Automatic Design of Efficient Visual Problem Representations

Stephen M. Casner
NASA Ames Research Center
Mail Stop 262-4
Moffett Field, CA 94035-1000
casner@eos.arc.nasa.gov

Abstract
Automated reasoning about the design of effective visual problem representations is possible when we adopt the view that visual problem representations, along with the perceptual procedures that humans use to manipulate them, can be described using information-processing models of the sort introduced by Newell and Simon (1972). This approach provides us not only with a means of characterizing visual problem representations in a formal syntax but also with a means of automatically mapping between "logical" and "perceptual" problem representations and procedures. An automated system called BOZ is described that begins with a logical problem representation and solution procedure, and generates an informationally-equivalent visual problem representation and procedure that allows the human user to obtain the solution more efficiently. BOZ's representation mapping technique proceeds by: (1) replacing demanding logical inferences in the solution procedure with efficient perceptual inferences; and (2) structuring information in the visual representation such that search is minimized. The extent to which the visual representations and procedures produced by BOZ agree with what users actually see and do is discussed.

Logical and Visual Problem Representations

Logical and visual representations of an airline reservation problem are compared to illustrate the advantages offered by visual problem representations, and to show how both logical and visual problem representations can be characterized using the same sort of information-processing model.

Logical Problem Representations

A *logical problem representation* includes a set of logical facts along with a logical procedure that operates on the facts to solve a stated problem. Figure 1 shows a logical representation for an airline reservation problem posed as follows:

> *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 four hours. Both flights that you choose must be available. The combined cost of the flights cannot exceed $500.*

The logical facts (top of Figure 1) describe relations between the elements of a collection of five domain sets: airline flights, departure and arrival times, costs, and availability. For example, the first fact in the list indicates that the origin of Flight 117 is PIT.

The logical procedure describes a series of search and computation steps (called *logical operators*) to be performed using the set of logical facts. Logical operators occur in two forms. *Search operators* such as Step 1 consider the logical facts one at a time, beginning at the top of the list each time, until a fact is located that satisfies the relation contained in the operator. In Step 1 the search is for a fact that satisfies `(Origin FLIGHT 'pit)`. This operator succeeds on the first fact in the list and the variable `FLIGHT` is instantiated with the value, `flight117`. *Computation operators* perform arithmetic or comparison operations on the values retrieved by search operators. Step 11 is a computation operator that subtracts `departure` from `arrival` and instantiates `LAYOVER` with the result. Computation operators performed with a logical representation are assumed to be mentally performed arithmetic or logical operations applied to a collection of data values stored in the user's memory.

Visual Problem Representations

A visual problem representation also consists of a collection of facts accompanied by a procedure for finding a solution to the problem. Facts in a visual problem representation, as illustrated in Figure 2, have two unique features. First, facts in a visual problem representation are encoded graphically by associating the elements of each domain set of information with the discriminable values of some graphical dimension, a technique first proposed by Mackinlay (1986). For example, each airline flight is represented by a rectangle. The horizontal position of the ends of the rectangle encode the departure and arrival times of that flight. The shading of each rectangle encodes the availability of that flight . Second, facts in a visual problem representation group together related information making all information required to draw a particular inference available at the same spatial location. For example, all information about each flight, origin, destination, cost, times, and availability, is grouped together in a single rectangle.

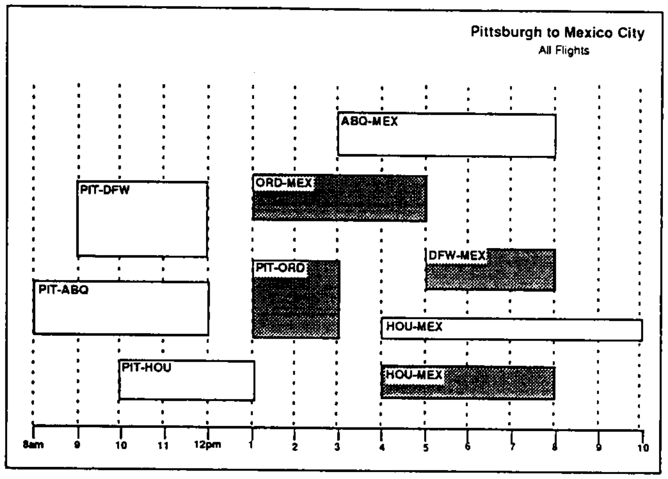
Perceptual procedures differ from logical procedures in that they are composed of a set of *perceptual operators* that describe search and computation steps performed specifically within the context of a visual representation. The perceptual operators that appear in the perceptual procedure allow the user to obtain the same results as do the logical operators that appear in the logical procedure since they compute the same functions only in a different way. The bottom of Figure 2 shows a perceptual procedure for the airline reservation problem.

The visual problem representation is arguably more efficient than the logical problem representation. The advantages of the visual problem representation appear to center around three notions demonstrated by Larkin and Simon (1987) and Casner (1990): (1) perceptual operators

```
(Origin flight117 pit)              (Cost flight117 179)
(Origin flight738 pit)              (Cost flight738 219)
(Origin flight839 pit)              (Cost flight839 319)
(Origin flight733 pit)              (Cost flight733 339)
(Origin flight938 hou)              (Cost flight938 199)
(Origin flight911 ord)              (Cost flight911 229)
(Origin flight912 hou)              (Cost flight912 119)
(Origin flight192 abq)              (Cost flight192 279)
(Origin flight120 dfw)              (Cost flight120 229)
(Destination flight117 hou)         (DepartureTime flight117 10)
(Destination flight738 abq)         (DepartureTime flight738 8)
(Destination flight839 dfw)         (DepartureTime flight839 9)
(Destination flight733 ord)         (DepartureTime flight733 13)
(Destination flight938 mex)         (DepartureTime flight938 16)
(Destination flight911 mex)         (DepartureTime flight911 13)
(Destination flight912 mex)         (DepartureTime flight912 16)
(Destination flight192 mex)         (DepartureTime flight192 15)
(Destination flight120 mex)         (DepartureTime flight120 17)
(Availability flight117 ok)         (ArrivalTime flight117 13)
(Availability flight738 ok)         (ArrivalTime flight738 12)
(Availability flight839 ok)         (ArrivalTime flight839 12)
(Availability flight733 full)       (ArrivalTime flight733 15)
(Availability flight938 full)       (ArrivalTime flight938 20)
(Availability flight911 full)       (ArrivalTime flight911 17)
(Availability flight912 ok)         (ArrivalTime flight912 22)
(Availability flight192 ok)         (ArrivalTime flight192 20)
(Availability flight120 full)       (ArrivalTime flight120 20)
```

```
1  while findFlightWithOrigin(Origin FLIGHT 'pit) do
2    if findAvailibility?(flight, 'ok) then
3      findDestination(flight, LAYOVERCITY)
4      determineArrivalTime(flight, ARRIVALTIME)
5      while findFlightWithOrigin(CONNECTING, layovercity) do
6      if available?(connecting = 'ok) then
7        determineDepartureTime(connecting, DEPARTURETIME)
8        if connecting?(departuretime > arrivaltime) then
9          findDestination(flight, FINALDESTINATION)
10         if landsInDestinationCity?(finaldestination = 'mex) then
11           computeLayover(departure - arrival = LAYOVER)
12           if layoverLessThanX?(layover < '4) then
13             determineCost(flight, COST1)
14             determineCost(connecting, COST2)
15             addCosts(cost1 + cost2 = TOTAL)
16             if costLessThanX?(total < '500) then
                 STOP
```

Figure 1: Logical Problem Representation



```
1  while  searchObjectWithLabel(FLIGHT, 'pit) do
2         if shaded?(flight = 'ok)  then
3             readLabel(flight, LAYOVERCITY)
4             ~~determineHorzPosition(flight, ARRIVAL)~~
5             while   searchObjectWithLabel(CONNECTING, layovercity)
6                     shaded?(connecting = 'ok)
7                     ~~determineHorzPosition(connecting, DEPARTURE)~~
8                     rightOf?(departure,arrival)
9                     readLabel(flight, FINALDESTINATION)
10                    if   sameLabels?(finaldestination = 'mex) then
11                        determineHorzDistance(departure - arrival = LAYOVER)
12                        if   leftOf?(layover < '4) then
13                            ~~determineHeight(flight, COST1)~~
14                            ~~determineHeight(connecting, COST2)~~
15                            stackHeights(cost1 + cost2 = TOTAL)
16                            if   shorter?(total, '500)
```

Figure 2: Visual Problem Representation

are sometimes performed more efficiently than logical operators; (2) operators in a perceptual procedure can sometimes be omitted; and (3) the graphical structuring of the information expedites search for needed information. Several instances of these advantages are apparent in the visual airline representation. First, the visual representation allows the user to substitute a quick distance judgement (`determineHorzPos`) in place of subtracting numerically expressed departure and arrival times (`computeLayover`). Second, Steps 4 and 7 can be omitted since the horizontal distance between two rectangles can be determined without knowing their exact horizontal positions. This savings corresponds to being able to subtract two numbers without knowing what the numbers are. The same reasoning applies to Steps 13 and 14. Third, the visual representation 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). Fourth, it allows users to limit their search for connecting flights to only those flights that appear to the right of the originating flight. Fifth, since shading can be processed pre-attentively, users may immediately exclude from their search any flight square that has no available seats. Sixth, users can immediately rule out "tall" flights from their search since these are likely to violate the cost constraint.

The goal in designing a good visual problem representation, then, is to begin with a logical problem representation, and perceptually encode and organize information so that an efficient perceptual procedure can be used to solve the problem. The following shows how this can be accomplished.

Design of Efficient Visual Problem Representations

BOZ is an automated tool that derives efficient visual problem representations from more demanding logical problem representations. BOZ requires as input a logical problem representation like the one shown in Figure 1: a set of logical facts, and a logical procedure that uses the facts to solve the problem. BOZ works in two steps, designing a perceptual procedure that allows the user to solve the same problem more efficiently, and then using this procedure to decide how the information should be graphically encoded and structured to best support the perceptual procedure. Thus BOZ delivers two things: a visual display and a perceptual procedure that describes how the display can be used to solve the problem.

Designing the Visual Procedure

BOZ derives perceptual procedures from logical procedures by considering each of the logical search and computation operators in the logical procedure and attempting to locate perceptual search and computation operators that give the user the same result. BOZ accomplishes this by searching a catalog of perceptual operators organized around the graphical dimensions used to encode information in a visual display. The perceptual operators used in the visual airline reservation procedure in Figure 2 illustrate the idea: searching for an object of a particular shading, estimating the horizontal distance between two graphical objects, comparing the size of two objects, etc. A perceptual operator qualifies as a legal substitution for a logical operator just when the two operators can be shown to be renamings of one another. For example, since we can create a one-to-one mapping between the arguments in the `findFlightWithOrigin` and the `searchObjectWithLabel` operators, we conclude that the perceptual operator can provide the same information as the logical operator.

It is often the case that more than one perceptual operator qualifies as a legal replacement for a given logical operator. For example, the perceptual operator `searchObjectWithHeight(RECTANGLE, cm)` would also qualify as a substitute for the `findFlightWithOrigin` operator since we can derive the one from the other by renaming. The choice of which particular perceptual operators to choose is guided by a two-tier ranking scheme that prioritizes perceptual operators in terms of their estimated human performance costs. The first tier prioritizes the perceptual operators by the difficulty of the function they compute (e.g., addition is more difficult than search which is more difficult than a comparison). The second tier orders the operators by differences in performance costs for operators that compute the same function (e.g., searching for an object of a particular color is easier than searching for an object of a particular size). When the perceptual operator replacement step is finished, the logical procedure (Figure 1) is transformed into a perceptual procedure (Figure 2). Perceptual search operators whose only purpose is to feed values to a computation operator can generally be skipped and are marked accordingly.

Designing the Visual Display

In two steps, BOZ derives the visual display using the perceptual procedure it has produced. First, by examining the perceptual operators chosen to manipulate each domain set of information, BOZ determines how each domain set of information is to be represented in the display. For example, the `determineDepartureTime` operator manipulates information about flights and their departure times. Since BOZ has decided that the best substitute for this logical operator is the `determineHorzPos` perceptual operator, BOZ is constrained to represent departure times as graphical objects meaningfully positioned along the horizontal axis of the display. Similarly, since the step of determining availability has been replaced by determining the shading of a graphical object, availability must be represented as shadings. Second, BOZ examines the relationships between operators in terms of the domain sets of information they manipulate to determine how related information should be grouped together to support efficient performance of each perceptual operator. For example, since the domain sets describing times, cities, costs, and availability are all used with the domain set `flight`, BOZ attempts to collocate all domain sets in a single graphical object (i.e., rectangle). Once the graphical objects have been designed BOZ translates the original set of logical facts to a set of visual facts and renders them on the computer screen using a technique first proposed by Mackinlay (1986). Fact translation is acheived by replacing the names and members of each domain set of logical information with the names and members of the graphical domain that BOZ has chosen to represent it. The final product is the perceptual procedure and display shown in Figure 2. The interested reader is referred to Casner (1991) for a more complete description of how BOZ works and the general problem of reasoning about logical and visual representations automatically [Mackinlay, 1986].

Psychological Validity of BOZ's Visual Problem Representations and Procedures

The following illustrates two ways in which the perceptual procedures that people follow using a BOZ-designed representation can differ significantly from the perceptual procedure generated using BOZ's one-to-one representation mapping approach.

A variety of different perceptual procedures can be derived by reordering the operators in the perceptual procedure. For example, the user, after locating two connecting flight rectangles, might check the heights of the rectangles (Step 15) prior to checking the distance between the rectangles (Step 11). In general, whenever the arguments to a series of operators do not depend on one another, the operators can be reordered to arrive at a variation on the original procedure. This observation does not pose a problem to BOZ since it is straightforward to enumerate all possible perceptual procedures derivable through operator reordering by examining the data dependencies between operators and generating that subset of the n! permutations that do not violate the dependencies. Moreover, the fact that we are unable to know in advance which orderings people will actually use presents no problem since there are in general no efficiency gains to be had through operator reordering alone. Thus, with respect to operator reordering, the perceptual procedure produced by BOZ is guaranteed to be optimally efficient.

A second, more dramatic way in which the perceptual procedures that people use can differ from those generated by BOZ occurs when users perceptually restructure the visual data itself. Visual problem representations offer the user the opportunity to parse the data along visual dimensions other than those used to create the representation. For example, the flight rectangles representation was created using a single entity, a rectangle, that corresponds to the logical entity that appears in the logical facts: a flight. However, the user is not perceptually limited to this ontology at all. Rather, they are free to carve up the visual representation in any manner that helps them solve the problem, and are furthermore free to invent novel procedures of a fundamentally different character that help them manipulate the data in their newly imagined form. For example, when searching for a pair of flights having a short layover the user can structure the display around the set of regions defined by the space in between the flight boxes shown in Figure 2. Using this perceptual restructuring, the user can guide his or her search using the area between the flights which forms somewhat of a number 6 in the representation in Figure 2. In order to find two flights having a short layover the user can consider pairs of flights that lie at the ends of the narrowest part of this area. For example, the user might immediately find the PIT-HOU-MEX connection since the PIT-HOU and HOU-MEX flight appear at the narrowest part of the intermediate region.

It is important to note that perceptual procedures derived through restructuring fall well beyond BOZ's capability for reasoning about them. Not only is BOZ unable to enumerate the kinds of procedures obtainable through restructuring but is also unable to know anything about how efficient they may be. While I have not undertaken a systematic study of perceptual restructurings nor of the efficiency advantages they yield, it is likely that many potential designs passed up by BOZ for a particular problem are in fact more efficient than the one it produces using its crude one-to-one representation mapping technique. Whether or not the possible perceptual restructurings can be enumerated for a given representation is an open question.

Since the logical and visual problem representations are expressed using the same formal notation, the alternative representations can be used directly as simulations to make comparisons regarding their relative efficiency. Casner and Larkin (1989) describes how these simulations can be used, in combination with subjects' performance time data for the airline reservation problem developed throughout the paper, to empirically determine to what extent people actually follow the hypothesized perceptual procedures.

Summary

A unifying framework was presented that allows logical and visual representations and the procedures that manipulate them to be represented using the same notation. An automated technique was demonstrated that allows efficient visual problem representations to be derived through transformation of an equivalent logical problem representation. Three forms of efficiency that visual representations appear to offer were examined. Two ways were discussed in which the visual problem representations and procedures that people use could differ from those produced using the automated technique.

References

Casner, S. M. (1991). A task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics* 4, April 1991.

Casner, S. M. (1990). *Task-Analytic Design of Graphic Presentations.* PhD Thesis, Intelligent Systems Program, University of Pittsburgh.

Larkin, J. H. and Simon, H. A. (1987). Why a diagram is (sometimes) worth 10,000 words. *Cognitive Science, 11*, 65-99.

Mackinlay, J. D. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics, 5 (2)*, 110-141.

Newell, A., and Simon, H. A. (1972). *Human Problem Solving.* Englewood Cliffs, NJ: Prentice-Hall.