Information Visualization and Semiotic Morphisms

Joseph A. Goguen and D. Fox Harrell

Dept. Computer Science & Engineering, Univ. California, San Diego

Abstract

Information visualization design is generally *ad hoc*, using trial and error, and perhaps drawing on experience with prior visualization systems. This paper suggests a different approach: general design principles based on a combination of algebraic abstract data type theory, semiotics, and social theory. Major concepts include *semiotic spaces* to describe systems of related signs, *semiotic morphisms* to describe representations of signs, and *preservation measures* to describe the quality of representations. Some examples are given, each with a critical discussion, illustrating how semiotic morphisms can help with design.

1 Introduction and Motivation

Appropriate visualizations of complex data sets can be an enormous aid to scientists in discovering, verifying, and predicting significant patterns. Unfortunately, it has proven difficult to find general principles for producing appropriate visualizations. One reason is the lack of a precise definition for the word "appropriate" in the previous two sentences. The present state of HCI research does not provide an adequate basis for the design of visualizations. A few precise laws are known, but they have very limited scope (e.g., Fitt's law); there are many case studies, but their generality is unknown; and there are many methods, but reliability is uncertain (e.g., protocol analysis, usability studies, interviews – see [13] for a survey). Meanwhile, both user communities and technology bases are expanding very rapidly, while the commercial sector continues to produce exaggerated claims and mediocre products, and faith in experimental psychology and ergonomics as foundations is eroded by developments in CSCW (Computer Supported Cooperative Work) and related areas which demonstrate that many difficulties arise from taking inadequate account of the social context in which interfaces are actually used, and of the meaning behind the interfaces. In this sad situation, we badly need to explore new directions for the construction of general theories.

Slightly revised version of published article

Many fundamental issues in information visualization can be understood in terms of *representation*: a visualization *is* a representation of some aspects of the underlying information, and major questions are *what* to represent, and *how* to represent it. An adequate theory of information visualization must take account not just of current display technology capabilities, but also of the structure of complex information such as scientific data, the capabilities and limitations of human perception and cognition, and the social context of work. For scientific visualization, the social context should include current scientific theories, conventional meanings of the signs and symbols used, the unequal importance of different patterns in the data, and the collaborative nature of scientific work. While it would be difficult to deny the importance of these factors for the design of visualizations and tools to support them, it would be foolish to believe that they are easy, and in particular, it would be foolish to believe that it is easy to get the designs of visualization or visualization tools right the first time, or that design can be fully automated. For this reason, both theories and tools need to be broad and flexible, supporting relatively painless reconfiguration and evolution.

Although it seems natural to try to use semiotics as the basis for a theory of representation, classical semiotics has unfortunately not developed in a sufficiently rigorous way for our needs, nor has it explicitly addressed the representation of complex signs; also, its approach to meaning has been naive in some crucial respects, especially in neglecting (though not entirely ignoring) the social basis and contextualization of meaning. So it is not surprising that semiotics has mainly been used in the humanities, where scholars can compensate for these weaknesses, rather than in engineering, where descriptions need to be much more explicit. Another deficiency of classical semiotics is its inability to address dynamic signs and their representations, as is necessary for interfaces that involve change, instead of presenting a fixed static structure, e.g., for standard interactive features like buttons and fill-in forms, as well as for more complex situations like animations and virtual worlds. We will suggest approaches to overcoming all these limitations.

Because we consider information visualization in particular, and user interface design in general, as problems in constructing appropriate representations, we need to know what representations are, and what makes them appropriate. For the first question, we consider a *representation* to be a mapping from one structured domain of signs, called a *semiotic space* or a *sign system*, to another such space. For the second question, we can measure the quality of a representation by how well it preserves what is most important to users, subject to any constraints imposed. These ideas might seem simple, but it is not so obvious how to make them precise. Here we use some algebraic methods developed for the theory of abstract data types [16]. More specifically, the structure of a sign system is given by an *algebraic theory* (consisting of a syntax declaration, similar to a context free grammar, and a set of equations)

plus some specifically semiotic features, including hierarchical levels for signs, and priorities on constructors; more details are given in the next section, and full details appear in [9]. Dynamic interfaces can be handled by generalizing from classical algebra to a variant called hidden algebra [15], as discussed further in Section 2.4 below.

The success of this approach can be judged by the analyses and suggestions for improvement it provides for concrete examples, as in Section 3 below. While sensitive designers might reach similar conclusions, algebraic semiotics does so in a systematic way, based on general principles (in any case, the original designers of the examples in Section 3 did not reach these conclusions). The mathematical formulation of the theory also raises hope for partial automation of the design process. Finally, since all communication is mediated by signs, there is promise for applications well beyond information visualization.

2 Algebraic Semiotics

We approach questions of representation and of the quality of representation through precise notions of semiotic space and semiotic morphism, the latter being a systematic translation between semiotic spaces. Though transformations are fundamental in many areas of mathematics and its applications (e.g., linear transformations, i.e., matrices), they have not been considered in classical semiotics. This section gives an intuitive introduction to some basic concepts. The main reference for algebraic semiotics is [9]; an informal exposition of some main ideas and their motivation is given in the webnote [6], and an (intendedly) amusing introduction is given in the UC San Diego Semiotic Zoo [7]. Further applications have been developed for a course on user interface design, some of which can be browsed at the class website [5].

2.1 Semiotic Spaces

Signs need not be the simple things that we usually call "signs," such as the letters of an alphabet, or traffic signs. In written natural language, sentences are composed from words, and words are composed from letters; also, user interfaces are often very complex systems that are usefully considered single complex signs. Semiotic systems[1] capture the systematic structure of signs. This subsection introduces some elements of this notion informally; see [9] for more formal details.

[1] This paper uses the terms "sign system," "semiotic system," and "semiotic space" interchangeably.

An important insight due to Ferdinand de Saussure [24] is that signs always come in systems. A typical example considered by Saussure is the tense system for the verbs of a language. For example, in English, adding "ed" to the end of a present tense (regular) verb makes it past tense, and adding "will" in front makes it future tense, as in "walk", "walked", and "will walk". Saussure's emphasis on the structure of systems of signs rather than isolated signs has been very influential, for example, in French structuralism and post-structuralism.

A basic strategy for making complex combinations of signs easier to understand is to divide their potential parts into **sorts**, and then discover rules for the ways that each sort can be used. For example, newspapers are composed from articles, ads, cartoons, etc., while articles are composed from headlines, paragraphs, photos, diagrams, etc., and paragraphs are composed from sentences. The so-called parts of speech in traditional grammars are also sorts in this sense. Sorts may have a hierarchical structure under a **subsort** partial ordering. For example, the sort NOUN is a subsort of the sort NOUN-PHRASE.

The rules for composing signs into more complex signs are of two kinds, called constructors and axioms. **Constructors** are functions that build new signs from others signs of given sorts, plus perhaps additional parameters. For example, a computer graphics image of a cat may be given as a constructor with parameters that determine its size, color, and location on the screen. There may also be functions and predicates defined on signs; for example, a LOCATION function for graphical objects, and a HIGHLIGHTED predicate for text. **Axioms** are logical formulae built from constructors, functions and predicates; they constrain the set of possible signs.

In many examples, some constructors for signs of a given sort are more important than others. For example, a warning popup window is more important than a virtual pet cat. This gives rise to a **priority** partial ordering on the constructors for each sort. For a different example, the pollutants in a lake may be prioritized by their toxicity, to aid in the design of an appropriate visualization.

Another fundamental strategy for managing complexity is to have a hierarchy of **levels**, with signs that are not atomic being constructed from other signs that are at lower (or possibly the same) levels. Thus linguistics has levels for phonology, morphology, lexicography, syntax, and discourse (i.e., multisentential units, such as stories). Similarly, standard GUI displays have windows, which may contain other windows.

It is clear that context, including the physical setting of a given sign, can be at least as important for meaning as the sign itself. In an extreme example, the sentence "Yes" can mean almost anything, given the right context. This corresponds to an important insight of Peirce [21], that meaning is *relational*,

not just denotational (i.e., functional); this is part of the point of his famous semiotic triangle. Using the ideas of this paper, we can consider constructors that place signs in context, by making them parts of larger signs. For example, the familiar 12 hour clock tells the correct 24 hour time in the context of external illumination, which can be considered an argument of a higher level constructor for clocks-in-context.

It is worth noting that neither semiotic theories nor semiotic morphisms describe relationships between signs and the realities (if any) that they represent; rather, it is the signs determined by the theories that can be taken to describe real situations. For example, a database schema might have fields for the age, condition, type, height, etc. of roses, but only a particular database can contain actual data about roses. Thus a semiotic theory determines a class of signs, which can potentially describe things in the world.

This paragraph contains some technical remarks for those who have the background and interest. A **semiotic system** $\mathcal{S}$ is a tuple (Σ, A, P, L), where Σ is a **signature** (or grammar) with a set N of **sorts** (or non-terminals) partially ordered by a **subsort** relation, A is a set of **axioms**, P is a **priority ordering** on constructors (which are in Σ), and L is a **level ordering** on sorts. Then the **signs** of $\mathcal{S}$ are the elements of an "initial" (i.e., standard, or "intended") model of $\mathcal{S}$, which is known to exist for many reasonable choices of a logic to use for Σ and A (for example, equational logics and Horn clause logics have initial models, as do all "liberal institutions" in the sense of [12]). More mathematical details can be found in [14].

2.2 Semiotic Morphisms and Design

Crafting a helpful explanation or a good "icon" (in the informal sense of computer graphics rather than in Peirce's technical sense), choosing a good file name, or using a mixture of media to present given content in a satisfactory way, are all problems of translating signs in one system to signs in another system. In such cases, we know the source system, and we seek a suitable target system and an appropriate transformation that presents the information of interest in an appropriate way; often we even know the target system. This is the problem of *design*. Conversely, we may know the target sign system, and seek to infer properties of signs in the source system from their images in the target system; this happens, for example, when we try to understand a poem, an equation, a drawing, or indeed, anything at all. Let's call this the *inverse problem*, as opposed to the *direct problem* of design.

Information visualization is an especially good source of illustrations for algebraic semiotics, due to two advantages that information visualizations have

over arbitrary design problems. These are that the source space is concrete and given in advance, and that the target space consists of visual signs. The designer must be sensitive to features of the data to create a useful visualization, but certain structural features may not be obvious, and it may be even less obvious which of them are the most important. The process of considering a visualization as a semiotic morphism can focus the designer on such basic structural issues, and thus help in creating a good graphical representation.

Because semiotic systems are theories rather than models, semiotic morphisms must be translations from one theory to another, rather than translations from one concrete sign to another. This may seem indirect, but it has important advantages. First, these are theories of systems of signs, rather than of particular signs. In the case of information visualization, each model of the source theory is a possible dataset to be visualized, and each model of the target theory is a possible graphic representation. Dealing with theories forces the designer to more carefully consider the space of possibilities, instead of being seduced by idiosyncratic features of some particular data sets that happen to be available. Second, taking theories as our basis allows new structure to be added later, by expanding the theory in a consistent way.

In general there are many different semiotic morphisms between two given semiotic spaces, each determining a different way to represent signs. For example, in scientific visualization, a database may be presented as a text file, or displayed graphically in many different ways. Semiotic morphisms take structure in the source space to structure in the target space, mapping sorts to sorts, subsorts to subsorts, constructors to constructors, etc. But in many real world applications, not everything *can* be preserved, so these maps must be *partial*. Axioms should also be preserved – but again in practice, not all axioms are preserved. Design is the problem of massaging a source space, a target space, and a morphism, to achieve acceptable quality, subject to constraints. The extent to which different kinds of structure are in fact preserved gives a way to compare the quality of semiotic morphisms, as discussed further in the next subsection. Semiotic morphisms should of course also preserve content, but there are many examples where this too is partial; for example, relatively little content is preserved in representing a book by its table of contents; and in scientific visualization, a major issue is what aspects of a dataset should *not* be displayed.

This paragraph continues the technical remarks at the end of Section 2.1 for those who have the background and interest. A **semiotic morphism** from $\mathcal{S}$ to $\mathcal{S}'$ consists of a partial theory morphism from (Σ, A) to (Σ', A') that partially preserves the priority and level orderings. Under certain reasonable conditions (e.g., if the logic in which theories are expressed is liberal in the sense of [12]), a semiotic morphism induces a (partial) homomorphism on the initial models, which maps the signs of $\mathcal{S}$ to signs of $\mathcal{S}'$. There is always

a natural "forgetful" mapping in the reverse direction. More mathematical details can be found [14].

2.3 Quality of Semiotic Morphisms

Each aspect of semiotic spaces that might be preserved gives rise of a different measure of quality, given as the degree to which this aspect is preserved. For example, given semiotic morphisms M_1 and M_2 from one semiotic space S_1 to another S_2, we may define $M_1 \sqsubseteq_C M_2$ if M_2 preserves every constructor that M_1 preserves, and $M_1 \sqsubseteq_A M_2$ if M_2 preserves every axiom that M_1 preserves. Other preservation relations are defined similarly [9]. There are also more refined orderings, e.g., $M_1 \sqsubseteq_{C,s} M_2$ if M_2 preserves every constructor of sort s that M_1 preserves; and we can define Boolean combinations of all these orderings, to get something appropriate for a particular application. For example, [10] applies these ideas in justifying design decisions for the user interface to a theorem proving system. Note that these quality measures are partial orderings, rather than linear numerical scales; this is appropriate because semiotic spaces are *qualitative*, in that they are concerned with structure. However, we can certainly define numerical scales if we wish to; for example, the percentage of constructors of sort s preserved corresponds to $\sqsubseteq_{C,s}$ but conveys much less information than $\sqsubseteq_{C,s}$ does, since the latter can be used to determine exactly which constructors are preserved (by comparing a given morphism with other morphisms).

2.4 Some Further Topics

Harvey Sacks' notion of *category system* [22] from the branch of ethnomethodology [3] called conversation analysis [23] is related to semiotic systems, but is less formal. Our previous work on the nature of information [8] also uses ideas from ethnomethodology, and can be seen as providing a philosophical and methodological foundation for algebraic semiotics, that takes account of the social nature of signs.

Lakoff, Johnson and others have developed the flourishing field of cognitive linguistics, building on previous careful studies of metaphor [19,18,20]. Fauconnier and Turner introduced the notion of *blending* [2], and demonstrated its importance for many aspects of cognition. See the blending website for much more information [27]. Simple examples from natural language include "house boat," "road kill," "artificial life," and "computer virus," each of which is a blend of its two component words. It happens that "boat house" has a different meaning from "house boat" because a different blend is computed. This is not because the order of the words is different, but because the same

two spaces can have many different blends [9]. Semiotic spaces significantly generalize the conceptual spaces used in cognitive linguistics, because they allow far more than just objects and binary relations. An appropriate generalization of blending is given in [9], covering many interesting examples in user interface design and information visualization. In this setting, a **blend** is built from two (or more) semiotic morphisms having a common source, called the **generic space**, with targets called the **input spaces**, by providing two (or more) semiotic morphisms from the input spaces to a **blend space**, subject to certain "optimality" conditions that rule out the uninteresting cases [14].

Hidden algebra extends the algebraic theory of abstract data types to handle states and dynamics, as well as concurrency and nondeterminism [15]. These are exactly the features needed to move algebraic semiotics from static signs to dynamic signs, for handling interactive interfaces, animated visualizations, virtual worlds [11], etc. Our approach requires that the cognitive and social dimensions of this extension should also be addressed. These can be explored using Gibson's notion of affordance, which he defined as "a capability for a specific kind of action, involving an animal and a part of its environment" [4]. For example, a BACK button on a browser provides an affordance for returning to the previously viewed page; Norman emphasizes that for designers, these should be "perceived affordances." Werner Kuhn has used semiotic morphisms, Gibsonian affordances, and blending to develop semantics for geographic information system interfaces [17].

3 Some Examples

Four examples are given in the following subsections, each with a discussion showing how semiotic morphisms can help with the design of information visualizations, including suggestions for improving some well known displays.

3.1 A Code Browser

Because a major intuition of semiotic morphisms is that they should preserve what is most important, it may be surprising that, if there is a conflict between structure and content (e.g., because not all the data can be displayed at once), it is more important to preserve structure than content. This is called **Principle F/C** in [14], and it is nicely illustrated by Figure 1, which is based on a code browser built at Bell Labs [1]. The content of this display, which is the code of some program, has been sacrificed in favor of its structure, which

Fig. 1. A Code Browser

is its division into files and procedures. Two spatial dimensions are used to represent this structure, while color (which shows up as shading in the black and white version) is very effectively used to represent the age of the code. (The superimposed window on the bottom gives an overview of the whole program, plus a close-up showing some actual text. This illustrates the overview and zoom features of the system.)

Without knowing the use of this system, it is impossible to know how appropriate its representation really is. Still, we can infer (this is an example of the "inverse problem") from the display that the designer thought that the age of code was the most important attribute, presumably because of its value in debugging. However, such a tool would be even more useful if it could be configured to highlight with colors a variety of features of interest for a variety of problems; such features might include references to certain variables, certain uses of pointers, certain kinds of recursion, etc. (e.g., consider what might be needed to work on the Y2K problem).

3.2 FilmFinder

Figure 2 illustrates FilmFinder, a system from Ben Shneiderman's group at the University of Maryland [25] for displaying films, with the vertical axis indicating popularity, the horizontal axis indicating date, and the color indicating genre[2]; the area on the right side is for controlling the system. We can see this display as the image under an appropriate semiotic morphism of a sign

[2] As before, this is indicated by tones of gray in our rendition of the display.

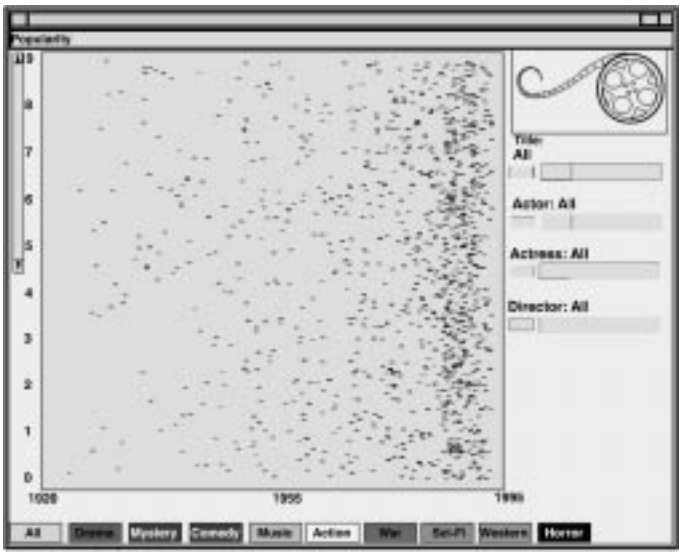

Fig. 2. FilmFinder

in a system of information about films, and we can infer what information the designer of this interface thought users would consider most important, namely the popularity, date, and genre of each film.

Treating this figure as a display of scientific data about the movie industry, we see that the density of films is significantly greater in the most recent years, except perhaps for those genres that are least popular; one can also notice other facts, such as that there has always been a higher percentage of drama, and that there are increasing percentages of action and horror.

However, this representation is not as useful as it could be. The problem is that too much content and not enough structure has been preserved. For example, it would seem better to aggregate all films having approximately the same attributes of interest into one blob, and then display the number of films in a blob using a distinct visual attribute, such as size or brightness. Successive blobs of the same kind could then be connected by lines having the same color as the blobs. Users could click on a blob to see what's in it, preferably displayed in a new popup window. These revisions could facilitate search.

3.3 A Later Version of FilmFinder

Figure 3 depicts a later version of the same tool as in Figure 2, for the same domain of films (the SpotFire version of FilmFinder, from IVEE Development in Sweden); the main improvement is to give the user more control over what is displayed and how it is displayed. The particular display shown uses length

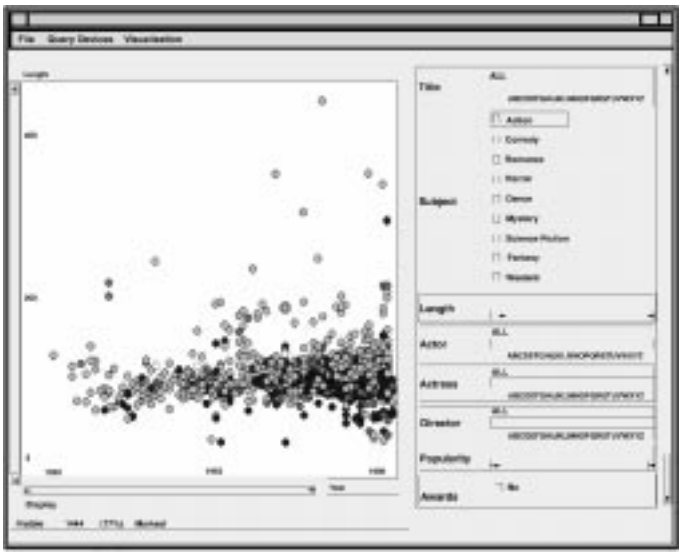

Fig. 3. The SpotFire version of FilmFinder

and date for its two axes, and again uses color for genre, though the genre color coding scheme is not indicated; prize winning films are highlighted by having a larger size. Here we can observe a clustering at around 90 minutes length, and we can again observe that there are too many dots to be useful, even though this particular display cuts off at 1990! If the user is looking for a particular film or class of films, she will have to narrow the focus by imposing additional constraints, and this single display does not give us enough information to know how effectively that can be done. We may presume that the (possibly imaginary) user who created this display thought that these particular attributes were the most interesting at a certain point during a sequence of displays constituting a search; but in fact, they do not seem particularly useful.

We can also infer what the designer of this version thought would be most important, by examining the controls on the right of the display; we may hope that these were determined by polling an adequate pool of typical users, but the key issue should be how easy it is to use these controls in scenarios that have been found to be of particular importance. Presumably typical users are more likely to be looking for a good video to rent, than they are to be analyzing trends in the movie industry. So once again, the controls should reflect the key features involved in typical searches, rather than just the most important attributes of films in general. It would take some experimental work to determine what these key search relevant attributes might be. But we can still criticize the design of the control console, because of its exclusive focus on simple attributes instead of structure. And we can criticize the fine grain control given to users over length and year, suggesting instead that soft constraints would be more appropriate; it also seems doubtful that length is a highly significant attribute for search. In addition, we can criticize its design philosophy, advocating instead a more socially oriented approach that relates

the profile of one user to the profiles of other users to select films that similar users have found interesting (there are numerous variations on this theme, such as listing films that a user's friends have liked). Finally, we can note that the design ideas proposed to improve the previous version of this system still apply to this version.

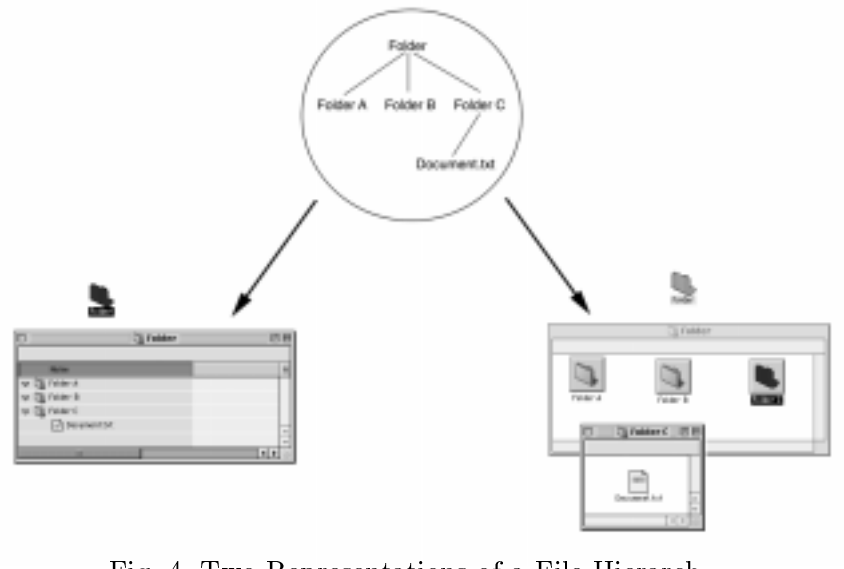

Fig. 4. Two Representations of a File Hierarchy

3.4 A File System

Figure 4 sketches a semiotic space for a file hierarchy, along with two semiotic morphisms, for visualizing it two different ways in the graphical user interface of Apple's Macintosh OS 8.6. The source space is a representation of the abstract structure of the file system; its structure is that of an ordered labeled finite tree. When Folder C is opened in the representation on the right, the location of file `Document.txt` is represented textually in the small area at the top of its window, whereas in the left representation, its location has a visual representation, based on position, including indentation. The left visualization is better, because it shows more of the source space structure in visual form, and also provides more browsing affordances in visual form. However, even more could be done in this direction.

4 Discussion

As the examples above illustrate, it is often more practical to apply algebraic semiotics informally, calling on precise definitions only when needed for es-

pecially difficult design decisions, and otherwise using the formal framework mainly as a way to guide the analysis. The examples also illustrate that even a little relevant theory can pinpoint significant deficiencies and suggest improvements. The UCSD Semiotic Zoo [7] displays a number of other graphical designs, and uses algebraic semiotics to analyze their deficiencies.

Measuring quality by what is preserved and how it is preserved seems a novel idea, at least when formulated with the precision and generality suggested here. The principle that it is more important to preserve structure than content when a trade-off is forced, has surprised even some design professionals, although it is in the literature for many special cases, for example in the books of Edward Tufte, e.g., [26]. Another non-obvious result is that preserving high level sorts is more important than preserving priorities, when a trade-off is necessary. The need to take account of social issues in user interface design, e.g., in our discussion of Figure 3, is also surprising to many people; for this reason, our version of semiotics is not just algebraic but also social. This insight is not unique to algebraic semiotics; for example, the importance of social factors in HCI is the focus of its CSCW subfield.

References

[1] Stephen Eick. Engineering perceptually effective visualizations for abstract data. In *Scientific Visualization Overviews, Methodologies and Techniques*, pages 191–210. IEEE, 1997.

[2] Gilles Fauconnier and Mark Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 1998.

[3] Harold Garfinkel. *Studies in Ethnomethodology.* Prentice-Hall, 1967.

[4] James Gibson. The theory of affordances. In Robert Shaw and John Bransford, editors, *Perceiving, Acting and Knowing: Toward an Ecological Psychology.* Erlbaum, 1977.

[5] Joseph Goguen. User interface design class notes. The CSE 271 website, at `www.cs.ucsd.edu/users/goguen/courses/271`.

[6] Joseph Goguen. Semiotic morphisms, 1996. Available on the web at `www.cs.ucsd.edu/users/goguen/papers/smm.html`. Early version in *Proc., Conf. Intelligent Systems: A Semiotic Perspective, Vol. II*, ed. J. Albus, A. Meystel and R. Quintero, Nat. Inst. Science & Technology, pages 26–31.

[7] Joseph Goguen. The UCSD Semiotic Zoo, 1996–2001. Website at URL `www.cs.ucsd.edu/users/goguen/zoo/`.

[8] Joseph Goguen. Towards a social, ethical theory of information. In Geoffrey Bowker, Leigh Star, William Turner, and Les Gasser, editors, *Social Science, Technical Systems and Cooperative Work: Beyond the Great Divide*, pages 27–56. Erlbaum, 1997.

[9] Joseph Goguen. An introduction to algebraic semiotics, with applications to user interface design. In Chrystopher Nehaniv, editor, *Computation for Metaphors, Analogy and Agents*, pages 242–291. Springer, 1999. Lecture Notes in Artificial Intelligence, Volume 1562.

[10] Joseph Goguen. Social and semiotic analyses for theorem prover user interface design. *Formal Aspects of Computing*, 11:272–301, 1999. Special issue on user interfaces for theorem provers.

[11] Joseph Goguen. Towards a design theory for vitual worlds: Algebraic semiotics, with information visualization as a case study. In *Proceedings, Virtual Worlds and Simulation*, pages 298–303. Society for Modelling and Simulation, 2001.

[12] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992.

[13] Joseph Goguen and Charlotte Linde. Techniques for requirements elicitation. In Stephen Fickas and Anthony Finkelstein, editors, *Requirements Engineering '93*, pages 152–164. IEEE, 1993. Reprinted in *Software Requirements Engineering (Second Edition)*, ed. Richard Thayer and Merlin Dorfman, IEEE Computer Society, 1996.

[14] Joseph Goguen and Grant Malcolm. *Algebraic Semantics of Imperative Programs*. MIT, 1996.

[15] Joseph Goguen and Grant Malcolm. A hidden agenda. *Theoretical Computer Science*, 245(1):55–101, August 2000. Also UCSD Dept. Computer Science & Eng. Technical Report CS97–538, May 1997.

[16] Joseph Goguen, James Thatcher, and Eric Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In Raymond Yeh, editor, *Current Trends in Programming Methodology, IV*, pages 80–149. Prentice-Hall, 1978.

[17] Werner Kuhn. Modeling the semantics of geographic categories though conceptual integration. In M.J. Egenhofer and D.M. Mark, editors, *Geographic Information Science, Second International Conference (GIScience 2002)*, pages 108–118. Springer, 2002. Lecture Notes in Computer Science, Vol. 2478.

[18] George Lakoff. *Women, Fire and Other Dangerous Things: What categories reveal about the mind*. Chicago, 1987.

[19] George Lakoff and Mark Johnson. *Metaphors We Live By*. Chicago, 1980.

[20] George Lakoff and Rafael Núñez. *Where Mathematics Comes from: How the Embodied Mind Brings Mathematics into Being*. Basic Books, 2000.

[21] Charles Saunders Peirce. *Collected Papers*. Harvard, 1965. In 6 volumes; see especially Volume 2: Elements of Logic.

[22] Harvey Sacks. On the analyzability of stories by children. In John Gumpertz and Del Hymes, editors, *Directions in Sociolinguistics*, pages 325–345. Holt, Rinehart and Winston, 1972.

[23] Harvey Sacks. *Lectures on Conversation*. Blackwell, 1992. Edited by Gail Jefferson.

[24] Ferdinand de Saussure. *Course in General Linguistics*. Duckworth, 1976. Translated by Roy Harris.

[25] Ben Shneiderman. *Designing the User Interface*. Addison Wesley, 1998. Third edition.

[26] Edward Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[27] The blending website. Maintained by Mark Turner, and available at the URL `www.wam.umd.edu/~mturn/WWW/blending.html`.