Automating the Design of Graphical Presentations of Relational Information

JOCK MACKINLAY
Stanford University

The goal of the research described in this paper is to develop an application-independent presentation tool that automatically designs effective graphical presentations (such as bar charts, scatter plots, and connected graphs) of relational information. Two problems are raised by this goal: The codification of graphic design criteria in a form that can be used by the presentation tool, and the generation of a wide variety of designs so that the presentation tool can accommodate a wide variety of information. The approach described in this paper is based on the view that graphical presentations are sentences of graphical languages. The graphic design issues are codified as expressiveness and effectiveness criteria for graphical languages. Expressiveness criteria determine whether a graphical language can express the desired information. Effectiveness criteria determine whether a graphical language exploits the capabilities of the output medium and the human visual system. A wide variety of designs can be systematically generated by using a composition algebra that composes a small set of primitive graphical languages. Artificial intelligence techniques are used to implement a prototype presentation tool called APT (A Presentation Tool), which is based on the composition algebra and the graphic design criteria.

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*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*device independence*; *ergonomics*

General Terms: Algorithms, Design, Human Factors, Languages, Theory

Additional Key Words and Phrases: Automatic generation, composition algebra, effectiveness, expressiveness, graphic design, information presentation, presentation tool, user interface

1. INTRODUCTION

Computer-based information plays a crucial role in our society. As a result, an important responsibility of a user interface is to make intelligent use of human visual abilities and output media whenever it presents information to the user. For example, a color medium makes it possible to use graphical techniques based on the fact that the human visual system is very effective at distinguishing a small number of distinct colors [13, 23]. A monochrome medium requires other graphical techniques that utilize other capabilities of the human visual system.

This work was supported in part by grant N00014-81-K-0004 from the Office of Naval Research.
Author's current address: Xerox PARC, 3333 Coyote Hill Rd., Palo Alto, CA 94304.

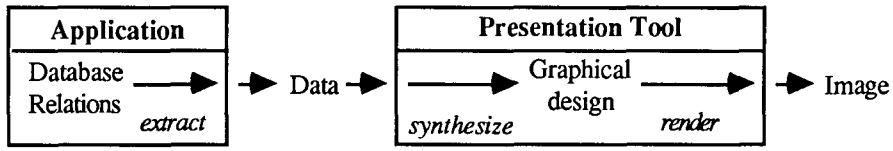

Fig. 1. A linear model for generating presentations. This simplified model, which does not include feedback loops that are required for difficult design problems, describes the fundamental process of generating a graphical presentation. A graphical design synthesized by a presentation tool describes the basic structure and meaning of a graphical presentation. The rendering process fills in the details that are required to form the final image.

Building user interfaces that intelligently present information is a difficult task. At the current time, application designers are forced to anticipate every presentation situation that might arise in an application and decide which graphical techniques are most effective in each situation. Not only do application designers have to "predesign" the presentations, but they must be graphic design experts to ensure that the resulting presentations are effective.

An obvious solution is to build a system, called a presentation tool, that automatically designs graphical presentations of information. Using such a system, application designers need not predesign the presentations, and the graphic design issues are the responsibility of the presentation tool. Figure 1 illustrates how application designers would use such a tool. The application extracts some information from its database (perhaps using statistical analysis). The presentation tool then synthesizes a graphical design and renders an image that presents this information. A *graphical design* is an abstract description of an image that indicates the graphical techniques (such as color variation or position on an axis) that are used to encode information.

There are two open problems that must be solved before such a presentation tool can be constructed: Graphic design criteria must be codified before the presentation tool can synthesize effective designs, and a wide variety of designs must be generated before the presentation tool can handle a wide variety of input.

This paper describes research that begins to solve these problems by focusing on automating the design of two dimensional (2-D) static presentations (such as bar charts, scatter plots, and connected graphs) of relational information. The cornerstone of this research is the development of precise definitions of graphical languages that describe the syntactic and semantic properties of graphical presentations. The framework established by these definitions addresses the two problems mentioned above. Graphic design issues are codified with expressiveness and effectiveness criteria. *Expressiveness criteria* identify graphical languages that express the desired information. *Effectiveness criteria* identify which of these graphical languages, in a given situation, is the most effective at exploiting the capabilities of the output medium and the human visual system. A wide variety of designs is systematically generated using a *composition algebra*, which is a collection of primitive graphical languages and composition operators that can form complex presentations. This framework is implemented with artificial intelligence techniques. A prototype application-independent presentation tool called APT (A Presentation Tool) has been built. Even though only the basic

framework has been implemented, APT can synthesize a wide variety of useful designs.

The paper is a top-down description of the two types of graphic design criteria and the composition algebra mentioned above. Related work is described in Section 2. Section 3 uses three examples to motivate the development of the criteria and algebra. Section 4 gives a detailed overview of the results described in this paper. The core of the paper is contained in Sections 5–7, which describe the details of the expressiveness and effectiveness criteria and the composition algebra. Sections 8 and 9 describe how APT uses these results to synthesize presentations. Finally, Section 10 considers how the research can be extended.

2. RELATED WORK

The automatic design of graphical presentations of information is a relatively unexplored research area. As a result, existing work has focused on restricted aspects of the problem. Aside from the early unfocused work, the following three foci categorize the existing work: content issues, graphic design issues, and design variation issues. Content issues are central to systems that automatically determine the content of presentations, such as adding or removing details to generate an effective presentation or developing a sequence of related presentations. The graphic design and design variation issues have already been described. Another useful categorization is the graphical techniques used by the system (2-D, 3-D, animation).

Two of the early, less focused pieces of work deserve mention. The first developed the AIPS system, which was one of the earliest attempts to separate the presentation process from the rest of an application [24]. AIPS used the KLONE representation system to specify and refine a high-level specification of a 2-D information display. The second piece of early work studied automatic animation scripting and the rule-based layout of node-link diagrams [12].

Content issues were the primary focus for the work on two systems. The first was the VIEW system developed by Friedell, which automatically generated 2-D icons describing the properties of ships stored in a naval database [10]. A stepwise refinement of icon templates, using subicon templates, terminated when sufficient detail was generated for a given icon size. The templates were designed by hand rather than by the system. The second was the APEX system developed by Feiner, which automatically generated a sequence of images that describe actions in a 3-D world consisting of some sonar cabinets [8]. The system carefully tailored the sequence of images to omit irrelevant or redundant details. The graphic design issues surrounding the merging of icons and 3-D images were also considered.

Two other pieces of work focused on graphic design issues. The first was the BHARAT system developed by Gnanamgari, which was an early effort at the automatic generation of 2-D presentation graphics [11]. It selected a pie chart, bar chart, or line chart design for a single unary function, which could have multiple numeric ranges. BHARAT was based on a simple design algorithm. When the function was continuous, a line chart design was used. When the user indicated that the range sets could be summed to a meaningful total, a pie chart design was used. Bar chart designs were used in the remaining cases. Although

multiple designs were generated, BHARAT's range of designs was limited. It could not present a collection of relations or nonfunctional relations. Gnanamgari's discussion focused on graphic design issues. However, her effectiveness criteria about issues such as font and color choice were "wired" into the code that rendered the design, making it difficult to extend the system to a broader range of input.

The second piece of work that focused on graphic design issues was Beach's system, which automated the low-level layout and design of 2-D tables whose high-level topology was specified by the user as a matrix of rows and columns [1]. This research utilized a specification called a *graphical style* [2], which allowed the explicit description of the graphic design properties of the table, such as line widths, background tints, and size constraints. As a result, the user could control parts of the design while the remainder was controlled by the existing default style. Explicit graphical styles made it possible for the system to format the table in different ways for different output media. Although the graphical styles had to be specified by hand, care was taken to make sure that graphic design issues could be addressed in the specification.

Design variation and graphic design are the primary foci of the work described in this paper. This work differs from the previous work in that it focuses directly on the generation of a comprehensive variety of designs for 2-D static presentations of relational information. The APT system uses artificial intelligence techniques to implement a "generate and test" search for a design: The composition algebra generates design alternatives, and the graphic design criteria test the generated alternatives.

3. THE GRAPHICAL PRESENTATION PROBLEM

The graphical presentation problem is to synthesize a graphical design that expresses a set of relations and their structural properties effectively. This problem is illustrated in this section by three examples that describe the desired behavior of a presentation tool. These examples describe the basic concerns that led to the criteria and algebra discussed in the remainder of the paper.

Given the process model in Figure 1, the examples of presentation tool behavior begin with the application's database. Figure 2 describes a collection of relation tuples about automobiles that might be contained in such a database. The structural properties of these relations, which might be in the database schema, are shown in Figure 3 using standard database notation [22]. Structural properties include the domain sets and their functional relationships. Given such a database, a typical input from the application to the presentation tool would be the following:

Present the Price and Mileage relations.
The details about the set of Cars can be omitted.

Note that the application can include additional requests, such as asking that the *Cars* details be omitted.

Given this input, a presentation tool produces two outputs: a graphical design and an image rendered from that graphical design. A graphical design, which is the primary concern of this paper, consists of a set of encoding relations between

Price(Accord, 5799)	*Price*(AMC Pacer, 4749)
Mileage(Accord, 25)	*Mileage*(AMC Pacer, 17)
Weight(Accord, 2240)	*Weight*(AMC Pacer, 3350)
Repair(Accord, Great)	*Repair*(AMC Pacer, Terrible)
Nation(Accord, Germany)	*Nation*(AMC Pacer, USA)
Price(Audi 5000, 9690)	*Price*(BMW 320i, 9735)
⋮	⋮

Fig. 2. Relation tuples about 1979 automobiles. This is an example of a table of relation tuples that might be generated by a database system in response to a query. A presentation tool can do much better than this.

Price: *Cars* → [3500, 13000]
Mileage: *Cars* → [10, 40]
Weight: *Cars* → [1500, 5000]
Repair: *Cars* → ⟨Great, Good, OK, Bad, Terrible⟩
Nation: *Cars* → {USA, Germany, France, ...}
Cars = {Accord, AMC Pacer, Audi 5000, BMW 320i, ...}

Fig. 3. Structural properties of the automobile relations. The arrow (→) indicates a functional dependency between domain sets. The square brackets ([ ]) describe domain sets that are quantitative ranges, the angle brackets (⟨ ⟩) describe domain sets that are ordered sets, and the curly braces ({ }) describe domain sets that are unordered sets.

a graphical image and the information it presents. For example, Figure 4 describes a scatter plot design for the price/mileage input. Graphical objects, such as points and line segments, encode the domains of the relations. Properties of those objects, such as their position, encode the functional information. The *Encodes* predicate is used to indicate these encoding relationships for both the graphical objects and their properties. Figure 5 contains the scatter plot image that APT rendered from this design.[1]

The following three price/mileage examples illustrate the expressiveness, effectiveness, and design variation concerns mentioned above. The scatter plot in Figure 5 illustrates the importance of the expressiveness concern. If the application had not requested that the details about the cars be omitted, the scatter plot in Figure 5 would not have expressed all the input. Without this request, the scatter plot would have had to include labels, as shown in Figure 6. These scatter plot labels, however, illustrate the importance of the effectiveness concern. The labels obscure the mark positions, and it is difficult to find individual cars. If the details about the cars are important, there are more effective design alternatives. For example, when the details of the cars are to be presented, the presentation tool should synthesize the aligned bar chart design

[1] In this paper, an *apt* in the lower right corner of a figure indicates that APT designed and rendered the diagram. Such diagrams are intended to illustrate APT's ability to deal with coarse-grained graphic design issues, such as the choice of the graphical techniques to encode information. Fine-grained rendering issues, such as line width, font choice, and precise graphical object placement are not a focus of this research. A production presentation tool will certainly be able to generate more graphically interesting diagrams than the ones shown here.

Encodes(*VertAxis*, [3500, 13000], *ScatterPlot*)
Encodes(*HorzAxis*, [10, 40], *ScatterPlot*)
Encodes(*Points*, *Cars*, *ScatterPlot*)
Encodes(*Position*(*Points*, *VertAxis*), *Price*(*Cars*), *ScatterPlot*)
Encodes(*Position*(*Points*, *HorzAxis*), *Mileage*(*Cars*), *ScatterPlot*)

Fig. 4. The graphical design for a scatter plot of the price mileage input. The *Encodes* relation indicates the relationship between graphical objects or properties and the information encoded. For example, the first line says that the vertical axis encodes the range of prices, and the fourth line says that the position of the points on the vertical axis encodes the prices of cars. The input relations are written as functions to simplify the description.

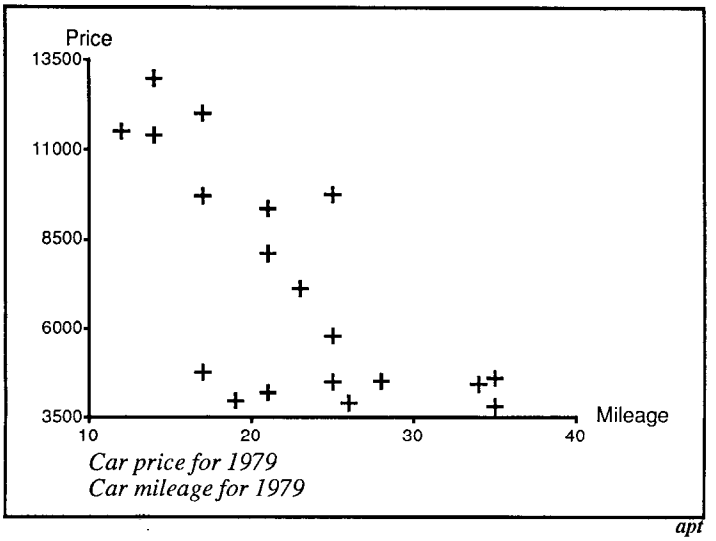

Fig. 5. Scatter plot of the price/mileage input. The graphical design for this image is in Figure 4. The design expresses the relations only if the application permits the details about the cars to be omitted. The *apt* in the lower right corner indicates that APT designed and rendered this diagram.

used to render Figure 7, rather than the scatter plot design. The aligned bar chart design makes it easy to find values associated with an individual car. The existence of this alternative design illustrates the importance of the design variation concern. A presentation tool should be able to generate an expressive and effective design for each presentation situation.

The following rough estimate indicates that there are many possible inputs to a presentation tool. The price/mileage input consists of two binary relations that share the same first domain set and are functional dependencies from a qualitative domain to a numeric range. The structural properties of the input relations are important factors for designing a graphical presentation. Different structural properties require the presentation tool to synthesize different designs. Furthermore, the ability to vary these properties independently leads to a combinatorial

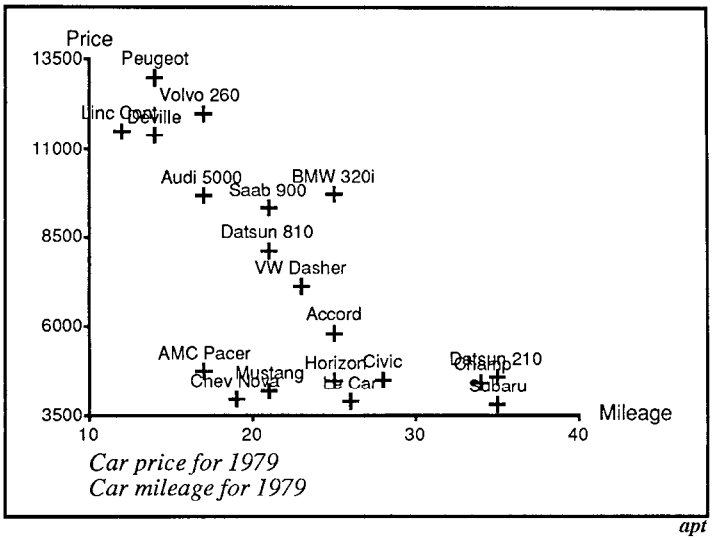

Fig. 6. Labeled scatter plot for the price/mileage input. Although a more sophisticated rendering could avoid the overlapping of the labels, two basic problems of a labeled scatter plot design reduce its effectiveness. First, labels make it difficult to perceive the positions of the points. Second, a given label is difficult to find.

explosion in the number of inputs that might be given to a presentation tool. To see this, abstract the two binary relations by ignoring the functional dependencies, the sharing of domain sets, and the properties of the domain sets:

$$\begin{array}{ll} & \overbrace{\quad\quad}^{d} \\ R: & A \quad B \\ S: & C \quad D \end{array} \Big\} r.$$

Given that there are on the average d domain sets for each relation and r relations in the input, there are

$$(2^d - 1)^r \times (dr)! \times 3^{dr}$$

different possible design problems. The $(2^d - 1)^r$ factor is based on the fact that each relation can be a functional dependency from 1 through d domain sets to the remaining domain sets and is equivalent to the number of nonempty subsets of the set of domain sets of the relation. The $(dr)!$ factor is based on the number of canonical permutation cycles that can be formed by the sharing of all the domain sets [14]. The 3^{dr} factor is based on the fact that each domain set can be one of the following three types [20]: A domain set is *nominal* when it is a collection of unordered items, such as {`Jay, Eagle, Robin`}. A domain set is *ordinal* when it is an ordered tuple, such as ⟨`Monday, Tuesday, Wednesday`⟩. A domain set is *quantitative* when it is a range, such as [`0, 273`].

The preceding formula indicates that there can be many inputs to the presentation tool. For two binary relations, there are over 17,000 possibilities. However,

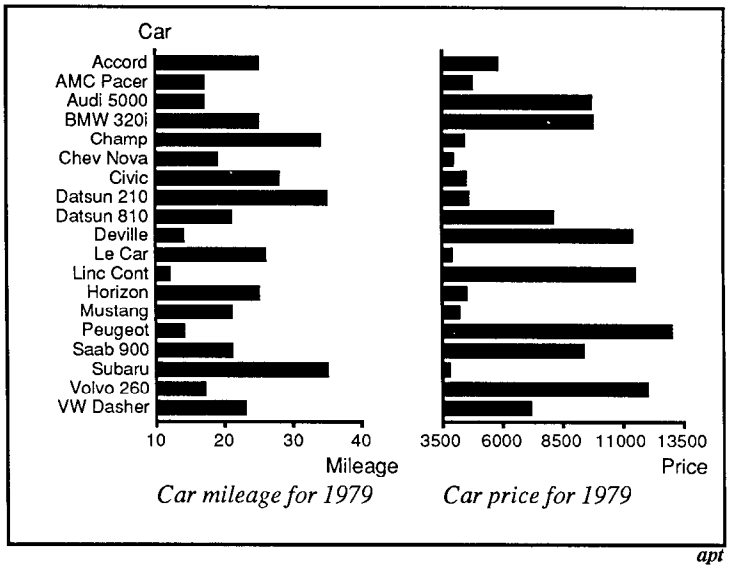

Fig. 7. Aligned bar chart for the price/mileage input. This diagram shows the detailed properties of the cars better than a scatter plot. However, the general relationships are not so easy to see.

the input can include more relations or relations with more domain sets. For example, four binary relations (similar to the automobile data presented in Figure 30) can generate over 21 billion different inputs. Furthermore, this estimate does not include other factors that increase the number of designs that must be generated by a presentation tool, such as application requests or properties of the output media.

4. APPROACH

The fundamental assumption of the approach described in this paper is that graphical presentations are sentences of graphical languages, which are similar to other formal languages in that they have precise syntactic and semantic definitions. The three concerns described in the previous section are handled by a careful analysis of the properties of these definitions. This analysis leads to expressiveness and effectiveness criteria for evaluating graphical designs and a composition algebra for generating design alternatives.

An expressiveness criterion, which is derived from a precise language definition, is associated with each graphical language. A graphical language can be used to present some information when it includes a graphical sentence that expresses *exactly* the input information, that is, all the information and only the information. Expressing additional information is potentially dangerous because it may not be correct.

Effectiveness criteria can be based on a number of different factors. For example, a design can be judged effective when it can be interpreted accurately or quickly, when it has visual impact, or when it can be rendered in a

Marks:	Points, lines, and areas
Positional:	1-D, 2-D, and 3-D
Temporal:	Animation
Retinal:	Color, shape, size, saturation, texture, and orientation

Fig. 8. Bertin's graphical objects and graphical relationships.

cost-effective manner. This paper concentrates on generating designs that can be accurately interpreted. Dealing with multiple, perhaps conflicting, effectiveness criteria is beyond the scope of this research.

Given the focus on accuracy, effectiveness criteria are based on the observation that a graphical language uses specific graphical techniques to encode information. When interpreting a graphical sentence, a person is confronted with perceptual tasks that correspond to these graphical encoding techniques. Since some perceptual tasks are accomplished more accurately than others, effectiveness criteria can be based on the comparison of the perceptual tasks required by alternative graphical languages.

Since most graphical presentations of relational information are based on a general vocabulary of graphical techniques, a wide variety of designs can be generated with a composition algebra that describes this graphical vocabulary. Figure 8 summarizes graphic designer Jacques Bertin's vocabulary of the graphical techniques commonly used to encode information in presentation graphics [3]. Graphical presentations use graphical marks, such as points, lines, and areas, to encode information via their positional, temporal, and *retinal* properties.[2] The composition algebra consists of a *basis set* that contains primitive graphical languages, each of which embodies one of Bertin's graphical techniques for encoding information, and *composition operators* that are able to generate a wide range of presentations by composing the primitive languages.

5. EXPRESSIVENESS

All communication is based on the fact that the participants share conventions that determine how messages are constructed and interpreted. For graphical communication these conventions indicate how arrangements of graphical objects encode information. This section shows how to formalize these conventions by taking the view that graphical presentations are actually sentences of graphical languages that have precise syntactic and semantic definitions. Such definitions make it possible to determine the expressiveness and effectiveness properties of graphical languages.

Intuitively, a set of facts is expressible in a language if there is a sentence of the language that encodes every fact in the set. The difficulty with this intuition is that the sentence may encode additional incorrect facts (this is discussed in detail elsewhere [17]). Therefore, the expressiveness criteria for languages contain

[2] The *retinal* properties are so called because the retina of the eye is sensitive to them, independent of the position of the object. Although they are included in the list of encoding relationships, 3-D position and animation are beyond the scope of this research.

two conditions:

A set of facts is *expressible* in a language if it contains a sentence that
(1) encodes all the facts in the set,
(2) encodes only the facts in the set.

This section presents two examples that demonstrate the importance of these two conditions. The first example shows a case in which position on an axis cannot express a one-to-many relation. The second example shows a case in which a bar chart expresses additional incorrect facts. Before the examples can be given, however, it is necessary to develop some formal machinery for defining the syntax and semantics of graphical languages. Such machinery makes it possible to determine what information is encoded by the sentences of a language. Evaluating expressiveness requires this ability.

The formal machinery required to define a graphical language is fairly straightforward. A *graphical sentence* s is defined to be a collection of tuples:

$$s \subset \{\langle o, l\rangle \mid o \in O \wedge l \in L\},$$

where O is a set of graphical objects and L is a set of locations. Each tuple, which is called a *located object*, indicates the placement of an object at a given location. The syntax of a *graphical language* is defined to be a set of well-formed graphical sentences. This paper assumes that O and L are restricted to the standard 2-D Cartesian plane, such that the objects in O are 2-D graphical objects that have a finite, nonzero height and width, and the locations in L are the conventional binary tuples that represent the x and y positions in the Cartesian plane.[3] The height and width of a sentence is determined in the normal manner from the size and location of the objects that make up that sentence. This paper uses a variety of intuitively named functions to describe geometric properties. For example, the functions *Xmin*, *Xpos*, and *Xmax* identify the x position of the left, center, and right of a located object. Precise definitions of these functions can be found elsewhere [16].

The syntax of a language can be described with a predicate that identifies the well-formed sentences of the language. Systematic syntactic conventions can be captured by conditions that indicate when this predicate is true. For example, consider the diagram in Figure 9 that encodes the *Price* relation. Intuitively, it is an example of a set of graphical sentences (i.e., a graphical language) that is based on the syntactic convention of placing a plus object above an axis. The syntax of this "horizontal position" language can be formalized with the unary predicate *HorzPos*, which is true for any graphical sentence that consists of a horizontal axis and a set of tuples placing a plus object at a constant height somewhere above the axis.

More formally, a graphical sentence s is a legal sentence of the horizontal position language when it consists of the union of a horizontal axis set h and a set of marks m such that each located object $\langle o, l\rangle$ in m is a plus object *plusobj*

[3] The resolution of a device can be represented by replacing the Cartesian plane with a grid of pixels [16].

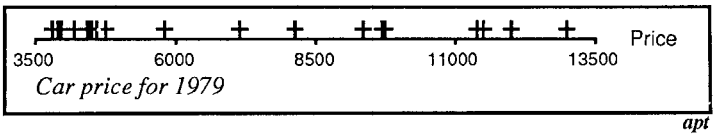

Fig. 9. The horizontal position sentence of the *Price* relation.

located at a constant height *const* above the axis:

$$\begin{aligned} &HorzPos(s) \Leftrightarrow \\ &\quad s = h \cup m \ \wedge \langle o, l \rangle \in m \Rightarrow [\\ &\qquad o = plusobj \wedge \\ &\qquad Ymax(h) \leq Ypos(l) = const \wedge \\ &\qquad Xmin(h) \leq Xpos(l) \leq Xmax(h)]. \end{aligned}$$

The symbols in this formula are used in a similar manner throughout the paper. In particular, the symbol h always stands for a horizontal axis,[4] and the symbol m always stands for a set of located objects (called *marks*) that have a related set of properties, such as objects positioned against the same axis.

Given a precise syntactic definition, the semantics of a graphical language can be specified using established formal techniques, such as denotational semantics [7]. A collection of located objects representing a graphical sentence can have a denotation in the same way as a collection of characters representing a logic formula. However, a presentation tool designs graphical sentences. It must be able to describe the semantic relationships between a graphical sentence and the encoded facts. For example, the *HorzPos* language encodes a binary relation with an axis, a set of marks, and the position of the marks on the axis. Formally, a relation called *Encodes*(*s*, *facts*, *lang*) is used to describe the semantic relationship between the objects and properties of a graphical sentence s and the set of *facts* that are encoded, given the semantic conventions of the language *lang*.[5] For example, given a relation r consisting of tuples $r(a_i, b_i)$, the following is a formal description of the three basic *Encodes* relationships for a sentence of the *HorzPos* language (see Figure 10 for an abstract description of these encodes relations). The axis h encodes the second domain of the relation, $Dom_2(r)$:

$$Encodes(h, Dom_2(r), HorzPos). \tag{1}$$

For the *Price* relation, the horizontal axis encodes the range [3500, 13000] of prices. Each located object o_i of the set of marks m encodes a unique domain value a_i of the first domain set of the relation $Dom_1(r)$:

$$Encodes(o_i, a_i, HorzPos). \tag{2}$$

Since this is true for every located object in the mark set m, the *Encodes* relation can be extended to the entire set:

$$Encodes(m, Dom_1(r), HorzPos).$$

For the *Price* relation, the marks encode the set of *Cars*.

[4] The symbol v stands for a vertical axis.

[5] Since an image might be a well-formed graphical sentence of more than one language, the *Encodes* relation includes the name of the language to indicate which semantic conventions are being described.

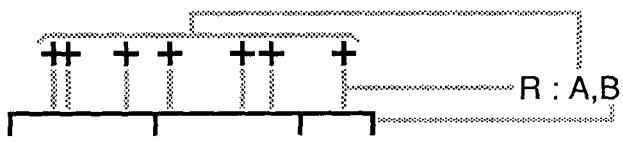

Fig. 10. The *Encodes* relationships for the horizontal position language. The graphical sentence is on the left, and the relation is on the right. The gray lines indicate that the domain sets are encoded by the objects, and the tuples of the relation are encoded by the relative positions of the marks.

Given the semantics for the objects in a graphical sentence, it is straightforward to describe the semantics for arrangements of objects. For the *HorzPos* language, the position of marks on the axis encodes the tuples of the r relation. That is, the first domain value a_i of a tuple $r(a_i, b_i)$ corresponds to a mark o_i as described in (2), and the second domain value b_i corresponds to the position of the mark o_i on the axis h, which is described with the binary function $Position(o_i, h)$. More formally, there exist two constants, *scale* and *offset*, that equate the domain value b_i with the axis position of the mark o_i for the domain value a_i:

$$\begin{aligned} &Encodes(o_i, a_i, HorzPos) \Rightarrow \\ &\quad b_i = scale \times (Position(o_i, h) + offset) \wedge \\ &\quad Encodes(Position(o_i, h), r(a_i, b_i), HorzPos). \end{aligned} \tag{3}$$

Since this is true for every mark in the mark set m, the *Encodes* relation can be extended to the domain sets. The positional encoding for the entire relation r can be described as follows:

$$Encodes(Position(m, h), r, HorzPos).$$

The presentation tool APT uses the domain set versions of these *Encodes* relations in the designs that it develops.

The formal machinery for describing the syntax and semantics of a graphical language leads to a precise statement of expressiveness:

$$\begin{aligned} &Expressible(facts, lang) \Leftrightarrow \exists s[lang(s) \wedge \forall f[\\ &\quad f \in facts \Rightarrow Encodes(s, f, lang) \wedge \\ &\quad f \notin facts \Rightarrow \neg Encodes(s, f, lang)]]. \end{aligned} \tag{4}$$

The remainder of this section gives two examples that illustrate these two conditions of expressiveness.

The first example, which focuses on expressing all the facts, is based on the *HorzPos* language. It turns out that it is possible to prove that one-to-many relations cannot be expressed in the *HorzPos* language.

THEOREM 1. *When r is a one-to-many relation, it cannot be expressed in the HorzPos language*:

$$\begin{aligned} &r(a_i, b_j) \wedge r(a_i, b_k) \wedge b_j \neq b_k \Rightarrow \\ &\quad \neg Expressible(r, HorzPos). \end{aligned}$$

PROOF. A proof by contradiction is straightforward, given the obvious geometric fact that a mark cannot have two positions on an axis. Assume that there

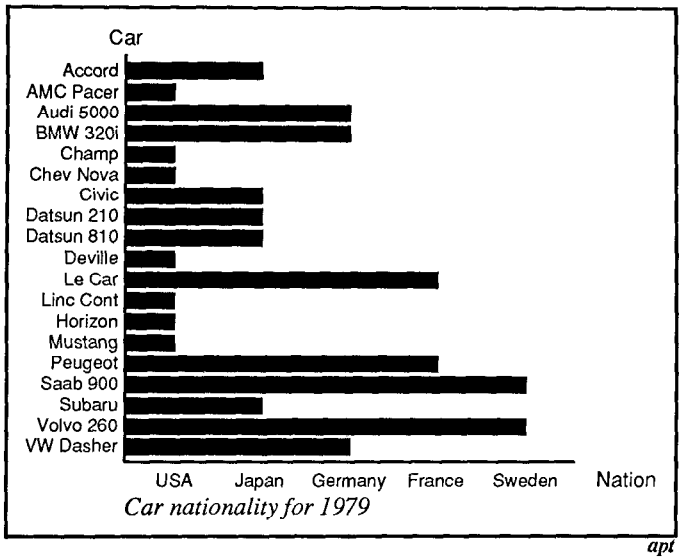

Fig. 11. Incorrect use of a bar chart for the *Nation* relation. The lengths of the bars suggest an ordering on the vertical axis, as if the USA cars were longer or better than the other cars, which is not true for the *Nation* relation.

exists a sentence s that satisfies (4) for the *HorzPos* language and relation r, which means that both tuples mentioned above are encoded in s. In particular, (2) indicates that there exists a mark o_i that is paired with the domain value a_i, and (3) indicates that $b_j = scale \times (Position(o_i, h) + offset) = b_k$, which contradicts the assumed inequality of the two domain values in the one-to-many relation. □

Although the previous theorem is not particularly deep, it illustrates the importance of precise definitions of the graphical conventions used to design and interpret information presentations. Not only do precise definitions make theorems possible, but they make clear which conventions are being used. Different conventions lead to different theorems. For example, the *HorzPos* language is based on the convention that the marks are uniquely paired with the domain values of the first domain set. Occasionally, graphical presentations use a different convention, that of pairing marks with the tuples rather than with the domain values of the first domain set. Given such a convention, the previous theorem is no longer valid because $r(a_i, b_j)$ and $r(a_i, b_k)$ can be encoded by the positions of different marks. However, this alternative convention is not so common as the *HorzPos* convention because it is natural to assume that marks are associated with domain values. For example, it is natural to assume that each mark in Figure 9 represents a unique car.

A second example, which focuses on the second expressiveness condition, illustrates the fact that some graphical languages encode additional information in the geometric relationships of the objects in a graphical sentence. Consider the bar chart diagram of the *Nation* relation in Figure 11. Most people perceive the lengths of the bars as an encoding of an ordered or quantitative set. That is, given domain tuples $r(a_i, b_i)$ and $r(a_j, b_j)$ and the corresponding bar objects bar_i

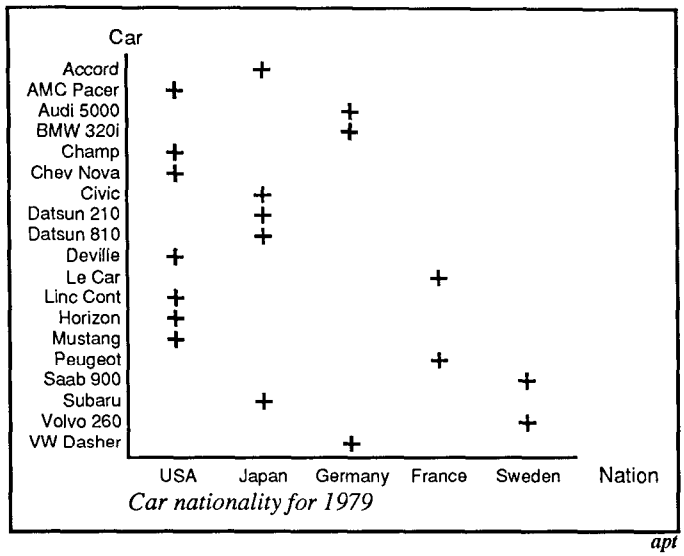

Fig. 12. Correct use of a plot chart for the *Nation* relation. Since bar charts encode ordered domain sets, plot charts are conventionally used to encode nominal domain sets. The ordering of the labels on the axes is ignored.

and bar_j, an ordering relationship among the bar lengths encodes an ordering relation among the domain values b_i and b_j:

$$Encodes(Length(bar_i) > Length(bar_j),\ b_i > b_j,\ BarChart),$$

where the rest of the *Encodes* relations for the *BarChart* language are similar to the ones for *HorzPos*. Given this *Encodes* relation, it is easy to prove that the bar chart in Figure 11 does not express the *Nation* relation because it expresses the fact that the countries are ordered, which is not correct (see Figure 3).

The plot chart of the *Nation* relation in Figure 12, which is an alternative design for encoding the *Nation* relation that avoids the bar length problem, also illustrates the importance of precise language definitions. Sometimes, people use the convention that the sequence of labels on an axis indicates an ordered domain set, which would ruin the expressiveness of the scatter plot design for Figure 12. However, the standard convention is to ignore this sequencing encoding. After all, when the second domain set is ordered, a bar chart can be used. This means that the precise definition for a plot chart language should not include an *Encodes* predicate for the ordering of the second domain set. Therefore, the plot chart in Figure 12 does *not* encode additional incorrect facts.[6]

[6] The rendering of the plot chart in Figure 12 has the independent domain set of the *Nation* function on the vertical axis, which does not conform to the often ignored convention of encoding the independent domain set of a function with the horizontal axis. In this case the rendering code flipped the axes to make the rendering of the car labels more legible, and it did not take into account the fact that such a flip might confuse the reader of the diagram about which domain set was the independent one. (The recent development of the Dot Chart design deals with this problem [5].) Trade-offs between conventions and rendering constraints often occur. In the future, the rendering component will also have to be involved in the search for the most effective design.

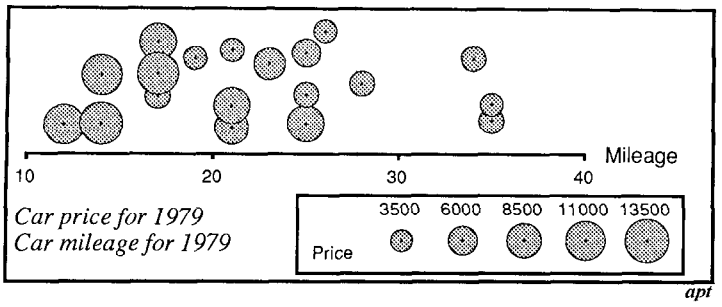

Fig. 13. Area/position presentation of the *Price* and *Mileage* relations. The vertical positioning of the marks reduces the chance that a mark is covered. This technique is called jittering; the vertical positioning does not encode any information.

6. EFFECTIVENESS

Given two graphical languages that express some information, the obvious question is which language involves a design that specifies the more effective presentation. For example, Figure 13 expresses the same price/mileage input as the scatter plot in Figure 5, but the prices are encoded with the area of the marks rather than with their position on a vertical axis. Which presentation is more effective?

Unlike expressiveness, which only depends on the syntax and semantics of the graphical language, effectiveness also depends on the capabilities of the perceiver. The difficulty is that there does not yet exist an empirically verified theory of human perceptual capabilities that can be used to prove theorems about the effectiveness of graphical languages. Therefore, one must conjecture a theory of effectiveness that is both intuitively motivated and consistent with current empirically verified knowledge about human perceptual capabilities. This section describes such a conjectural theory.

The core of this conjectural theory is an observation, made by Cleveland and McGill, that people accomplish the perceptual tasks associated with the interpretation of graphical presentations with different degrees of accuracy [6]. Cleveland and McGill focused on the presentation of quantitative information. They identified and ranked the tasks shown in Figure 14. Higher tasks are accomplished more accurately than lower tasks. Furthermore, they have some experimental evidence that supports the basic properties of this ranking.

Although the ranking in Figure 14 can be used to compare alternative graphical languages that encode quantitative information, it does not address the encoding of nonquantitative information, which involves additional perceptual tasks and different task rankings. For example, texture is not mentioned in Figure 14, and color, which is at the bottom of the quantitative ranking, is a very effective way of encoding nominal sets [23]. Therefore, it was necessary to extend Cleveland and McGill's ranking, as shown in Figure 15. Although this extension was developed using existing psychophysical results and various analyses of the different perceptual tasks, it has not been empirically verified [16].

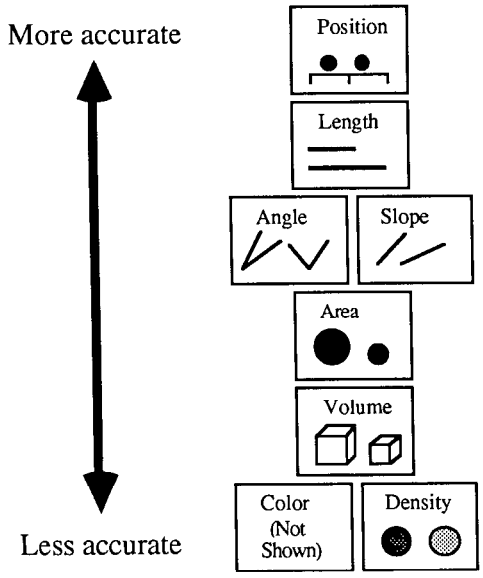

Fig. 14. Accuracy ranking of quantitative perceptual tasks. Higher tasks are accomplished more accurately than lower tasks. Cleveland and McGill empirically verified the basic properties of this ranking.

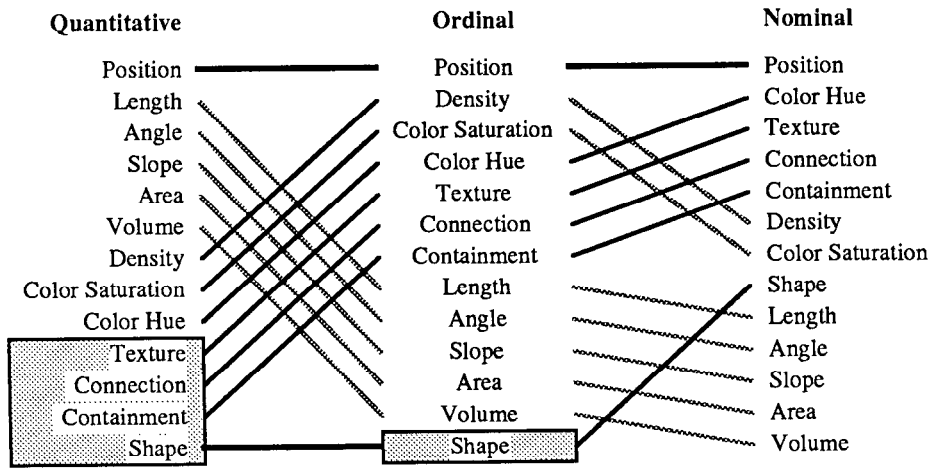

Fig. 15. Ranking of perceptual tasks. The tasks shown in the gray boxes are not relevant to these types of data.

An example analysis for area perception is shown in Figure 16. The top line shows that a series of decreasing areas can be used to encode a tenfold quantitative range. Of course, in a real diagram such as Figure 13, the areas would be laid out randomly, making it more difficult to judge the relative sizes of different areas accurately (hence, area is ranked fifth in Figure 14). Nevertheless, small misjudgments about the size of an area only leads to small misperceptions about the corresponding quantitative value that is encoded. The middle line shows that area can encode three ordinal values. However, one must be careful to make sure

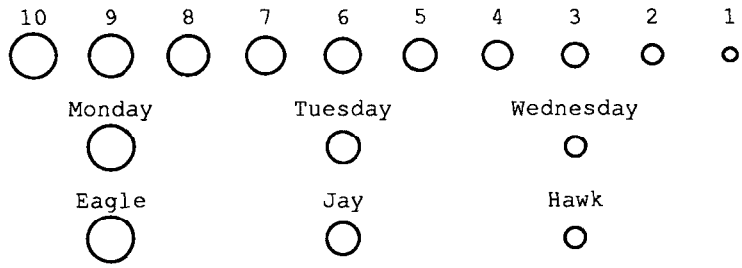

Fig. 16. Analysis of the area task. The top case shows that area is moderately effective for encoding quantitative information. The middle case shows that it is possible to encode ordinal information as long as the step size between areas is large enough so that the values are not confused. The bottom case shows that it is possible to encode nominal information, but people may perceive an ordinal encoding.

that ordinal areas are different enough so that they are not confused with each other. This is indicated by the fact that only three days of the week are encoded. If more days of the week were encoded, the step size between Tuesday's area and Monday's area would get small enough for people to start confusing them, which is quite different from confusing two quantitative values that are almost equal. The bottom line shows that area can encode three nominal values. Besides the fact that stepping is also required for nominal information, this case illustrates the additional problem that people often perceive area as an encoding of ordinal information. This analysis indicates that area should have a moderate quantitative ranking and a low nonquantitative ranking.

The ranking in Figure 15 can often be used to determine the relative effectiveness of different graphical languages. For example, Figure 17 compares the scatter plot and area/position designs for the price/mileage input. Since position has a higher ranking than area for quantitative data, it is clear that the scatter plot is a more effective design.

The ranking in Figure 15, however, does not specify a total ordering on the effectiveness of graphical languages. For example, Figure 18 compares two encodings of the *Price*, *Mileage*, and *Weight* relations. Both designs are scatter plots with information also encoded in the area of the marks. Since both designs require the same perceptual tasks, the ranking in Figure 15 does not indicate which design is more effective. The ranking can be extended to generate a lexicographic ordering with the following principle:

Principle of Importance Ordering: Encode more important information more effectively.

That is, the input to the presentation tool is actually a tuple of relations that indicates the relative importance of the relations. For example, the input

$$\langle \textit{Price}, \textit{Mileage}, \textit{Weight} \rangle$$

should be presented with scatter plot 1, which has the *Weight* relation encoded with area.

	Price	*Mileage*
Scatter plot	position	position
Area/Position	area	position

Fig. 17. Comparison of perceptual tasks for the price/mileage designs.

	Price	*Mileage*	*Weight*
Scatter plot 1	position	position	area
Scatter plot 2	area	position	position

Fig. 18. Example of designs not ordered by the effectiveness ranking.

7. COMPOSITION

Expressiveness and effectiveness criteria, which were described in the previous two sections, are not very useful without a method for generating alternative designs. The naïve approach is simply to develop an ad hoc list of graphical languages that can be filtered with the expressiveness criteria and ordered with the effectiveness criteria for each input. The major difficulty with this approach is that there is no guarantee that there will exist appropriate designs for a wide variety of presentation situations. A minor difficulty is that the entire list must always be considered, even when only a few alternatives are suitable for a given input. This section describes an alternative approach based on the idea of a composition algebra. Such an algebra consists of a basis set containing primitive graphical languages and some composition operators that can generate composite designs. Described is a specific choice of basis set and composition operators that generate many of the designs commonly found in presentation graphics [3–5, 15, 19, 21]. The study of alternative composition algebras is an open area of research.

The idea of a composition algebra occurred to me when I looked at a diagram that was very similar in design to the diagram in Figure 19. The design used in Figure 19 combines two encoding techniques that are generally not seen together. First, the prerequisites among computer science classes are encoded with links that connect nodes that encode the classes. Second, a class schedule is encoded by the position of the nodes on a vertical axis. This diagram is an example of a composite design. The primitive languages used to form this composite design are a node/link language (see Figure 20) and a vertical-axis language (see Figure 21). Given this unusual example of a composite design, I realized that many presentations could be described as compositions of a set of primitive languages.

7.1 A Basis Set of Primitive Graphical Languages

A basis set of primitive graphical languages derived from Bertin's vocabulary of graphical encoding techniques (see Figure 8) is listed in Figure 22. The primitive languages have been classified by their primary encoding technique. *Single-position languages* encode information by the position of a mark set on one axis. *Apposed-position languages* encode information by a mark set that is positioned

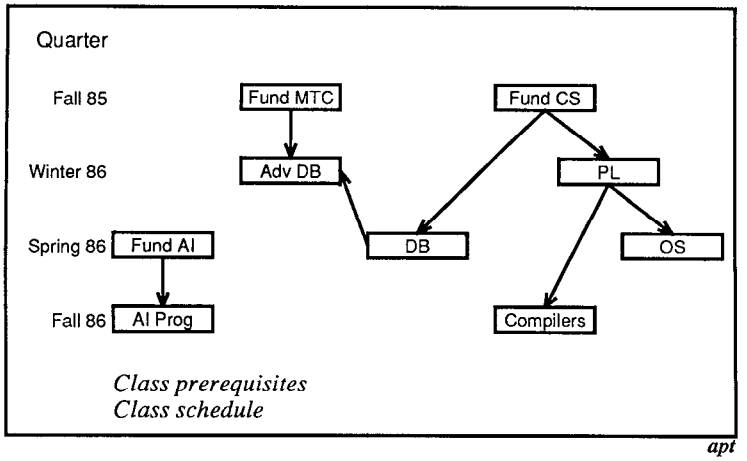

Fig. 19. Composite presentation for the prerequisite and schedule relations. The links encode the prerequisite relationships between computer science classes. The position on the vertical axis encodes the scheduling of the classes. Note that the advanced database class is scheduled before its prerequisite.

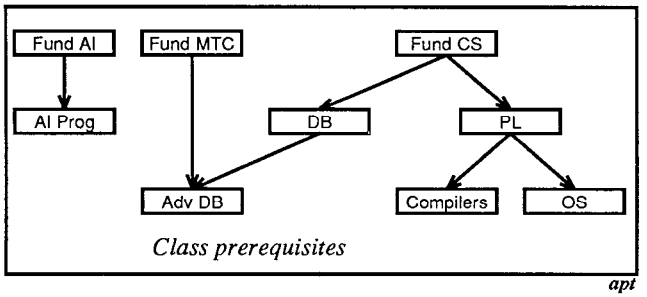

Fig. 20. Network presentation for the prerequisite relation.

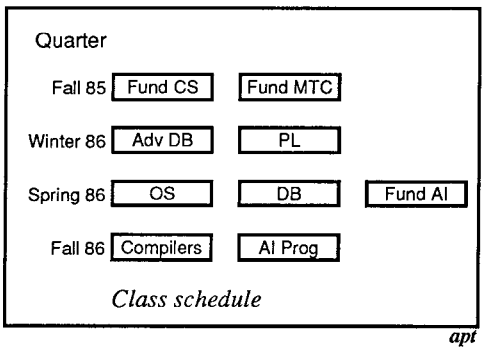

Fig. 21. Vertical-axis presentation for the schedule relation.

Encoding Technique	Primitive Graphical Language
Single-position	Horizontal axis, vertical axis
Apposed-position	Line chart, bar chart, plot chart
Retinal-list	Color, shape, size, saturation, texture, orientation
Map	Road map, topographic map
Connection	Tree, acyclic graph, network
Misc. (angle, contain, ...)	Pie chart, Venn diagram, ...

Fig. 22. A basis set of primitive graphical languages.

Encoding Technique	Syntactic Structure
Single-position	$h(m)$ or $v(m)$
Apposed-position	$vh(m)$
Retinal-list	m
Map	$vh(m)$
Connection	$m_n(m_1)$
Miscellaneous	$vh(m)$

Fig. 23. Syntactic structure of primitives. The notation is described in the text.

between two axes.[7] *Retinal-list languages* use one of the six retinal properties of the marks in a mark set to encode information. Since the positions of these marks do not encode anything, the marks can be moved when retinal list designs are composed with other designs. *Map languages*, which have fixed positions, encode information with graphical techniques that are specific to maps. *Connection languages* encode information by connecting a set of node objects with a set of link objects. *Miscellaneous languages* encode information with a variety of additional graphical techniques.

Figure 23 summarizes the basic syntactic structure of the primitive languages. The notation used in this figure is based on the fact that graphical sentences are sets of located object tuples. Except for connection languages, it turns out that every sentence of the primitive languages described in Figure 22 can be divided into the disjoint subsets

$$m \cup v \cup h,$$

where m is a set of marks, v contains at most a vertical axis, and h contains at most a horizontal axis. Furthermore, both the objects and positions of the mark sets have additional properties. The objects are either a collection of points, lines, or areas, and their positions are always fixed relative to the existing axes.[8] The notation always uses m for a set of marks, v for a vertical axis, and h for a

[7] It turns out that apposed-position languages can be described as the composition of single-position languages [16].

[8] This assumes that languages that restrict the positions of mark sets without a visible axis object, such as maps, define their sentences as having invisible axis objects.

Encoding Technique	Expressiveness Criteria
Single-position	$X \rightarrow Y$ (X is nominal)
Apposed-position	$X \times Y$ (X, Y are not nominal)
Retinal-list	X, or $X \rightarrow Y$ (X is not quantitative)
Map	$L \rightarrow X_1, \ldots$ (L is a location)
Connection	$X \times X$ (X is nominal)
Miscellaneous (angle, contain, ...)	Generally, $X \times Y$

Fig. 24. Expressiveness criteria for the primitive languages. These are the general restrictions. Specific languages can have additional restrictions.

	Nominal	Ordinal	Quantitative
Size	–	•	•
Saturation	–	•	•
Texture	•	•	
Color	•	*	
Orientation	•		
Shape	•		

Fig. 25. Expressiveness of retinal techniques. The – indicates that size and saturation should not be used for nominal measurements because they will probably be perceived to be ordered. The * indicates that the full color spectrum is not ordered. However, parts of the color spectrum are ordinally perceived [23].

horizontal axis. The notation also uses parentheses to indicate that there is a positional constraint on a set of marks. For example, the positional constraints of bar charts are described by $vh(m)$, where v and h are not empty. Sentences of connection languages consist of two sets of marks: the set of nodes m_n and the set of links m_l. The nodes constrain the position of links. The notation can also be extended to more complex designs. For example, the two bar charts in Figure 7 that are aligned on their vertical axes have the structure $v(h_i(m_i), h_j(m_j))$.

An analysis of the semantic properties of these languages leads to the expressiveness criteria shown in Figures 24 and 25. Figure 24 describes the basic expressiveness criteria for each type of primitive language. For example, single-position languages can only express binary relations that have a functional dependency. Figure 25 describes the expressiveness criteria of various retinal techniques for nominal, ordinal, and quantitative information. There are additional requirements not mentioned in Figures 24 and 25. For example, line charts can only be used when a relation describes values of a continuous function.

7.2 Some Composition Operators

The composition operators associated with the primitive languages in Figure 22 are based on a single principle:

Principle of Composition: Compose two designs by merging parts that encode the same information.

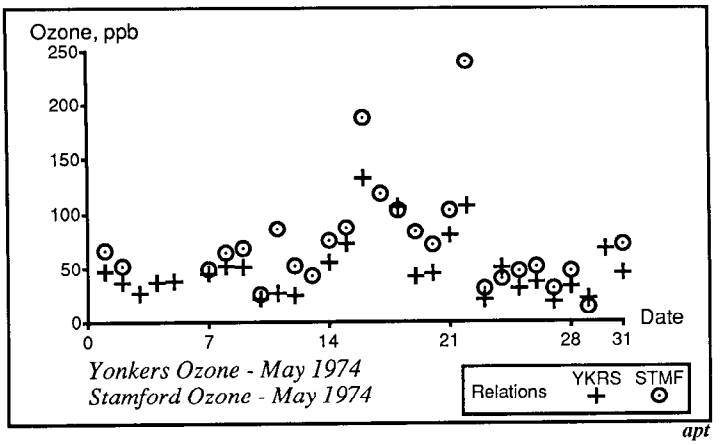

Fig. 26. Example of double-axes composition. The diagram describes a month of ozone measurements for two cities. The reason that a line chart design was not used for these data is that there are missing measurements for some of the days; a line chart is only used for continuous functions.

This principle leads to three composition operators that are based on the objects used by the primitive languages: *double-axes composition* $\bowtie_d$, *single-axis composition* $\bowtie_s$, and *mark composition* $\bowtie_m$. The following description of these composition operators gives an informal and formal definition of each of them.

Double-axes composition can compose graphical sentences that have identical horizontal and vertical axes. The multiple scatter plot in Figure 26 is an example of double-axes composition. The component designs are two plot charts that describe ozone measurements in two different cities. Since the measurements were taken in the same month, the axes of these two plot charts are identical. The composite is generated by merging the identical axes and copying the mark sets from the two component designs.

Formally, the double-axes composition $s_i \bowtie_d s_j$ is well defined if s_i contains $v_i h_i(m_i)$ and s_j contains $v_j h_j(m_j)$ such that the axis sets are not empty and encode the same information. That is, given that s_i is a sentence of language l_i and s_j is a sentence of language l_j, the *Encodes* relations indicate that both horizontal axes encode the same domain set x and both vertical axes encode the same domain set y:

$$v_i = v_j \neq \{\} \wedge h_i = h_j \neq \{\} \wedge$$
$$Encodes(h_i, x, l_i) \wedge Encodes(h_j, x, l_j) \wedge$$
$$Encodes(v_i, y, l_i) \wedge Encodes(v_j, y, l_j).$$

The composite contains $v_i h_i(m_i', m_j')$. The prime is required when the marks are changed by the composition. For example, composed bar charts move the bars next to each other. When the prime is not needed, double-axes composition is commutative. It is always associative.

Single-axis composition aligns two sentences that have identical horizontal or vertical axes. The diagram in Figure 7 is an example of single-axis composition.

The two component designs are bar charts that describe the price and mileage of some cars. Since the vertical axes encode the same set of cars, the composite can be generated by placing one diagram next to the other such that the vertical axes are aligned.

Formally, the single-axis composition $s_i \bowtie_s s_j$ is well defined if s_i contains $v_i h_i(m_i)$, s_j contains $v_j h_j(m_j)$, and the following condition is satisfied:

$$[v_i = v_j \neq \{\} \wedge \mathit{Encodes}(v_i, y, l_i) \wedge \mathit{Encodes}(v_j, y, l_j)] \vee$$
$$[h_i = h_j \neq \{\} \wedge \mathit{Encodes}(h_i, x, l_i) \wedge \mathit{Encodes}(h_j, x, l_j)],$$

where x, y, l_i, and l_j are defined in the same manner as for double-axes composition. The composite contains $v_i' h_i'(m_i')$ and $v_j' h_j'(m_j')$, where the positions of the objects are modified to place the diagrams next to each other in the viewing area. Single-axis composition is not commutative because the positions of the diagrams reverse. However, it is associative.

Mark composition is more complicated than the axis composition operators because it actually merges mark sets. For example, a mark set of uniform size that encodes information with color can be merged with a mark set of uniform color that encodes information with size. The resulting mark set uses both color and size to encode information (see Figure 30 for an example).

Mark composition merges mark sets by pairing each and every mark of one set with a compatible mark of the other set. The diagram in Figure 19 is an example of mark composition. The component design is a directed-acyclic-graph design for the prerequisite relation, which is rendered in Figure 20, and a vertical-axis design for the scheduling relation, which is rendered in Figure 21. The two mark sets are compatible because they encode the same information, and any shared positional or retinal constraints are identical. The composite is generated by merging the mark sets. That is, the composite includes a mark set that corresponds to the two mark sets of the components. The position and retinal properties of this composite mark set are based on the constrained position and retinal properties of the component mark sets. Compatibility makes sure that the component's properties do not conflict. Additional properties and objects in the component sentences are copied over into the composite. For example, the composite in Figure 19 includes the vertical-axis object from the diagram in Figure 21.

Formally, the mark composition $s_i \bowtie_m s_j$ is well defined if s_i contains $v_i h_i(m_i)$, and s_j contains $v_j h_j(m_j)$, and the marks in m_i and m_j can be paired such that each pair of located objects o_i and o_j encode the same domain value a:

$$[\mathit{Encodes}(o_i, a, l_i) \wedge \mathit{Encodes}(o_j, a, l_j)].$$

Furthermore, the position and retinal properties of these mark pairs must encode the same information. For position, an existing pair of axes means that the positions must be identical:

$$[v_i = v_j \neq \{\} \Rightarrow \mathit{Position}(o_i, v_i) = \mathit{Position}(o_j, v_j)] \wedge$$
$$[h_i = h_j \neq \{\} \Rightarrow \mathit{Position}(o_i, h_i) = \mathit{Position}(o_j, h_j)].$$

For the retinal properties, a little more formal machinery is required. A set of marks can have retinal constraints based on the six retinal properties identified

Fig. 27. The interaction of size and shape.

by Bertin. For example, the size of the marks in Figure 13 encode the values of the *Price* relation. Six functions identify the six retianl values of an object. These functions can also be used to indicate when the retinal properties of a mark set are constrained. Given a retinal function f, a set of marks m, and a graphical sentence s that contains m, the relation $Rt(s, m, f)$ is true when the retinal properties corresponding to f encode information for the mark set m. Given this relation, the following indicates that the paired marks must have the same constraints on their retinal properties:

$$Rt(s_i, m_i, f) \wedge Rt(s_j, m_j, f) \Rightarrow f(o_i) = f(o_j).$$

That is, when a retinal property is used by two mark sets to encode information, the objects for each pair of marks must have identical retinal properties.

The composite sentence $vh(m)$, generated by the mark composition $v_ih_i(m_i) \bowtie_m v_jh_j(m_j)$, is constructed in the following manner. The vertical axis v is v_i if it is not empty and v_j otherwise. The horizontal axis h is h_i if it is not empty and h_j otherwise. For each pair of marks, construct a composite mark from the constrained retinal and position properties of the component marks (which must be identical) and any remaining properties from m_i. The final condition means that $\bowtie_m$ is not commutative. It is easy to show that $\bowtie_m$ is associative.

The conditions for using these three composition operators are sufficiently general for them to be used together. The major difficulty is specifying exactly how the parts of the component sentences that are not part of the composition should be handled. The best approach is to merge pairs of axes or mark sets of the component sentence that satisfy the conditions of the composition operators. This will reduce redundancy in the composed diagram.

A rough effectiveness ranking can also be assigned to the composition operators. Mark composition is the most effective because it merges the component sentences in such a way that the number of graphical objects does not increase. Single-axis composition is the least effective because it does not actually merge the designs, which makes it harder to perceive all the information at once.

Composition can have side effects that must be addressed. For example, when size and shape are composed together, the perception of shape is made difficult when the sizes are small. For example, the marks in Figure 27 are a composition of shape and size. The shapes of the small objects begin to look the same. Therefore, care must be taken to avoid situations in which such interactions reduce the effectiveness of a composite design. APT's rendering component makes sure the marks do not get too small.

8. IMPLEMENTATION

The theoretical results described in the previous three sections have been combined in a synthesis algorithm that generates designs in order of the effectiveness

criteria described above. The synthesis algorithm has been implemented in APT, which consists of a design component followed by a rendering component (see Figure 1). The design component uses logic programming techniques to implement the synthesis algorithm. The rendering component uses object-oriented programming techniques and a device-independent graphics package to render the resulting designs. The rendering component, which was not the focus of this research, places an implementation restriction on the set of primitive languages in Figure 22 that can be used by APT to generate presentations. As of this writing, the following primitive languages remain unimplemented: orientation, texture, line charts, maps, and miscellaneous. Even with these restrictions, the prototype can generate a wide range of useful presentations. The diagrams in this paper with *apt* in the lower right corner are examples of APT's output.

APT's synthesis algorithm is based on a divide-and-conquer search strategy. The algorithm has three steps: partition, selection, and composition. These steps, which are described below, involve choices. When a particular set of choices does not lead to a composite design, backtracking is used to consider other choices. APT uses depth-first search with simple backtracking.

(1) *Partitioning.* The set of relations to be presented is divided into partitions that match the expressiveness criteria of at least one of the primitive languages. This is done recursively. For example, the input

$$\langle \textit{Price}, \textit{Mileage}, \textit{Repair}, \textit{Weight} \rangle$$

can be partitioned into the sets

$$\langle \textit{Price} \rangle \quad \text{and} \quad \langle \textit{Mileage}, \textit{Repair}, \textit{Weight} \rangle.$$

The right partition must be recursively divided because it does not match any of the primitive languages.

The principle of importance ordering is addressed by making sure that the choices among alternative partitionings give preference to the important information. The input shown above is a tuple that indicates the importance ordering for the automobile relations. It is partitioned so that the most important relation, which is the *Price* relation, will get first chance at being matched to an effective primitive language.

Since relations, as well as sets of relations, can be partitioned [16], it is possible for the input to include relations that have more than two domain sets. The cumulative bar chart in Figure 28 is an example of a design that is generated by relation partitioning. The input is a ternary relation, which is a function of years and quarters, to the Ph.D.s conferred in each quarter for a range of years. Binary relations are generated by fixing the year. The stacked bar designs corresponding to the binary relations are composed with single-axis composition to generate the cumulative bar chart.

(2) *Selection.* Given expressiveness and effectiveness criteria, selection is straightforward. For each partition generated by the previous step, the primitives are filtered, with their expressiveness criteria used to generate a list of candidate designs. For example, the list of candidate designs for the *Price* partition does

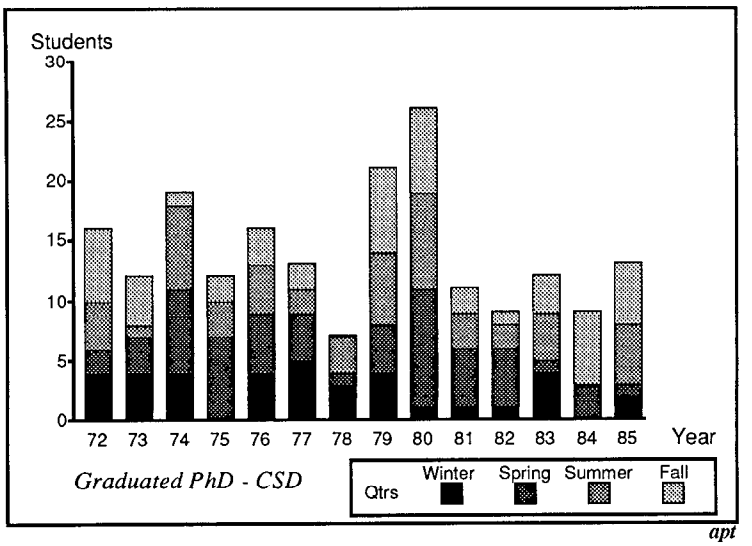

Fig. 28. A relation partitioning example. The cumulative bar chart describes the number of Ph.D. students that graduated each quarter for a range of years. The input is a relation of three domain sets, which are partitioned into a set of binary relations. The composite is generated by the single-axis composition of the stacked bar design chosen for each of these binary relations.

not include maps because the *Price* relation does not satisfy the map expressiveness criterion (see Figure 24).

The effectiveness criteria are used to order the candidate designs so that the most effective design will be the first choice. For the *Price* relation the effectiveness ordering depends on whether the application has requested that the details about the cars be placed in the background. Apposed-position languages are the most effective when the details are required, and single-position languages are the most effective otherwise. The other primitive languages are less effective because position is at the top of the perceptual task ranking shown in Figure 15.

(3) *Composition.* Composition operators are used to compose the individual designs into a unified presentation of all the information. Given designs for two partitions, the three composition operators are checked to see if they can be applied. During the generation of the composite designs, additional conditions, such as the interaction of shape and size, can be checked.

The synthesis algorithm involves choices that might not lead to a design, which means that backtracking will occur. When backtracking occurs, the next most effective primitive language or composition operator is chosen until a design is found for all the information. For example, given a request to omit the details about the cars, APT processes the automobile input shown above as follows. The partitioning step generates a partition for the *Price* relation and a partition for the *Mileage* relation. The first selection choice is the vertical-axis primitive

$$
\begin{array}{ll}
rel = x \rightarrow y \wedge \neg Numeric(x) \wedge Numeric(y) \wedge & \left.\begin{array}{l}\\\end{array}\right\} \text{Expressiveness} \\
Cardinality(x) < 20 \wedge & \left.\begin{array}{l}\\\end{array}\right\} \text{Effectiveness} \\
LineObjs(barchart, lines) \wedge VertAxis(barchart, vaxis) \wedge & \left.\begin{array}{l}\\\\\end{array}\right\} \\
Encodes(lines, x) \wedge Encodes(vaxis, y) \wedge & \left.\begin{array}{l}\\\\\end{array}\right\} \text{Assumables} \\
Length(lines, len, vaxis) \wedge Encodes(len, rel(x)) \wedge \cdots & \left.\begin{array}{l}\\\\\end{array}\right\} \\
\quad \Rightarrow Presents(barchart, rel) &
\end{array}
$$

Fig. 29. APT's bar-chart rule. The expressiveness conditions state that the relation must be a functional dependency from a nonnumeric set to a numeric set. The effectiveness condition limits the number of bars because too many bars make the presentation difficult to read. The assumables are *Encodes* relations that connect the relation and the presentation. The independent set is connected to the bar lines and the dependent set is connected to the vertical axis.

language for both of these partitions (because the relations have the same structure). However, the composition step fails to compose the resulting designs because mark composition, which is the only applicable operator, cannot merge marks that have two different positions on an axis. This failure triggers backtracking. The next most effective choice for the *Mileage* partition is the horizontal-axis primitive language. Mark composition succeeds for this choice, and the search proceeds to deal with the remaining relations in the input. (Given an indication from the application that the output medium includes color, the resulting design for the four automobile relations is the scatter plot rendered in Figure 30.)

The logic program implementing the synthesis algorithm is based on a depth-first[9] backward chaining version of a deductive algorithm called Residue [9]. Residue is useful for design problems because predicates describing the design can be declared to be assumable. During a deduction, an assumable predicate can be assumed to be true to make the deduction proceed. At the end of the deduction, all assumed predicates are returned as conditions that must be satisfied. For design problems, these conditions are exactly the design constraints. For example, Figure 29 describes APT's bar chart rule. The assumables are the *Encodes* relations of the bar-chart primitive language. When these predicates are assumed, they can be used to compose this design with others and to render the final image.

APT was deveoped on a Symbolics LISP Machine using MRS, a representation and logic programming system [18]. APT is a functional prototype, and no effort has been made to make it efficient, although designs are typically generated in 1–2 minutes, and images are rendered in less than a minute. The logic program is about 200 rules, and the rendering system is about 60 pages of LISP code.

APT demonstrates that a synthesis algorithm based on composition can be used to generate automatically effective designs that can express a wide variety of input. Although inefficient, the deductive search strategy used in APT has a number of advantages that recommend it over more procedural approaches. APT

[9] Depth-first search can be used because the effectiveness criteria place a total ordering on designs that are understood by APT. As the theory of effectiveness becomes more sophisticated, it is likely that the control strategy will also have to become more sophisticated.

can be sensitive to many factors while generating designs. The next section describes how APT is sensitive to the output medium. APT is also sensitive to requests from the application. An application can indicate that a particular primitive language should be used, even when it contradicts the effectiveness ranking that APT would normally use. This makes it possible for an application to tailor a presentation to fit the profile of a particular user. Another advantage of the logic programming approach is that it is flexible. The range of designs that can be generated by APT can be modified by changing the rules associated with the primitive languages and the composition operators. The search order can also be modified by changing the expressiveness and effectiveness criteria. This is important because presentation graphics and human perceptual abilities are not yet well understood. As our understanding advances, it will be possible to make modifications to APT that will enable it to generate even more effective designs.

9. MEDIA SENSITIVITY

Since the synthesis algorithm searches for designs, it can be sensitive to the capabilities of the output medium. For example, when the application indicates that the output medium includes color, APT designs the scatter plot shown in Figure 30 for the automobile input, which is the mark composition of two single-position designs and two retinal-list designs. When the application restricts APT to a monochrome medium, APT designs the aligned bar chart shown in Figure 31, which is the single-axis composition of four bar-chart designs.[10] Given a color medium, the *Repair* relation can be encoded by the color-list primitive language. However, given a monochrome medium, the only available retinal-list primitive languages for ordinal information are texture, saturation, and size (see Figure 25). The texture primitive language was rejected because the rendering portion of APT does not implement texture. The saturation primitive language was rejected by its effectiveness criterion because five levels of gray blend together, making the repair values blend together. The size primitive language can be selected. However, the scatter plot design requires that size also be used for one of the other relations in the input, and mark composition cannot merge two designs that use size to encode different domain sets. APT ultimately settles on the aligned bar chart shown in Figure 31.

10. DISCUSSION

The research described in this paper sets the framework for the development of presentation tools that can automatically design effective graphical presentations for a wide variety of information. The formalization of graphical presentation as a collection of graphical languages makes it possible to develop expressiveness

[10] APT always generates the aligned bar design before the scatter plot design when the application does not indicate that the details about the cars can be omitted, because the bar charts contain the names of the cars. Labels on points in the scatter plot obscure information. When the output device is a computer monitor, however, it is generally better to omit detail, because omitted details can be obtained by interacting with the display through the use of techniques such as pick-sensitive objects and pop-up windows.

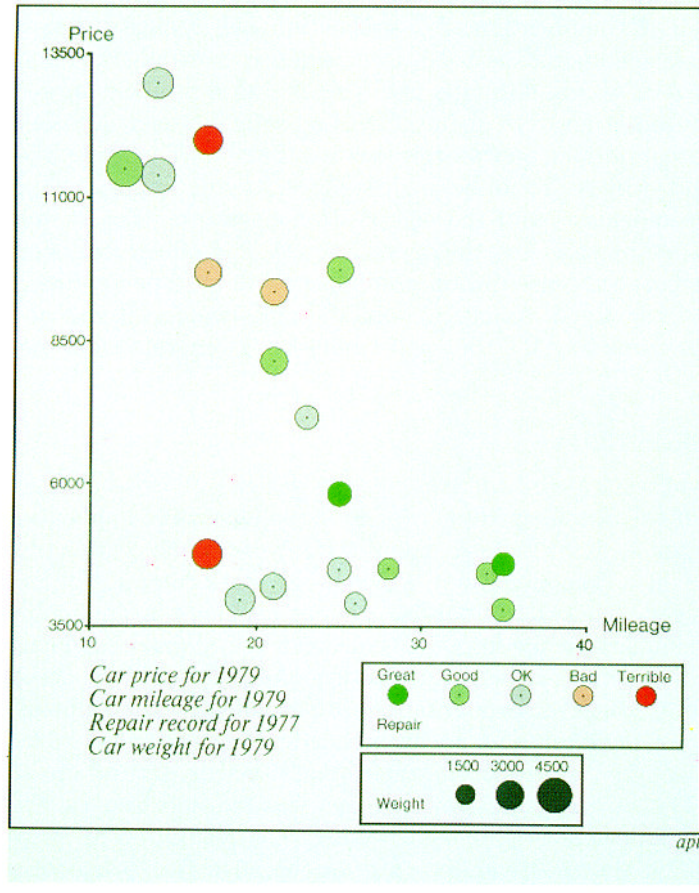

Fig. 30. Color scatter plot for four automobile relations. The design expresses the relations only if the application permits the details about the cars to be omitted.

and effectiveness criteria and a composition algebra. This formalization provides the basis of a logic program that designs presentations automatically. The prototype implementation, called APT, demonstrates the feasibility of this approach.

Many problems associated with the automatic generation of graphical designs remain to be solved. The engineering of robust presentation tools will raise many questions about the correct search criteria. Animation and 3-D presentation appear to be very powerful techniques for presenting symbolic information and should be incorporated into future tools. Larger search spaces, which can be generated with finer grained sets of primitive languages, make it more difficult to search for an appropriate choice in real time. However, it may be possible to build a discovery system that searches this larger space for unusual but effective

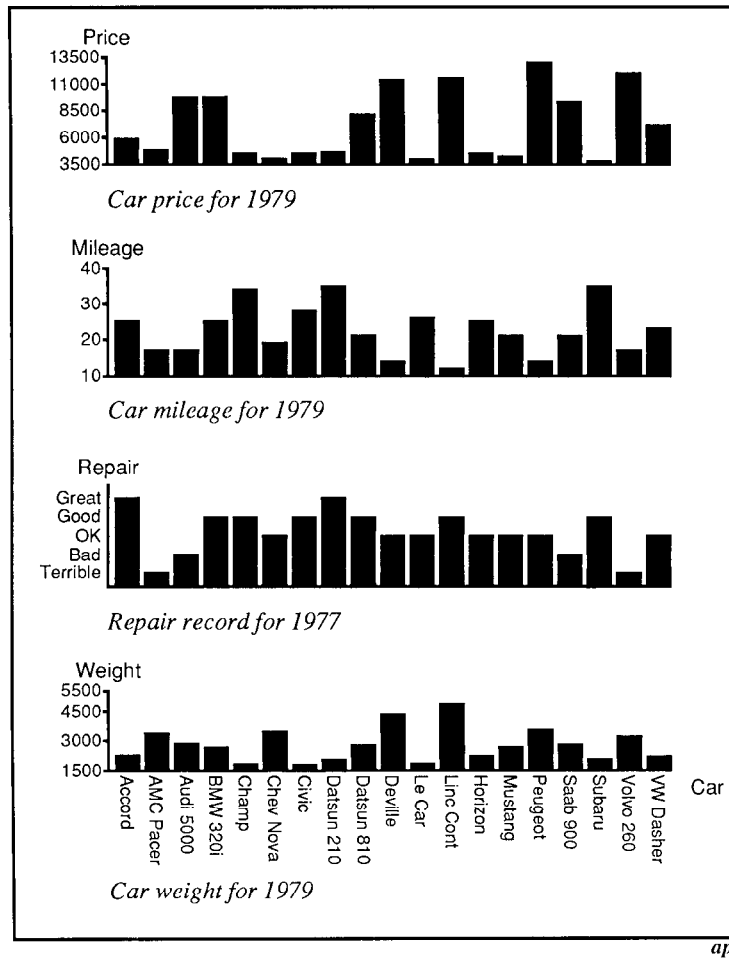

Fig. 31. Aligned bar chart of automobile data. This diagram shows the details about the car domain set. However, the general relationships are not so easy to see as in the scatter plot design.

designs. These designs can then be cached along with the primitive languages described in this paper to form a small but comprehensive search space.

This research on intelligent presentation applies artificial intelligence techniques to part of the user interface design problem—that of choosing an appropriate graphical presentation of relational data. Graphic design issues are an important concern of user interface design. This presentation research incorporates a formalized body of graphic design knowledge. Future work with these techniques can address other aspects of user interface management systems, perhaps choosing or adapting the dialogue specifications appropriate to the observed skill level of the user. When research develops theoretical results, such as the graphic design criteria and composition algebra described in this paper,

from a careful analysis of user interface systems, artificial intelligence techniques can be used to develop an intelligent user interface.

ACKNOWLEDGMENTS

I would like to thank the following people: Polle Zellweger for her inspired suggestions that improved every draft of this paper, Rick Beach for his constructive suggestions just before the submission deadline, Matt Ginsberg and Eric Bier for their helpful suggestions on early drafts, and Michael Genesereth, my advisor, for his encouragement and suggestions throughout the research. The suggestions of the referees are also appreciated.

REFERENCES

1. BEACH, R. J. Setting tables and illustrations with style. Ph.D. dissertation, Dept. of Computer Science, Univ. of Waterloo, Waterloo, Ont., Canada, 1985. Also Xerox PARC Tech. Rep. CSL-85-3.
2. BEACH, R., AND STONE, M. Graphical style—towards high quality illustrations. *Computer Graph. (SIGGRAPH) 17*, 3 (1983), 127–135.
3. BERTIN, J. *Semiology of Graphics*, W. J. Berg, Tr. University of Wisconsin Press, Milwaukee, Wis., 1983.
4. BOWMAN, W. J. *Graphic Communication.* Wiley, New York, 1968.
5. CLEVELAND, W. S. *The Elements of Graphing Data.* Wadsworth Advanced Books and Software, Monterey, Calif., 1980.
6. CLEVELAND, W. S., AND MCGILL, R. Graphical perception: Theory, experimentation and application to the development of graphical methods. *J. Am. Stat. Assoc. 79*, 387 (Sept. 1984), 531–554.
7. ENDERTON, H. B. *A Mathematical Introduction to Logic.* Academic Press, Orlando, Fla., 1972.
8. FEINER, S. APEX: An experiment in the automated creation of pictorial explanations. *IEEE Comput. Graph. Appl. 5*, 11 (Nov. 1985), 29–37.
9. FINGER, J. J., AND GENESERETH, M. R. RESIDUE—A deductive approach to design synthesis. Tech. Rep. KSL-85-1, Computer Science Dept., Stanford Univ., Stanford, Calif., Jan. 1985.
10. FRIEDELL, M. Automatic graphics environment synthesis. Ph.D. dissertation, Dept. of Computer Engineering and Science, Case Western Reserve Univ., Cleveland, Ohio, 1983. Also Computer Corporation of America Tech. Rep. CCA-83-03.
11. GNANAMGARI, S. Information presentation through default displays. Ph.D. dissertation, Dept. of Decision Sciences, The Wharton School, Univ. of Pennsylvania, Philadelphia, Pa., May 1981.
12. KAHN, K. M. Creation of computer animation from story descriptions. Ph.D. dissertation, MIT-AI-540, Massachusetts Institute of Technology, Cambridge, Mass., Aug. 1979.
13. KAHNEMAN, D., AND HENIK, A. Perceptual organization and attention. In *Perceptual Organizaton.* M. Kubovy and J. R. Pomerantz, Eds. Lawrence Erlbaum, Hillsdale, N.J., 1981, pp. 181–211.
14. KNUTH, D. E. *The Art of Computer Programming*, vol. 1. Addison-Wesley, Reading, Mass., 1973, pp. 176–179.
15. LOCKWOOD, A. *Diagrams: A Visual Survey of Graphs, Maps, Charts and Diagrams for the Graphic Designer.* Watson-Guptill, 1969.
16. MACKINLAY, J. Automatic design of graphical presentations. Ph.D. dissertation, Computer Science Dept., Stanford Univ., Stanford, Calif., 1986. Also Tech. Rep. Stan-CS-86-1038.
17. MACKINLAY, J., AND GENESERETH, M. R. Expressiveness and language choice. *Data Knowl. Eng. 1*, 1 (June 1985), 17–29.
18. RUSSELL, S. The compleat guide to MRS. KSL-85-12, Computer Science Dept., Stanford Univ., Stanford, Calif., June 1985.
19. SCHMID, C. F. *Statistical Graphics: Design Principles and Practices.* Wiley, New York, 1983.
20. STEVENS, S. S. On the theory of scales of measurement. *Science, 103* 2684 (June 1946), 677–680.

21. TUFTE, E. R. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Conn., 1983.
22. ULLMAN, J. D. *Principles of Database Systems.* Computer Science Press, Rockville, Md., 1980.
23. WARE, C., AND BEATTY, J. C. Using colour as a tool in discrete data analysis. Tech. Rep. CS-85-21, Computer Science Dept., Univ. of Waterloo, Waterloo, Ont., Canada, Aug. 1985.
24. ZDYBEL, F., GREENFELD, N. R., YONKE, M. D., AND GIBBONS, J. An information presentation system. In *7th International Joint Conference on Artificial Intelligence* (Vancouver, Canada, Aug.). AAAI, Menlo Park, Calif., 1981, pp. 978–984.

Received July 1986; revised October 1986; accepted October 1986