one even if that perceptual procedure is eventually performed more efficiently or accurately. The learning issue is relevant when graphics are presented in "walk up and use" situations where it is unsafe to assume prior knowledge or skill on the part of the user. Learning issues are less important in skilled performance task situations. The idea of using different graphics to support different tasks may further require the user to learn several different sets of graphic conventions. For example, the airline graphics in Figures 1, 2, 3, 4, and 17 all require the user to manipulate the same information in different ways.

Two arguments can be made in defense of BOZ with respect to the learning issue. First, the individual graphical conventions that follow from BOZ's current repertoire of graphical presentation objects and perceptual operators do not extend beyond what is used in popular graphic presentations in current use. Even though each of the airline schedule graphics are somewhat unique, they are all composed using familiar conventions such as aligning data values along horizontal and vertical scales, and color coding. Consequently, the learning requirements for BOZ-designed graphics extend beyond those of existing graphics only in that the user must understand novel combinations of familiar perceptual operators. Second, since there is a cost associated with learning to use any artifact, in each situation we must ask whether or not the artifact offers benefits to the user that justify the learning cost. Taking airline scheduling as a case in point, if the graphical airline schedules result in allowing customers to gain control of their own flight scheduling, the benefits may far outweigh the initial learning costs.

Real-Time Automated Graphical Presentation. Aside from an articulation of a cognitive theory of graphic presentation design, BOZ appears potentially useful as a tool for the automated design and generation of graphic presentations in computer information systems. However, two limitations of the present model prevent BOZ's current use in real-time applications. First, the logical task descriptions required by BOZ as input must presently be hand-generated. A future research topic is to investigate ways of automatically generating task descriptions and eliminating the need for human intervention. SAGE [27] uses a discourse processor that allows descriptions of simple operators to be generated by analyzing simple natural language queries made by the user. However, this approach is unable to generate descriptions of complex procedures defined using collections of many operators. Second, while the run time complexity of BOZ may theoretically be able to meet the demands of on-line information systems, the present implementation fails to produce graphics in a time that would be considered acceptable by computer users. The rendering component is particularly slow for graphics containing many graphical objects. The search complexity for BOZ's perceptual operator substitution component is presently: $T_{\text{operator substitution}} = n*c*t$, where n is the number of logical operators appearing in a procedure, c is the number of possible operator classes that each logical operator must be matched against,

and t is the time required to find all matches for a single operator. The Xerox 1186 implementation of BOZ required about 9 s to classify the airline reservation task operators, as shown in Figure 7. BOZ's perceptual data structuring component is linear in the number of logical operators n and domain sets d: $T_{\text{data structuring}} = n*d$. BOZ required about 2 s to design the initial perceptual data structure shown in Figure 11. The perceptual operator selection component runs n^2 in the number of logical operators. The perceptual operator selection component required 7 s to select the perceptual procedure and data structure shown in Figures 12 and 13. The object-oriented rendering component is linear in the number of structured facts to be presented f and the number of primitive graphical languages p appearing in each structured fact: $T_{\text{rendering}} = f*p$. The rendering component required approximately 12 s to render the flights presentation in Figure 17 and approximately 1 min, 15 s to generate the seating chart presentation in Figure 18.

Overall, BOZ designed both presentations in about 18 s, rendering the flights and seating charts presentations after 30 s and 1 min, 45 s, respectively. BOZ's current run time does not fall within an acceptable standard for real-time data presentation. A future research topic is to investigate ways of making BOZ operate more efficiently.

Executable Logical and Perceptual Procedures. BOZ contains an additional component that allows the logical procedures and the perceptual procedures produced by BOZ to be compiled into executable simulations. These simulations manipulate databases of logical and graphical facts such as those shown in Figures 6 and 16. The simulation component allows alternative procedures to be executed while the number of operator firings and items searched are counted for any combination of logical or perceptual procedure and graphic. These measures can be used to obtain detailed quantitative predictions on the effectiveness of any procedure and graphic produced by BOZ and may avoid the need to perform time-consuming experimental studies with real users such as the one described in Section 10. Casner [7] uses the simulation component to explore other cognitive advantages of graphic presentation-based task performance and problem solving and to better understand the details of how efficiencies in inferencing and search are obtained through the use of graphic presentations.

ACKNOWLEDGMENTS

I thank Jill Larkin, Stellan Ohlsson, Ken Koedinger and three anonymous reviewers for their contributions to this research.

REFERENCES

1. ASCH, S. E. Studies of independence and submission to group pressure: A minority of one against a unanimous majority. *Psychological Monographs*, 1956, 70.
2. BERTIN, J. *Semiology of Graphics*, W. Berg (Transl). Univ. of Wisconsin Press, Madison, 1983.
3. BRACHMAN, R. J., AND SCHMOLZE, J. G. An overview of the KL-ONE knowledge representation system. *Cognitive Sci.* 9 2, (1985), 171-216.

4. BRAINERD, W. S., AND LANDWEBER, L. H. *Theory of Computation*, Wiley, New York, 1974.
5. CARD, S. K., MORAN, T. P., AND NEWELL, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum, Hillsdale, N. J., 1983.
6. CASNER, S. M. Building customized diagramming languages. In *Visual Languages and Visual Programming*, S. K. Chang Ed. Plenum Press, New York, 1990.
7. CASNER, S. M. *Task-Analytic Design of Graphic Presentations*. Ph.D. dissertation, Intelligent Systems Program, Univ. of Pittsburgh, Aug. 1990.
8. CASNER, S. M., AND LARKIN J. H. Cognitive efficiency considerations for good graphic design. In *Proc. 11th Annual Conf. Cognitive Science Society* (Ann Arbor, Mich., Aug. 1989).
9. CLANCEY, S. M., AND HOYER, W. J. Effects of age and skill on domain-specific visual search. In *Proc. 9th Annual Conf. Cognitive Science Society*, (Seattle, Wash., 1987), 398-404.
10. CLEVELAND, W. S. *Elements of Graphing Data*. Wadsworth Advanced Books and Software, Monterey, Calif., 1985.
11. CLEVELAND, W. S., AND MCGILL, R. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *J. Amer. Stat. Assoc. 79*, 387 (Sept. 1984), 531-554.
12. DAVIDOFF, J. B. The role of colour in visual displays. *Int. Rev. Ergonomics 1*, (1987), 21-42.
13. FALLSIDE, D. Understanding machines in motion. Ph.D. dissertation, Dept. of Psychology, Carnegie Mellon Univ., Pittsburgh, Pa., May 1988.
14. FEINER, S. APEX: An experiment in the automatic creation of pictorial explanations. *IEEE Comput. Graph. Appl.* (Nov. 1985), 29-37.
15. FRIEDELL, M. Context-sensitive, graphic presentation of information. *Comput. Graphic. 16* 3, (July 1982), 181-188.
16. GNANAMGARI, S. Information presentation through default displays. Ph.D. dissertation, Univ. of Pennsylvania, May 1981.
17. HEGARTY, M., AND JUST, M. Understanding machines from text and diagram. In *Knowledge Acquisition from Text and Picture*, H. Mandl and J. Levin Eds., North-Holland, Amsterdam, 1988.
18. HUDSON, W. The study of the problem of pictorial perception among accultured groups. *Int. J. Psychology 2* (1968), 89-107.
19. JARVENPAA, S. L., AND DICKSON, G. W. Graphics and managerial decision making: Research Based Guidelines. *Commun. ACM 31*, 6 (June 1988), 764-774.
20. JENKS, C. F., AND KNOS, D. S. The use of shading patterns in graded series. *Ann. Assoc. Am. Geographers 51* (1961) 316-334.
21. KIERAS, D., AND POLSON, P. G. An approach to the formal analysis of user complexity. *Int. J. Man-Mach. Stud. 22* (1985), 365-394.
22. KLAHR, D. Quantification processes. In *Visual Information Processing*, W. G. Chase Ed., Academic Press, Orlando, Fla, 1973, pp. 3-34.
23. KOEDINGER, K. R., AND ANDERSON, J. R. Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Sci. 14*, 4 (1990), 511-550.
24. LARKIN, J., AND SIMON, H. Why a diagram is (sometimes) worth 10,000 words. *Cognitive Sci. 11* (1987), 65-99.
25. LUSK, E. J., AND KERSNICK, M. The effect of cognitive style and report format on task performance: The MIS design consequences. *Manage. Sci. 22*, 3 (1979), 787-798.
26. MACKINLAY, J. "Automating the design of graphical presentations of relational information." *ACM Trans. Graph. 5*, 2 (Apr. 1986), 110-141.
27. ROTH, S. F., MATTIS, J., AND MESNARD, X. Graphics and natural language as components of automatic explanation. In *Architectures for Intelligent Interfaces: Elements and Prototypes*, J. Sullivan and S. Tyler Eds. Addison-Wesley, Reading, Mass., 1989.
28. SCHMID, C. F. *Statistical Graphics: Design Principles and Practices*, Wiley, New York, 1983.
29. SCHNEIDER, W. Training high-performance skills: Fallacies and guidelines. *Hum. Factors 27*, 3 (1985), 285-300.

30. STEVENS, S. S. On the theory of scales of measurement. *Science 103*, 2684 (June 1946), 677-680.
31. TEGHTSOONIAN, J. The judgement of size. *Am. J. Psychology 78* (1965), 392-402.
32. TUFTE, E. R. *Envisioning Information*. Graphics Press, Cheshire, Conn., 1990.
33. TUFTE, E. R. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Conn., 1983.
34. ULLMAN, S. Visual routines. *Cognition 18* (1984), 97-159.
35. YAMADA, H. An analysis of the standard English keyboard. Tech. Rep. 80-11, Dept. of Information Science, Univ. of Tokyo, 1980.
36. ZDYBEL, F., GREENFIELD, N. R., YONKE, M.D., AND GIBBONS, J. An information presentation system. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence* (Aug. 1981), 978-984.

Received May 1989; revised December 1989; accepted September 1990