Steps towards a Design Theory for Virtual Worlds

Joseph A. Goguen
Department of Computer Science and Engineering
University of California at San Diego

Abstract Virtual worlds, construed in a broad enough sense to include text-based systems, as well as video games, new media, and user interfaces of all kinds, are increasingly important in scientific research, entertainment, communication, and art. However, we lack scientific theories that can adequately support the design of such virtual worlds, even in simple cases. Semiotics would seem a natural source for such theories, but this field lacks the precision needed for engineering applications, and also fails to addresses interaction and social issues, both of which are crucial for applications to communication and collaboration. This paper suggests an approach called algebraic semiotics to help solve these and related problems, by providing precise application-oriented basic concepts such as sign, representation, and representation quality, and a calculus of representation that includes blending. This paper also includes some theory for narrative and metaphor, and case studies on information visualization, proof presentation, humor, and user interaction.

1 Introduction: Motivation, Difficulties and Approaches

The term "virtual world" is used in many ways, but perhaps virtual worlds can be broadly characterized as the class of media experiences that provide some sense of immersion and closure. By *immersion*, which is sometimes also called *virtuality*, we mean a sense of being engaged with non-physically present entities through material mediation in the immediate real world, but not with the other aspects of the immediate real world, and by *closure* we mean that the virtual world gives an appearance of relative completeness, although it may of course be changing. A lecture, a conversation, a movie, a magazine, a formal paper, a video game, a user interface, can all be virtual worlds in this sense. A major factor in creating immersion and closure is the *coherence* of the world; of course, there are many other factors, relating for example to the situation, background, and interests of participants, but this paper is focused on ways to achieve coherent representations.

Given the enormous cultural and economic importance of current media for communication, entertainment, and art, as well as the promise of new media, there would be many uses for scientific theories that could provide guidance for difficult tasks, such as the following:

- designing new media (e.g., virtual reality environments with haptics);
- creating new metaphors (e.g., beyond the desktop for PCs);
- making new hardware (such as wireless applicances) more usable;
- designing new genres (such as interactive poems); and
- supporting non-standard users (e.g., with disabilities).

Because virtual worlds are user interfaces in some broad sense, and because user interface design is a well-developed area of computer science (which is also known as human-computer interaction, or

HCI, or sometimes CHI), this would seem another good place to look for appropriate theory. But most HCI results are either very precise but also highly specialized and therefore not very useful (e.g., Fitt's law), or else they are very general but of uncertain reliability and generality (e.g., protocol analysis, questionnaires, case studies, usability studies).

Another plausible place to seek a theory of virtual world design would be semiotics, a subject founded by Charles Sanders Peirce [Peirce, 1965] and Ferdinand de Saussure [Saussure, 1976] in the late nineteenth century. Peirce was an American logician concerned with problems of meaning and reference, who concluded that these are relational rather than denotational, and who also made an influential distinction among modes of reference, as symbolic, indexical, or iconic. Saussure, a Swiss linguist, wanted to understand how features of language relate to meanings, and he emphasized binary features and denotational meaning. More recent thinkers like the French literary theorist Roland Barthes [Barthes, 1968] combined and extended these theories, creating a powerful language for cultural and media studies, which in various versions has been called semiotics, semiology, structuralism, and finally post-structuralism. Unfortunately, this tradition:

1. does not have mathematical precision needed to integrate well with engineering processes;
2. does not consider representing signs in one system by signs in another, as is needed for the study and design of interfaces;
3. has not addressed dynamic signs, which are necessary for the study and design of interaction;
4. has not much considered social issues, such as arise in shared worlds;
5. tends to ignore the situated, embodied aspects of sign use;
6. tends towards a Platonistic view of signs, as actual existing abstract entities; and
7. often considers only single (complex) signs (e.g., a novel or a film), rather than systems of signs;

Therefore semiotics needs to address some significant problems before it can meet all our needs. This paper sketches how algebraic semiotics attempts to bridge this gap. The theory originated in an attempt to understand data from an early experimental study of multimedia learning [Goguen and Linde, 1984], and was later elaborated for applications to user interface design; more complete expositions appear in [Goguen, 1999a, Goguen and Harrell, 2003] and [Goguen, 2003], though the theory is still evolving. Here we focus more on motivation and applications.

There are at least two perspectives that one might take towards the study of signs and representations: *pragmatic* and *theoretical*. The first is the perspective of a designer, who has a job to get done, often within constraints that include cost, time, and stylistic guidelines; we may also call this an engineering perspective, and it will generally involve negotiating trade-offs among various values and constraints. The second is the perspective of a scientist who seeks to understand principles of design, and is thus engaged in a process of constructing and testing theories. From the second perspective, it makes sense to describe semiotic theories in a detailed formal way, and to test hypotheses by doing calculations and experiments with users. But from the pragmatic perspective, it makes sense to formalize only where this adds value to the design process, e.g., in especially tricky cases, and even then, only to formalize to the minimum extent that will get the job done. Our experience is that one can often get considerable benefit from applying principles of algebraic semiotics, such as identifying and preserving key features of the source theory, without doing a great deal of formalization.

From either the pragmatic or theoretical perspective, one should seek to model semiotic theories as simply as possible, since this will simplify later tasks, whether they are engineering design or scientific theorizing and experimentation (not forgetting that the conceptual simplicity of a theory does not necessarily correspond to the simplicity of its expression in any particular language). However, from a pragmatic perspective, good representations need not be the simplest possible, for reasons that include engineering tradeoffs, the difficulty (and inherent ambiguity) of measuring simplicity, and social and cultural factors, e.g., relating to esthetics. Similar considerations apply, though to a notably lesser extent, to the simplicity of semiotic theories, since creating such theories is itself a design task, subject to various trade-offs. It may be reassuring to be reminded that in general there is no unique best representation.

Sections 2 and 3 develop some basic theory of algebraic semiotics. Two main concepts are semiotic theory and semiotic morphism, which generalize the conceptual spaces and conceptual mappings of Fauconnier and Turner [Fauconnier and Turner, 1998, Fauconnier and Turner, 2002], by taking account of structure and dynamics. Some measures of quality and design principles are given, including a trade-off between form and content. Although similar principles can be found in many places, none seem to be either as precise or as general as those described here. This section also discusses metaphor and blending in natural language, and gives some basics of a calculus of representation. Section 4 describes a number of case studies, including information visualization, proof presentations, humor, and user interaction; a discussion of narrative is included here. Some conclusions, future research directions, and social implications for virtual worlds are discussed in Section 5.

Before beginning, it may help to be clear about the philosophical orientation of this work, because it is very common in Western culture for mathematical formalisms to claim and be given a status beyond what is warranted. For example, Euclid wrote, "The laws of nature are but the mathematical thoughts of God." Similarly, the "situations" in the situation semantics of Barwise and Perry, which resemble conceptual spaces (but are more sophisticated – perhaps *too* sophisticated) are considered to be actually existing, ideal Platonic entities [Barwise and Perry, 1983]. Somewhat less grandly, one might consider that conceptual spaces are somehow directly instantiated in the brain. However, the point of view of this paper is that such formalisms are constructed in the course of some task, with the heuristic purpose of facilitating consideration of certain issues in that task, which might be scientific study or engineering design. Under this non-Platonist view, all theories are situated social entities, mathematical theories no less than others; of course, this by no means implies that they cannot be useful.

Acknowledgements I thank Drs Kirstie Bellman and Christopher Landauer of Aerospace Corp for inviting me to the 2001 Virtual Worlds and Simulation Conference, and for their valuable comments. I also thank the students in my UCSD classes CSE 171 and 271 for their feedback on topics in this paper.

2 Algebraic Semiotics

The basic notions of algebraic semantics are sign (or semiotic) system, semiotic morphism, and representation quality; these are discussed in the following subsections.

2.1 Signs and Sign Systems

The definition below of sign system incorporates the insight of Saussure that individual signs should not be studied in isolation, but rather as elements of systems of related signs [Saussure, 1976], of Peirce that signs may have parts, subparts, etc., that play different roles [Peirce, 1965], and of Goguen that sign parts have different saliencies, depending on the roles that they play [Goguen, 1999a].

The structure of a sign system can be described by an algebraic theory, since they are in particular abstract data types, and it is well known that these can be defined algebraically [Goguen and Malcolm, 1996]. In addition, signs become what they are by virtue of attributes that differ from those of other signs, as shown for example by vowel systems (how the space of possible vowel sounds is divided into specific vowels for a given dialect of a given language), as well as by traffic signs, alphabets, numerals, etc. However, these attributes need not be binary, as was supposed to Saussure and his followers in the French structuralist movement, such as Levi-Strauss and early Barthes. Also, the same sign in a different system can have a different meaning, as illustrated by the way similar characters in different alphabets can take different meanings, e.g., in the Roman and Cyrillic alphabets, the token "P" denotes different sounds.

We formalize[1] sign system as many sorted loose algebraic theories with data, plus two additional items that are specifically semiotic:

Definition 1 *A* **sign system***, also called a* **semiotic system** *or* **semiotic theory***, consists of:*

1. *a* **signature***, which declares sorts, subsorts and operations (including constructors and selectors);*
2. *a subsignature of* **data sorts**[2] *and* **data functions***;*
3. *axioms (e.g. equations) as constraints;*
4. *a* **level ordering** *on sorts, including a maximum element called* **top***; and*
5. *a* **priority ordering** *on constructors at the same level.*

The non-data sorts classify signs and their parts, just as in grammar, the "parts of speech" classify sentences and their parts. There are two kinds of operation: constructors build new signs from old signs as parts, while selectors pull out parts from compound signs. Data sorts classify a special kind of sign that provides values serving as attributes of signs. Axioms act as constraints on what count as allowable signs for this system. Levels indicate the whole/part hierarchy of a sign, with the top sort being the level of the whole; priorities indicate the relative significance of subsigns at a given level; social issues play a dominant role in determining these. The above definition follows [Goguen, 1999a], where the special treatment of data sorts follows [Goguen and Malcolm, 2000]. The first four items constitute what is called an **algebraic theory** when all axioms are equations (e.g., [Goguen and Malcolm, 1996, Goguen et al., 1978]); it can be proved that this special case is sufficient for our needs.

The approach of Definition 1 differs from the more traditional set-based approaches of Gentner [Gentner, 1983], Carroll [Carroll, 1982], etc., in that it is *axiomatic*, i.e., it does *not* present signs as particular models, but rather, a particular theory expressed in a formal language describes a space of possible signs, which are *models* of the theory, in the sense of that term in logic, providing concrete *interpretations* for the things in the theory: sorts are interpreted as sets; constant symbols are interpreted as elements; constructors are interpreted as functions, etc.; i.e., the theory is a *language* for talking about such models. This approach allows both *multiple models* and *open structure*, both of which are important for applications. The first point means, for example, that a semiotic space of books should allow anything having the structure of a book as a model; it also means that designers and implementers have the freedom to optimize implementations so long as they respect the constraints of

[1]Due to the nature of this paper, sign systems are not fully formalized, and in particular, signatures are treated rather informally, because they are sufficiently complex that a formal definition would distract from the flow of ideas; see [Goguen and Malcolm, 1996, Goguen et al., 1978] for the formal definition of signature, and see [Goguen, 1999a] for the formal definition of sign system.

[2]These are for fixed data types like integers, booleans and colors, which are always interpreted in a standard way.

the given axioms. The second point says that structure can be extended and combined without violating the specifications, which is not necessarily the case for models. For example, we might want to extend a basic sign system for books with some further structure for a certain series of books from a particular publisher. In addition, it is also more natural to treat levels and priorities in axiomatic theories than in set-based models.

For an example of axioms, in formalizing indices of books, we might well want to impose an axiom requiring that indexed items must be phrases of 1, 2 or 3 words, but not more. In a sign system for books, the top level might be occupied by the sort for books, the next level by author, title, publisher, and the third level by the first and last names of author, and by the name and location of the publisher; see Figure 1. Here last name has priority over first name, and publisher name has priority over publisher location. This is similar to the nesting structure used in XML documents.

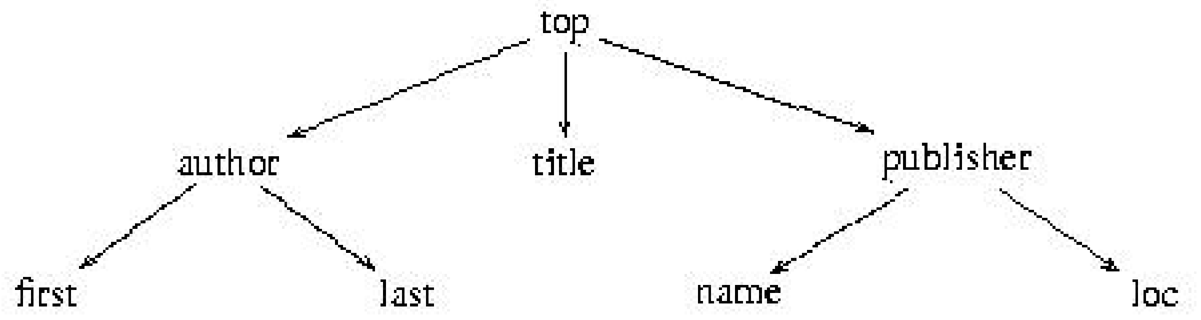


Figure 1: Levels for the Book Semiotic Theory

The following are some further informal examples of sign systems: dates; times; bibliographies (in one or more fixed format); tables of contents (e.g., for books, again in fixed formats); newspapers (e.g., the *New York Times* Arts Section); and a fixed website, such as the CNN homepage (in some particular instance of its gradually evolving format). Note that each of these has a large space of possible instances, but a single fixed structure.

There is a basic duality between theories and models. We have already discussed one aspect: A semiotic theory determines the class of models that satisfy it, which we call its **semiotic space**[3]. The other aspect is more subtle: a class of models has a unique (up to equivalence) most restrictive theory whose models include it[4]. This duality helps to justify our occasional use of the term “space” when we really mean “theory”; this is mainly done for consistency of terminology when discussing conceptual blending theory.

Gilles Fauconnier introduced **mental spaces** for studying meaning in natural language from a cognitive point of view [Fauconnier, 1985]. The abstract mathematical structure of a mental space is a set of atomic elements together with a set of relation instances among those elements [Goguen, 1999a], and as such is a very special case of a sign system. Any such representation necessarily omits the qualitative, experiential aspects of what is represented (these aspects are often called “qualia”), since formal representations cannot capture meaning in any human sense. Moreover, mental spaces are not powerful enough for designing virtual worlds or other applications where *structure* and *dynamics* are important; obvious examples include wikis, web sites, and music.

[3]This use of the word “space” conflicts with cognitive linguistics, where conceptual spaces, which are discused below.

[4]This duality is a Galois connection between algebraic theories and their models; it does not involve the levels or priorities.

The conceptual spaces of Fauconnier and Turner [Fauconnier and Turner, 1998, Fauconnier and Turner, 2002] are mental spaces, and hence share their limitations. For example, conceptual space theory can help us understand *concepts about* music, but semiotic spaces and structural blending are needed for an adequate treatment of the *structure of* music, e.g., how a melody can be combined with a sequence of chords. Conceptual spaces are good for talking about concepts about (e.g., how we talk about) things, but are awkward for talking about the structure of things. It is also interesting to notice that greater cultural variation can be found in conceptual blending than in structural blending, because the former deals with concepts about something, whereas they latter deals with the structure of its instances and/or its representations. Mathematically, a conceptual space is a single model, consisting of items and assertions that certain relations hold of certain of those items; it is not a theory or a class of models.

Our suggested methods for determining semiotic spaces are grounded in ideas from sociology, especially ethnomethodology, but this paper is not the right place to discuss such issues; see [Goguen, 1997, Goguen, 1994], beyond noting that *semiosis*, which is the creation of meaning, is always situated and embodied, and in particular always has a social context. Immersion arises in part through embodiment (even if only metaphorical embodiment, e.g. in text-based virtual worlds).

2.2 Representations

Mappings between structures became increasingly important in twentieth century mathematics and its applications; examples include linear transformations (and their representations as matrices), continuous maps of spaces, differentiable and analytic functions, group homomorphisms, and much more. Mappings between sign systems are only now appearing in semiotics, as uniform representations for signs in a source space by signs in a target space. Since we formalize sign systems as algebraic theories with additional structure, we should formalize **semiotic morphisms** as theory morphisms; however, these must be *partial*, because in general, not all of the sorts, constructors, etc. are preserved in real world examples. For example, the semiotic morphism that produces an outline from a book, omits the sorts and constructors for paragraphs, sentences, etc., while preserving those for chapters, sections, etc. In addition to the formal structure of algebraic theories, semiotic morphisms should also (partially) preserve the priorities and levels of the source space. The extent to which a morphism preserves the various features of semiotic theories is important in determining its quality, as the case studies to follow will show.

The design of virtual worlds and more generally, of user interfaces, is the art of creating representations, for example, representing the functionality of an operating system using icons, menus, buttons, etc., or using haptics and virtual reality. The basic insight is that a representation is a mapping $M: S_1 \to S_2$ of sign systems that preserves as much as is reasonable. The following formalizes this insight:

Definition 2 *A* **semiotic morphism** $M: S_1 \to S_2$ *from a semiotic system* S_1 *to another* S_2 *consists of the following partial mappings:*

1. *from sorts of* S_2 *to sorts of* S_2*, so as to preserve the subsort relations,*
2. *from operations of* S_2 *to operations of* S_2*, so as to preserve their source and target sorts,*
3. *from levels of* S_1 *sorts to levels of* S_2*, so as to preserve the ordering relation, and*
4. *from priorities of* S_1 *constructors to priorities of* S_2 *constructors, so as to preserve their ordering relations,*

so as to strictly preserve all data elements and their functions.

It is not always possible or even desirable for a semiotic morphism to preserve everything. For example, sometimes we just want to *summarize* some dataset, e.g., the table of contents of a book, in which case much of the structure and information are intentionally deleted. Another important observation is that not all representations are equally desirable. For example, one way to parse the sentence "Time flies like an arrow" in the following "bracket" (or "bracket-with-subscript") notation, which is widely used in linguistics:

$$[[time]_{\mathrm{N}}[[flies]_{\mathrm{V}}[[like]_{\mathrm{P}}[[an]_{\mathrm{Det}}[arrow]_{\mathrm{N}}]_{\mathrm{NP}}]_{\mathrm{PP}}]_{\mathrm{VP}}]_{\mathrm{S}}$$

However, this notation is not very satisfactory for humans, who would find it easier to discern the syntactic structure by examining a parse tree, or using the algebraic "constructor" notation given later. Some criteria for judging the quality of representations are discussed in Section 2.4 below.

The duality between theories and models means that there is an inherent ambiguity about the direction of a semiotic morphism. For example, if B is is a sign system for books and T is one for tables of contents, then books (which are models of B) are mapped to their tables of contents, which are models of T, but this map on models is determined by, and is dual to, the theory inclusion $T \to B$, which expresses the fact that the structure of tables of contents is a substructure of that of books. In informal discussions we will often take the direction to be that of the models, which is perhaps more intuitive; however, in formal discussions, it is much better to use the direction of the underlying theory morphism, which is opposite to that of the models.

There are at least three "modes" in which one might consider representations: analytic, synthetic, and conceptual. In the *analytic* mode, we are given one or more sign from the representation (i.e., the target) space, and we seek to reconstruct both the source space and the representation. In the *synthetic* mode, we are given the source space and seek to construct a good representation for the signs in that space, using some given technology (such as command line, or standard GUI widgets, or virtual reality) for the target space. In the *conceptual* mode, we seek to analyze the metaphorical structure of the representation, in the style of cognitive linguistics [Turner, 1997, Fauconnier and Turner, 2002]; for example, how is Windows XP like a desktop, or how is a scrollbar like a scroll? A treatment in this mode will involve conceptual spaces, in the sense of cognitive linguistics (see Section 3.1 below). In each mode, particular aspects of the cultures involved can be very significant.

2.3 Simple Examples

This subsection gives rather informal descriptions of some simple examples of semiotic theories and semiotic morphisms, to illustrate the concepts, rather than to demonstrate their applicability to virtual world design, since these examples could only be considered virtual worlds in a trivial sense. The following sign systems are considered:

1. Lists of (potential) words with punctuation, denoted S_W.
2. Parse trees for sentences of a formal grammar G, denoted S_T.
3. A printed page format, denoted S_P.

Then the following are some interesting morphisms:

1. Let $P\colon\ S_W \to S_T$ give parse trees for lists from S_W that are G-sentences.
2. Let $H\colon\ S_T \to S_P$ give bracket representations of parse trees.
3. Let $F\colon\ S_W \to S_P$ give bracket representations of lists from S_W that are G-sentences.

The morphism F is "composed" from P and H, by first doing P and then doing H; we denote this by $P; H$, where ";" denotes the composition operation. Composing morphisms corresponds to composing representations, which is the essence of iterative design, an important technique for any complex design task. By Definition 2, a semiotic morphism M has four component mappings, for sorts, operations, levels, and priorities; let us denote these M_1, M_2, M_3 and M_4, respectively. Then the **composition** $M; M'$ of morphisms M and M' can be defined by the formula $(M; M')_i = M_i; M'_i$ for $i = 1, 2, 3, 4$.

The sign system S_W for punctuated lists of words can be described roughly as follows: Its sorts are `char`, `alpha`, `punc`, `puncword`, `alphaword`, `word`, and `list`, where the sorts `alpha` and `punc` are subsorts of `char`, the sorts `alpha`, `puncword` and `alphaword` are subsorts of `word`, and the sort `word` is a subsort of `list`. These subsort relationships are shown in Figure 2. The sorts `char`, `alpha`, and `punc` are respectively for character, alphabetic character, and punctuation character; the sorts `puncword`, `alphaword`, and `word` are respectively for words consisting of alphabetic characters with a final punctuation sign, words with all alphabetic characters, and the union of these two.

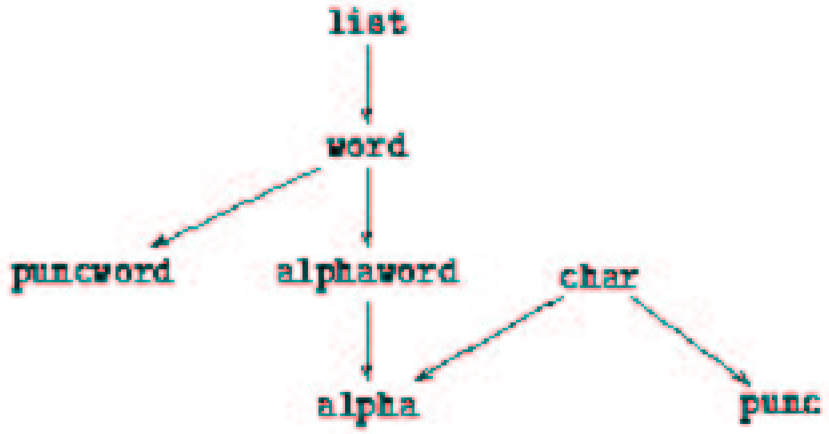


Figure 2: Sorts and Subsorts for S_T

The sort `list` is level 1, the "top" sort of this system; `word` is level 2; `alphaword` and `puncword` are level 3; and `char`, `alpha` and `punc` are level 4. The punctuation characters are comma and period (of course we could add more). The following defines concatenation constructors for constructing a list of alphabetic characters as a `alphaword`, a list `alphaword` followed by a punctuation as a `puncword`, and a list of `word`s as a `list`; a functional notation is used:

```
_ _ : alpha alphaword -> alphaword
_ _ : alphaword punc  -> puncword
_ _ : word list       -> list
```

Here the two underbars give the syntactic form of the function, which is to concatenate its two arguments, which respectively have the two types given between the colon and the arrow; the result type then comes after the arrow. These three operations also satisfy associativity equations, such as the following:

$$(\forall WLL')\ (WL)L' = W(LL')\,.$$

The context free grammar grammar G given below allows some lists of punctuated words to be recognized as legal sentences of that grammar; such sentences can be parsed, which means dividing them into **phrases**, which are sublists, each with its "part of speech" (or syntactic category) explicitly given. The grammar G will become the signature of the sign system S_T. The non-terminals of G are `S`, `NP`, `VP`, `N`, `Det`, `V`, `PP` and `P`, which stand for sentence, noun phrase, verb phrase, noun, determiner, verb, prepositional phrase, and preposition, respectively. Then the rules of a simple example G might be the following:

```
S  -> NP VP
NP -> N
NP -> Det N
VP -> V
VP -> V PP
PP -> P NP
```

The non-terminals of this grammar (i.e., the "parts of speech" `S`, `NP`, `VP`, etc.) are the sorts of the sign system S_T, and the rules of the grammar will become its constructors. For example, the first rule says that a sentence can be constructed from an `NP` and a `VP`. There should also be some constants of the various sorts, such as

```
N -> time
N -> arrow
V -> flies
Det -> an
Det -> the
P -> like
```

Then, for example, a parse tree for the sentence "time flies like an arrow" is shown in Figure 3.
By the way, if we add the productions below to the grammar G, then the sentence gets another parse, a fact which the reader might enjoy checking.

```
NP -> N N
V  -> like
```

There is a systematic way to convert context free rules into constructor operations in the signature of a sign system; for the above grammar G, it is as follows, written in a functional notation:

```
  sen : NP VP -> S
  nnp : N     -> NP
  np  : N Det -> N
  vvp : V     -> VP
  vp  : V PP  -> VP
  pp  : P NP  -> PP
 time :       -> N
flies :       -> V
  .....
```

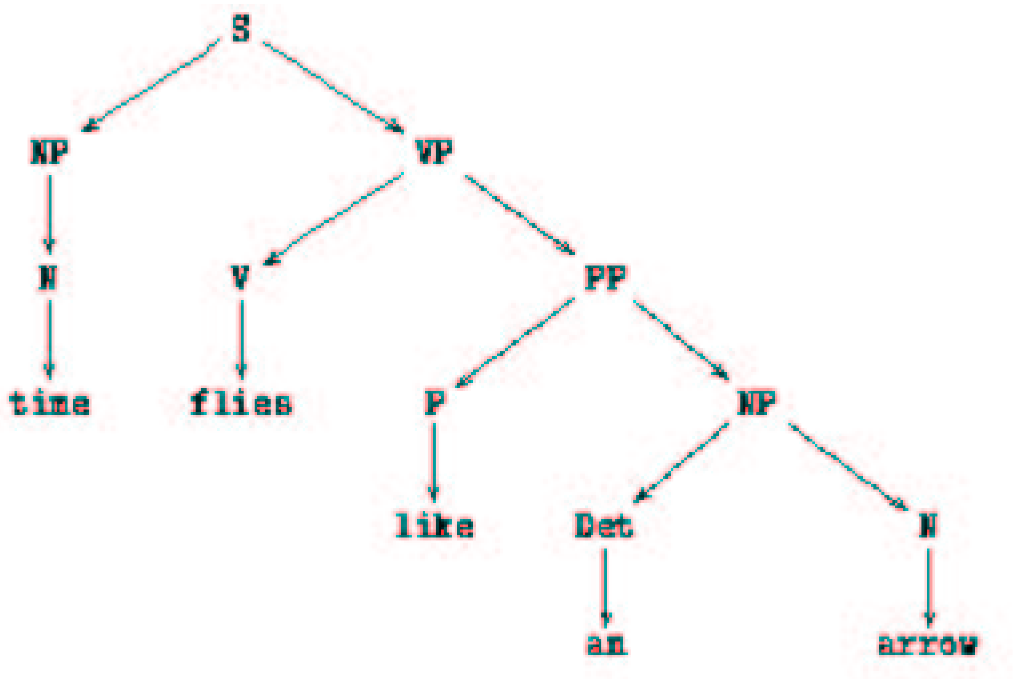


Figure 3: Parse Tree for S_T

In this context, it is more elegant to regard `N` as a subsort of `NP`, and `V` as a subsort of `VP`, rather than to have monadic operations `N -> NP` and `V -> VP`. This sign system gives what computer scientists call *abstract syntax* for sentences; it gives an abstract algebraic representation for syntactic structure, in which the operations above generate a *free algebra* of terms that describe parses. For example, the term that represents the syntax of our example sentence is:

```
    sen(time, vp(flies, pp(like, np(an, arrow)))).
```

Equations can be used in this algebraic setting to express constraints on sentences, for example, that the number of the subject and of the verb agree (i.e., both are singular or else both are plural). Each of the concrete ways to realize abstract syntax (trees, terms, bracket notation, and lists) can be considered to give a model of the sign system S_T, providing a set of signs for each sort, and operations on those sets which build new signs from old ones.

The sign system S_P should have sorts for lines and pages, and could also have different fonts and subscripts, in order to display the bracket notation to display parses. We omit the details, which are not very different from those above, except for an equation to limit the length of lines, e.g., to 80 characters, such as the following:

$$(\forall\, L\colon \texttt{line})\ \mathrm{length}(L) \leq 80\,.$$

The morphism $P\colon\ S_W \to S_T$ is very partial, since it is defined on a list ℓ iff ℓ can be parsed using G; thus the subset of lists on which it is defined is the set of sentences generated by G, which is usually denoted $L(G)$. If $fr(t)$ denotes the **frontier**, or list of leaf nodes, of a parse tree t, then $fr(P(\ell)) = \ell$ for all $\ell \in L(G)$, which is a strong preservation property, although it only holds on a small subset of lists of words. The morphism P also preserves all of the sort hierarchy in Figure 2.

We do not describe the morphism $H\colon\ S_T \to S_P$ in as much detail as we did the morphism $P\colon\ S_W \to S_T$. The morphism H is essentially a pretty printer for parse trees; it could use any of the representations we have been discussing, and it could even just print the frontier of the parse tree, although this would preserve much less structure (see the next subsection for more discussion of structure preservation). As already mentioned, the morphism $F\colon\ S_W \to S_P$ is the composition $P;H$; it pretty prints the parse trees of those lists of words that can be parsed by G.

2.4 Quality of Representation

It is easy to define sort preserving, constructor preserving, level preserving, content preserving (where **content** refers to the values of selector operations, such as size and color), etc. But this is not as useful as one might hope, because in practice, these are often *not* preserved. Instead, we define the *comparative* notions of more sort preserving, more level preserving, more constructor preserving, more content preserving, etc. [Goguen, 1999a]. These notions define orderings on morphisms, which can be logically combined to get the right one for a given application [Goguen, 1999a]. This is important because given morphisms M, M', one may preserve more levels, while the other preserves more content, and similarly for the other concepts. Empirical work has validated the following general principles:

1. It is more important to preserve structure than content (this is called **Principle F/C**).
2. It is more important to preserve level than priority.
3. Structure and content at lower levels should be sacrificed in favor of those at higher levels.
4. Lower level constructors should be sacrificed in favor of higher level constructors.

The first principle is perhaps the most important, and at first might seem counter-intuitive, many special cases can be found in the design literature, e.g., [Tufte, 1983]. It asserts that when a trade-off is necessary, form should be weighted more heavily than content; in general, the right balance between form and content can only be determined after knowing how a representation will actually be used. Also, we are fortunate that it is easier to describe structure than content.

These principles do not explain everything, for example, they do not explain why the tree representation of phrase structure is better than the bracket representation, since these two representations have exactly the same structure and content, but display them differently. In fact, the advantage of the tree representation arises from human visuo-cognitive capabilities, which prefer a more explicit diagrammatic representation of phrases and subphrase relations as nodes and edges, over a linear symbolic representation that requires counting brackets. Preservation of form and content can respectively be formalized as preservation of constructors and selectors, in the sense of abstract data type theory [Goguen et al., 1978, Goguen and Malcolm, 1996].

3 Fragments of a Calculus of Representation

Section 2.3 defined the composition of semiotic morphisms and showed that it can be important for applications. It is easy to prove that this definition of composition obeys the following identity and associative laws, in which $A\colon\ R \to S,\ B\colon\ S \to T$ and $C\colon\ T \to U$,

$$A\,;1_S = A$$
$$1_S\,;B = B$$
$$A\,;(B\,;C) = (A\,;B)\,;C$$

where 1_S denotes the identity morphism on S. These three laws are perhaps the most fundamental for a calculus of representation, since they imply that semiotic theories and their morphisms form what is called a "category" in the relatively new branch of mathematics called *category theory* [Mac Lane, 1998]. The basic ingredients of a **category** are objects, morphisms, and a composition operation that satisfies the above three laws, and that is defined on two morphisms if and only if they have matching source and target. Perhaps surprisingly, many important mathematical concepts can be defined abstractly in the language of category theory, without reference to how objects are represented, using only morphisms and composition; moreover, many general laws can be proved about such concepts, and these automatically apply to every category.

Three of the simplest categorical concepts are isomorphism, sum and product. A morphism $A\colon\ R \to S$ is an **isomorphism** if and only if there is another morphism $B\colon\ S \to R$ such that $A\,;B = 1_R$ and $B\,;A = 1_S$, in which case B is called the **inverse** of A and denoted A^{-1}; it can be proved that the inverse of a morphism is unique if it exists. The following laws can also be proved, assuming that $A\colon\ R \to S$ and $B\colon\ S \to T$ are both isomorphisms (and no longer assuming that B is the inverse of A).

$$1_R^{-1} = 1_R$$
$$(A^{-1})^{-1} = A$$
$$(A\,;B)^{-1} = B^{-1}; A^{-1}$$

Because sign systems and their morphisms form a category, these three laws apply to representations. In Section 3.2 below, we discuss sums of semiotic morphisms as a special case of blends of semiotic morphisms (blends are discussed in the next subsection), and we also give some laws for blends and sums in Section 3.2. The standard mathematical reference for category theory is by Sanders Mac Lane [Mac Lane, 1998], but [Pierce, 1990] is one computer science oriented introduction, among several others.

3.1 Metaphor and Blending

Research in cognitive linguistics by George Lakoff and others under the banner of "conceptual metaphor theory" (abbreviated "CMT") has greatly deepened our understanding of metaphor [Lakoff and Johnson, 1980, Lakoff, 1987], showing that many metaphors come in families, called **image schemas**, that share a common pattern. One example is BETTER IS UP, as in "I'm feeling up today," or "He's moving up into management," or "His goals are higher than that." Some image schemas, including this one, are grounded in the human body[5] and are called **basic image schemas**; they tend to yield the most persuasive metaphors, as well as to enhance the sense of immersion in virtual worlds.

Fauconnier and Turner [Fauconnier and Turner, 1998, Fauconnier and Turner, 2002] study **blending**, or **conceptual integration**, claiming it is a basic human cognitive operation, invisible and effortless, but nonetheless fundamental and pervasive, appearing in the construction and understanding of metaphors, as well as in many other cognitive phenomena, including grammar and reasoning. Many simple examples are blends of two words, such as "houseboat," "jazz piano," "roadkill," "artificial life," "computer virus," "blend space," and "conceptual space." To explain such phenomena, blending theory (abbreviated "BT") posits that concepts come in clusters, called **conceptual spaces**, which consist of

[5]The source UP is grounded in our experience of gravity, and the schema itself is grounded in everyday experiences, such as that when there is more beer in a glass, or more peanuts in a pile, the level goes up, and that this is a state we often prefer; therefore the image schema MORE IS UP, discussed in [Lakoff, 1987], is even more basic.

certain items and certain relations that hold among them. Such spaces are relatively small constructs, selected on the fly from larger domains, to meet an immediate need, such as understanding a particular phrase or sentence[6]. The abstract mathematical structure of a conceptual space consists of a set of atomic elements together with a set of relation instances among those elements [Goguen, 1999a]. **Conceptual mappings** are partial functions from the item and relation instances of one conceptual space to those of another, and **conceptual integration networks** are networks of conceptual spaces and mappings that are to be blended together.

$$\begin{tikzcd}
 & B & \\
I_1 \arrow[ur] & & I_2 \arrow[ul] \\
 & G \arrow[ul] \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

Figure 4: A Blend Diagram

We now describe our generalization of blending from conceptual spaces to semiotic theories. A simple example where this generality is needed is in the integration of a window with its scrollbar, which is structural, not conceptual, although conceptual aspects of this blend could also be studied; this example is discussed in considerable detail in [Goguen, 2003]. To indicate this added generality, we will use the terms **structural blending** or **structural integration** for the blending of semiotic systems, which in general involves non-trivial constructors; but for consistency with BT, we use the phrase "semiotic space" instead of "semiotic theory" in this discussion. The simplest form of blend is as shown in Figure 4, where I_1 and I_2 are called the **input spaces**, B the **blend space**, and G the **generic space**, which contains conceptual structure that is shared by the two input spaces[7]. Let us call I_1, I_2, and G together with the two morphisms $G \to I_1$ and $G \to I_2$ an **input diagram**. Then a **blendoid** over a given input diagram is a space B together with morphisms $I_1 \to B$, $I_2 \to B$, and $G \to B$, called **injections**, such that the diagram of Figure 4 **weakly commutes**, in the sense that both compositions $G \to I_1 \to B$ and $G \to I_2 \to B$ are **weakly equal** to the morphism $G \to B$, in the sense that each element in G gets mapped to the same element in B under them, provided that both morphisms are defined on it[8]. The special case where all four spaces are conceptual spaces gives conceptual blends. This diagram is "upside down" from that used by Fauconnier and Turner, in that our arrows go up, with the generic G on the bottom, and the blend B on the top. Our convention is consistent with duality mentioned earlier, as well as with the way that such diagrams are usually drawn in mathematics, and with the image schema MORE IS UP (since B is "more"). Also, Fauconnier and Turner do not include

[6]However, we do not assume that they are necessarily the **minimal** such spaces needed to understand a given blend, since that can only be determined after the blend has been understood. Moreover, different blends may ignore different elements of the input spaces, and it may also be necessary to recruit additional information from other spaces in order to understand a blend.

[7]The term "**base space**" is used in [Goguen, 1999a], because it is considered to better describe how this theory is used in applications to user interface design.

[8]Strict commutativity, usually called just **commutativity**, means that the compositions are strictly equal, i.e., one morphism is defined on an element if and only if the other is, and then they are equal.

the map $G \to B$. By definition, the maps $G \to I_1$ and $G \to I_2$ are total, not partial, and if the input spaces were minimal, then the maps $I_1 \to B$, $I_2 \to B$, $G \to B$ would also be total.

Usually an input diagram has many blendoids, only a few of which are interesting. Weak commutativity of the blend diagram, which is included in the definition, is a good first step, but still leaves too many possibilities. Therefore additional principles are needed for identifying the most interesting possibilities, so that we can define a **blend** to be a blendoid that is *optimal* with respect to these principles. Fauconnier and Turner suggest a number of "optimality principles" that serve this purpose (see Chapter 16 of [Fauconnier and Turner, 2002]), but they are too vague to be easily formalized. A tentative and difficult but precise mathematical approach is given in Appendix B of [Goguen and Malcolm, 1996], based on a modification of the category theoretic notion of "pushout" [Mac Lane, 1998]; this modification takes advantage of an ordering relation on morphisms, along the lines discussed in Section 2.4. The intuition is that nothing can be added to or subtracted from such an optimal blendoid without violating consistency or simplicity in some way. However, there can still be more than one blend in this sense, as an example discussed below will make very clear. It should also be noted that this notion of blend easily generalizes to any number of semiotic spaces, and even to arbitrary diagrams of semiotic spaces and morphisms, for which there are many significant applications. Thus, the emphasis on double scope blending in [Fauconnier and Turner, 2002] seems somewhat out of place in algebraic semiotics, because its major applications typically involve multiple "scopes" arising from multiple spaces and morphisms among them.

It has perhaps not been sufficiently emphasized in the BT literature that blending does not always give a unique result. For example, the following are four different blends of conceptual spaces for "house" and "boat":

1. houseboat;
2. boathouse;
3. amphibious RV; and
4. boat for moving houses.

The last may be a bit surprising, but I once saw such a boat in Oban, Scotland, transporting prefabricated homes to a nearby island. There are also some other, even less obvious blends [Goguen and Harrell, 2004].

In the UCSD Meaning and Computation Lab, Fox Harrell and I have been experimenting with a blending algorithm, which has generated novel metaphors, which in turn were used in generating poems [Goguen and Harrell, 2004], with some success before a live audience. The algorithm uses dynamic programming to generate blends in approximate order of optimality, and if requested, can generate all possible blends, including even very bad ones. One surprise was that there were so many blends, for example, 48 for the (small) house and boat spaces.

The CMT view of metaphor associates aspects of one domain to another, and describes this association using a mapping, of which the target domain concerns what the metaphor is "about." On the other hand, BT views metaphors as "cross-space mappings" that arise from blending conceptual spaces extracted from the domains involved. For example, the metaphor "my love is a rose" arises from blending conceptual spaces for "my love" and "rose," such that the identification of the two items "love" and "rose" in the blend space gives rise to a correspondence between certain items in the rose space and the target love space. Such **metaphoric blends** are *asymmetric*, in that as much as possible of the target space is imported into the blend space, whereas only key aspects from the source space, associated with elements that have been identified with elements of the target space, are imported, e.g., sweet smell and attractive color; moreover, names from the top space take precedence over those in the source space, so that relations in the source space become “attributed” to items in the target space. Our approach differs from orthodox BT not only in that we allow many more kinds of structure in our spaces, but also in that we do not first construct a minimal image in the blend space and then “project” that material back to the target space, but instead, we construct the entire picture in the blend space. Thus it is not the case for us that, in forming the blend, elements are preferentially omitted from the target space, only to be restored upon projection, as described in [Grady et al., 1999]. Since CMT has been mainly concerned with families of metaphors having a shared pattern, and BT has been more concerned with how novel metaphors can be understood, the two theories are compatible, and can both play a role in understanding complex language. This and related issues are discussed with many interesting details and examples in [Grady et al., 1999].

Algebraic semiotics also goes beyond conceptual spaces in allowing entities that have dynamic states; this is necessary for applications to the dynamic entities that appear in user interface design and virtual worlds. Actually, two kinds of dynamics are involved in blending: the process of blending itself, and entities with internal states. Whereas cognitive linguistics has so far focussed mainly on the former, algebraic semiotics is more concerned with the latter. Another difference from BT is that relations like causality are represented as ordinary relations rather than being given a special *ad hoc* status.

The conceptual spaces, mappings and blending of cognitive linguistics seem well adapted for treating many aspects of literature, as in [Turner, 1997], as well as some recent trends in art, including (the very aptly named) conceptual art movement, and with the conceptual aspects of works in many other styles, which are often designed to provoke conceptual conflicts or to force unusual conceptual blends. One important application is the combination of music with lyrics, as skillfully studied using cross-domain mappings by Lawrence Zbikowski [Zbikowski, 2002]. Unfortunately, the framework of conceptual blending seems too restricted for studying blending *within* music, e.g., harmony, polyphony, polyrhythm, etc., because musical structure is inherently hierarchical, and hence cannot be adequately described using only atomic elements and relation instances among them. Understanding how a particular melody, chord sequence, and rhythm can work together requires attention to the component notes, phrases, chords and beats, as well as to their subcomponents. However, it seems that the added generality of semiotic spaces and semiotic morphisms is adequate for such purposes.

3.2 Some Laws

This subsection gives some further fragments of a calculus of representation; see [Goguen, 1999a] for more detail. Here a, b, c are semiotic morphisms, and $\diamond$ denotes some choice of a blend that is maximal with respect to some optimality criterion:

$$\begin{aligned} a \diamond b &\cong b \diamond a \\ a \diamond (b \diamond c) &\cong (b; a) \diamond c \\ (a \diamond b) \diamond c &\cong a \diamond (b; c) \end{aligned}$$

In the following, A, B, C may be either semiotic morphisms or just semiotic systems. Sums, denoted $+$, are the special case of blend where the base theory is $\mathbf{1}$, which is the theory having exactly one constant, its top element, and nothing else.

$$\begin{aligned} A + \mathbf{1} &\cong A \\ \mathbf{1} + A &\cong A \\ A + B &\cong B + A \\ A + (B + C) &\cong (A + B) + C \end{aligned}$$

It should be noted that products of models correspond to sums of theories, that is, a model of a sum of theories is a product of models of the summand theories, and *vice versa*, or even more formally, there is an isomorphism of categories of models,

$$Mod(A + B) \cong Mod(A) \times Mod(B)\,,$$

where A and B are semiotic theories; see [Goguen, 1999a] for details.

4 Case Studies

This section surveys some case studies applying algebraic semiotics. Noting that we have already discussed blending and metaphor, the following additional case studies are considered:

1. information visualization;
2. proof presentation;
3. humor; and
4. user interaction.

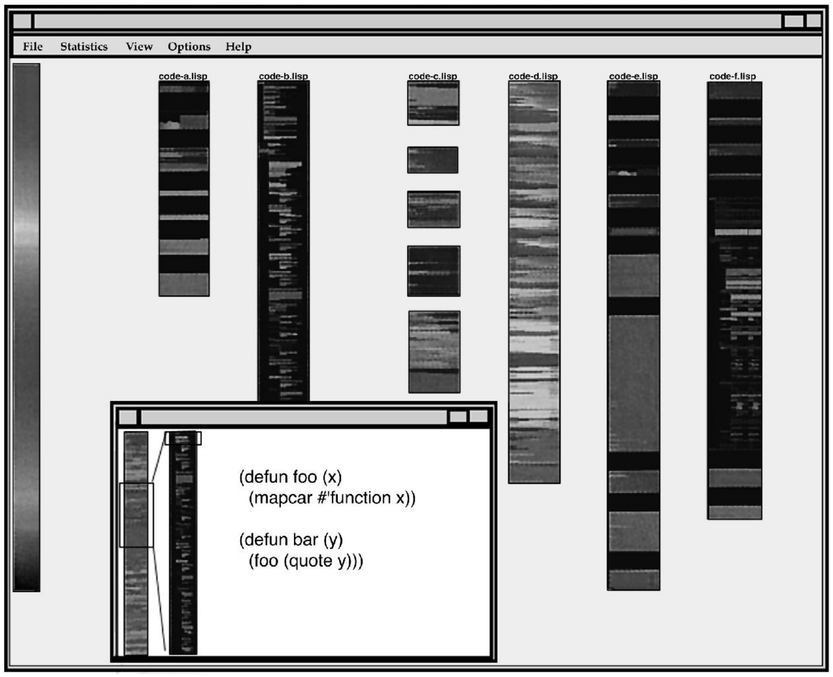
The first category in this list actually contains three small case studies, and the second can also be considered a special case of it; proof visualization is our most extensive case study, part of a large project to produce a web-based system to support theorem proving. The study of humor is somewhat of a digression, but it is hoped that the reader will find it amusing. Proof navigation is used to illustrate how interaction is treated in algebraic semiotics, although many details are left out, because the formal theory of dynamic signs is technically rather complex.

4.1 Information Visualization

Visualizing complex data can help to discover, verify and predict patterns, and to quickly locate specific information; but it can be difficult to construct the appropriate visualizations for these purposes. Because visualizations are representations, our theory applies to them, and in particular, our quality measures apply. The following subsections analyze three real visualization systems as semiotic morphisms, and on that basis, suggest some improvements. We found it convenient to use algebraic semiotics in a semi-formal style, letting the ideas and results guide the re-design, and introducing formal details only to the degree that they actually help with decisions. Many aspects of these discussions follow [Goguen and Harrell, 2003].

4.1.1 Code Visualization

A visualization tool for code developed at ATT Bell Labs, and discussed in [Shneiderman, 1997], displays the large grain structure of code by omitting details; see Figure 5. This is an excellent illustration of Principle F/C (Section 2.4): commands are indistinguishable lines, but files and procedures are easily distinguished, and the age of code is highlighted with color (though it shows up as shades of gray in this figure), presumably because code age is so important for software maintenance, which accounts for most of the cost of large software systems. Moreover, code at the command level can be viewed in a separate window, which is activated by "zooming in" from the main overview window. However, software engineers often need to find other specific features of code, such as:



```
(defun foo (x)
  (mapcar #'function x))

(defun bar (y)
  (foo (quote y)))
```

Figure 5: A Code Browser

1. occurrences of particular variables,
2. certain uses procedure calls, and
3. certain uses of pointers.

Or consider what would be needed to work on the Y2K problem. To support this kind of flexibility, the system should allow users to select and highlight a variety of features to be displayed with color, not just code age; indeed, each feature listed above could be highlighted with a different color, because these features are binary (i.e., they either occur or do not occur, at any given point in the program), rather than, like age, being measured on a (nearly) continuous scale.

4.1.2 A Film Visualizer

Figure 6 shows FilmFinder, a system to help consumers find films, designed in Ben Shneiderman's group at the University of Maryland, as described in [Shneiderman, 1998]. The vertical axis indicates popularity, the horizontal axis indicates the release date, and the color[9] indicates the genre; the area on the right side of the display is for controlling the system. This complex sign is the image under an appropriate semiotic morphism of a sign in a space of information about films. From this, we infer that the designer of the system thought users would consider the popularity, date, and genre to be the most important attributes of films.

Instead of thinking of it as a consumer product, it is interesting to think of this system as a scientific tool for displaying data about the movie industry. Using it in this way, we can see that the density of films increases rapidly in the most recent years displayed, except perhaps for those genres that are the

[9]But as before, gray tones appear in our rendition of the display.

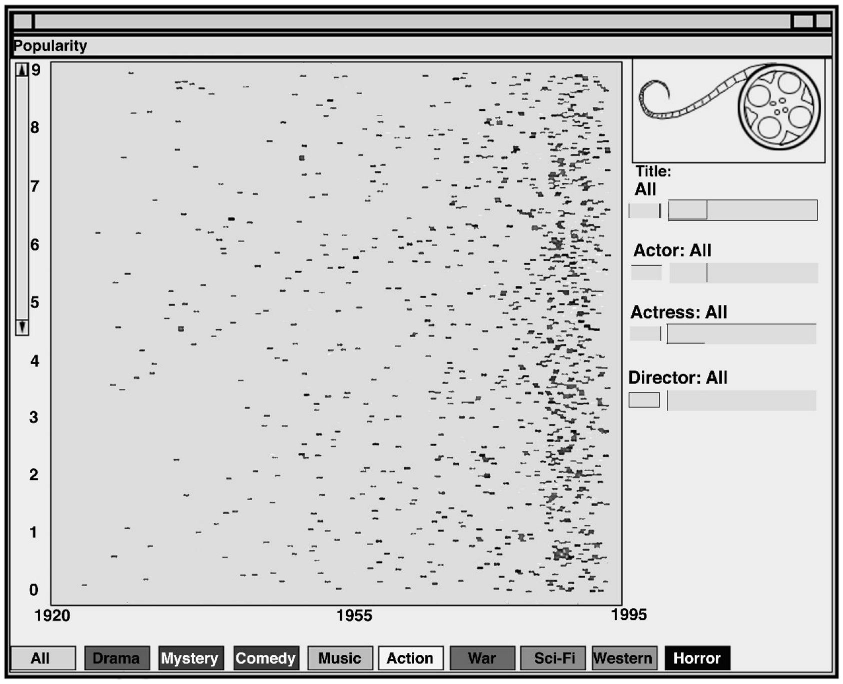


Figure 6: FilmFinder Display

least popular; and we can also easily see some 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 less useful than it could be for this purpose. The problem is again that too much content and not enough structure have been preserved. For example, it would be 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 graphically in a new popup window. These revisions would facilitate hypothesis formation, and would also make the tool more useful for consumers, especially when (as in the most recent years that are not represented in the figure) there are many more films.

4.1.3 A Later Version

Figure 7 shows a later version (SpotFire from IVEE Development in Sweden) of the FilmFinder tool in Figure 6; the main improvement is that users have more control over what is displayed and how it is displayed. This particular display has length and date as its axes, and again uses color for genre, although the genre color coding scheme is not explicitly shown; prize winning films are highlighted by having a larger size. It is interesting to observe a clustering at around 90 minutes length. But once again, the display is difficult to use because there are too many dots, even though this display cuts off at 1990! If the user is seeking a particular film or class of films, she will want to narrow the search focus by imposing additional constraints, but from this single display, it is difficult to know how easily that could be done. We are presumably supposed to assume that the (possibly imaginary) user who

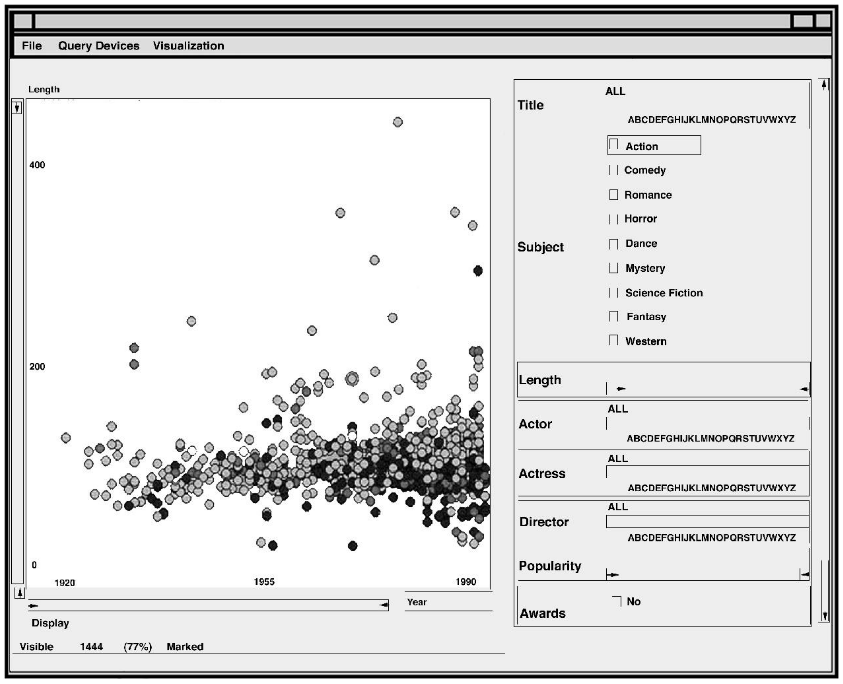


Figure 7: SpotFire Version of FilmFinder

created this display considered these particular attributes the most interesting at a certain point during a sequence of displays constituting a search; but in fact, they do not seem especially useful for any particular purpose.

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 can hope that these were determined by polling an adequate pool of typical users. But the key issue is how convenient these controls are for scenarios that typical users find particularly important; most likely, those typical users are looking for a good video to rent, rather than analyzing trends in the movie industry, and so the controls should reflect the key actions involved in those searches, rather than just the most important general attributes of films. It would take some experimental work to determine these most relevant search attributes, 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 also criticize the fine grain control that it gives users over length and year, and suggest instead that soft constraints would be more appropriate; it also seems doubtful that length is a highly significant attribute for search. Moreover, we can criticize the design philosophy, advocating instead a more social 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, such as listing films that a user's friends have liked; Amazon has exploits similar strategies very successfully). Finally, we can note that the design ideas proposed to improve the previous version of this system also apply to the new version.

4.2 Proof Representation and Understanding

It is well known (perhaps *too* well known to many unhappy students) that understanding mathematical proofs can be very difficult. But *why* is it difficult? And *how* can this situation be improved? The

UCSD Tatami project [Goguen et al., 2000] aims to make proofs more interesting and even enjoyable to read, by viewing them as representations of their underlying mathematics; this lets us apply algebraic semiotics, including the theory of representation quality.

The Kumo system generates websites, based on user-provided proof sketches in a language called Duck. The pages are in XML, displayed using XSL style sheets, and can be viewed over the web using any browser. The complex signs that users actually see are called **proofwebs**, consisting of English phrases and sentences, mathematical signs, navigation buttons, formal input and output for a mechanical theorem prover, etc. [Goguen et al., 2000, Goguen, 1999b].

Our view of what constitutes the underlying mathematics to be displayed is unusual: we consider it to include not just the tree structure of proofs, decorated with formal sentences and rules, as is common among computer scientists and logicians, but also:

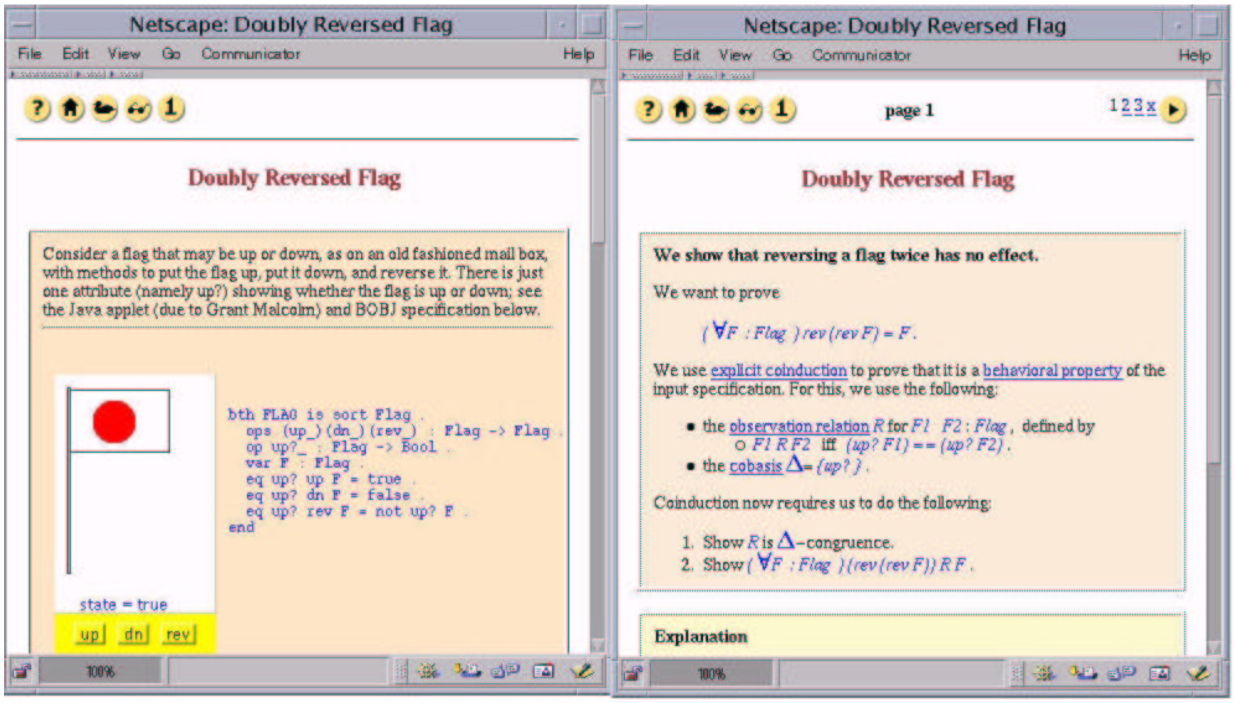
1. a dramatic structure, following Aristotle (see below);
2. a narrative structure (following ideas of Labov [Labov, 1972] and Linde [Linde, 1981], as briefly described below);
3. hyperlinks to related material, including tutorials for proof rules used, input and output to a formal theorem prover (if available), and motivation and explanation for proof strategies and steps; and
4. image schemas (in the sense of Lakoff [Lakoff and Johnson, 1980, Lakoff, 1987]). As with any virtual world, image schemas can make the language more direct and powerful, and hence easier to follow.

Aristotle said, "*Drama is conflict*" [Aristotle, 1997], which suggests providing conflict to add drama to proofs. Finding a non-trivial proof usually requires exploring many dead ends, errors and misconceptions, some of which may be very subtle. Therefore the process of proving can be full of disappointed hopes, unexpected triumphs, repeated failures, and even fear and interpersonal conflict. All this is typically left out when proofs are written up, leaving only the map of a path that has been cleared through the jungle. But proofs can be made much more interesting and understandable if some of the conflicts that motivated their difficult steps are integrated into their structure; proof obstacles are exactly what is needed for drama. Of course, this must be done with care, and it should not be overdone, just as in a good novel or movie. Our Kumo theorem proving system [Goguen et al., 2000] used these ideas to structure the websites that it generates to display proofs. Aristotle also gave other useful suggestions, including unity of time and place, and having a beginning, middle, and end to a drama [Aristotle, 1997].

William Labov [Labov, 1972] showed that oral narratives of personal experience have a precise internal structure, which includes the following:

1. an optional **orientation section**, which provides basic orientation information, such as the time and place of the story, and perhaps some major characters;
2. a sequence of **narrative clauses** which describe the events of the story;
3. the **narrative presupposition**, which by default assumes that the ordering of the narrative clauses corresponds to the temporal ordering of the events that they describe;
4. **evaluative material** interleaved with the narrative clauses, which "evaluates" the events, in the sense of relating them to socially shared values; and finally
5. an optional **closing** section, which may contains a "moral" or summary for the story.

The above follows [Linde, 1981, Linde, 1993], which describe developments subsequent to the classic treatment of [Labov, 1972]. Although this empirical research used oral narratives of personal experience as data, its results apply much more broadly (though in general less precisely), since the class of narratives is the core around which many discourse types are built.


Netscape: Doubly Reversed Flag
File Edit View Go Communicator Help
100%
Netscape: Doubly Reversed Flag
File Edit View Go Communicator Help
page 1
1 2 3 x
100%

Doubly Reversed Flag

Consider a flag that may be up or down, as on an old fashioned mail box, with methods to put the flag up, put it down, and reverse it. There is just one attribute (namely up?) showing whether the flag is up or down; see the Java applet (due to Grant Malcolm) and BOBJ specification below.

state = true

up dn rev

```
bth FLAG is sort Flag .
  ops (up_)(dn_)(rev_) : Flag -> Flag .
  op up?_ : Flag -> Bool .
  var F : Flag .
  eq up? up F = true .
  eq up? dn F = false .
  eq up? rev F = not up? F .
end
```

Doubly Reversed Flag

We show that reversing a flag twice has no effect.

We want to prove

$(\forall F : Flag)\, rev(rev\, F) = F$.

We use explicit coinduction to prove that it is a behavioral property of the input specification. For this, we use the following:

- the observation relation R for $F1\ F2 : Flag$, defined by
  - $F1\ R\ F2$ iff $(up?\ F1) == (up?\ F2)$.
- the cobasis $\Delta = \{up?\}$.

Coinduction now requires us to do the following:

1. Show R is Δ−congruence.
2. Show $(\forall F : Flag)(rev(rev\, F))\, R\, F$.

Explanation

Figure 8: A Typical Tatami Homepage and Proof Page

To aid our discussion of proofs as representations, we introduce terminology for the source and target semiotic spaces: let us call their elements **abstract proofwebs** and **proof displays**, respectively, and perhaps also use the term **display proofweb** for target signs. In addition, the term **unit** refers to a block of information of the same kind in a proof display. The display proofwebs generated by Kumo adhere to the following style guidelines, called the **tatami conventions** [Goguen et al., 2000]. The first eight are justified mainly by narratology:

1. **Homepages** are provided for every major proof part; homepages introduce and motivate the problem to be solved and the approach taken to the solution for that part, and correspond to the orientation sections of Labov's narrative structure; they may contain graphics, applets, and of course text. Homepages appear in the same window as their tatami pages (see next the item) because they are part of the same narrative flow. See Figure 8.
2. **Tatami pages**, also called **proof pages**, are the basic constituents of display proofwebs; they are XML pages containing one or more proof unit, with its inference rule applications, interleaved with one or more explanation units. This interleaving follows the interleaving of narrative and evaluative material in Labov's theory. Limiting the number of non-automatic proof steps on tatami pages to approximately 7 is justified by classic work of Miller on limitations of human cognitive capacity [Miller, 1956]; this limitation also makes it feasible to place both proof and explanation units on the same proof page. See Figure 8.
3. The **explanation units** of tatami pages are prover-supplied informal discussions of proof concepts, strategies, obstacles, etc. They correspond to the evaluative material in Labov's theory, and motivate important proof steps by relating them to values shared in the appropriate community of provers.

4. Tatami pages can be browsed in an order designed by the prover to be helpful and interesting to the reader; if possible, they should tell a story about how obstacles were overcome (or still remain). This narrative order again comes from Labov's theory, while including obstacles comes from work of Campbell [Campbell, 1973] and others on "heroic" narratives.
5. Major proof parts, including lemmas, have their own subwebsites, each with the same structure as the main proof, including homepage and explanation units. These appear in a separate dedicated persistent popup window. Having separate hyperlinked websites for major proof parts is similar to the way that flashbacks and other temporal dislocations occur in stories. It is helpful to have them in a separate window in order to clarify their relation to the main sequence of proof steps.
6. Tatami pages also have associated formal proof scores, which appear in another separate popup window when summoned from a tatami page. The separate window is convenient because users typically want to look at the formal proof and its motivation at the same time as the proof score. Users can also request proof score execution, and the result is displayed in the same window as the score, so that one can easily alternate between them. (The proof score is sent to an OBJ server and the result is returned for display.) This hiding of routine details is similar to human proofs, which use it to highlight the main ideas [Livingston, 1987].
7. Major proof parts can have an optional closing page, to sum up important results and lessons, again following Labov's theory. They appear in the same window as proof pages, again because they are part of the same narrative flow.
8. A menu of open subgoals appears on each homepage, and error messages are placed on appropriate pages. Open subgoals are important to provers when they read a proof, since proving new results is a major value within this community.

Now we give further style guidelines, with justifications based on algebraic semiotics:

9. **Windows:** The main contents of a display proofweb are its proof steps, informal explanations, tutorials, and mechanical proof scores. These four are also the main contents of abstract proofwebs, and their preservation has much to do with the quality of their representation. These four basic sorts of the abstract data type for proofwebs are reflected in our choice of windows for displaying them. Because tatami pages are the main constituent of proofwebs, theirs is the master window, and because explanation pages are so closely linked, they share that window; each unit is enclosed is its own "box." Tutorial and machine proof score pages each have a separate window. All this preserves the hierarchical structure and priorities of the underlying mathematics.
10. **Backgrounds:** Each major sort of unit has its own background color: proof units have light beige, explanations have light yellow, tutorials have yellow marble, and proof scores have light purple. Although the choice of colors is somewhat arbitrary, and is easily changed by editing the XSL style file, their distinctness reflects the importance of distinguishing these four units.
11. **Navigation:** Similar considerations hold for navigation. Each page has a title, supplied by the user in the Duck script (or a simple default if no title is supplied). Buttons are used to move to other pages of the same sort, and to open widows that display information of other sorts. Each persistent window has somewhat different layout and navigation buttons, reflecting its different typical uses. For example, the master tatami window has buttons to step through the narrative ordering of tatami pages, both forward and backward, and a button to return to the homepage.

12. **Mathematical Formulae:** `gif` files are used for mathematical symbols, in a distinctive blue color, because mathematical signs come from a domain that is quite distinct from that of natural language.

Some additional applications of semiotic morphisms to the user interface design of the Tatami system are described in [Goguen, 1999b], in a more precise style than here, although they are based on an older version of the system. For example, [Goguen, 1999b] shows that certain early designs for the status window were incorrect because the corresponding semiotic morphisms failed to preserve certain key constructors.

4.3 Humor

We have studied a corpus of over 50 "humorous oxymorons" (phrases like "military intelligence," "good grief," and "almost exactly"). Dictionaries say an "oxymoron" is a phrase having contradictory (or incongruous) components. But this is not what happens in a humorous oxymoron: instead, there are two distinct meanings, one of which is conventional, and the other of which has some contradictory components; i.e., there are two different blends, one of which has conflicts. When we are told that something is an oxymoron, we seek out that second, conflictual blend, and we feel pleasure when we find it.

We also studied over 40 newspaper cartoons, and found that about 75 percent have a similar pattern, but instead of two blends existing simultaneously, the reader is first led to form one blend, and then led by new information to form a completely different blend, usually in partial conflict with the first; that is, there is a kind of dynamic reblending.

Thus in each case, it is not just the existence of more than one blend, but rather the process of *reblending* that produces the humorous effect, and I conjecture that reblending in fact *characterizes* humor. This is relevant to HCI and the design of virtual worlds, because humor is sometimes used in computer system interfaces, often very badly. For example, the paperclip in MicroSoft Office creates a poor impression in part because the sensation of reblending loses loses its effect if it is repeated many times, and eventually becomes "stale" or even unpleasant. See [Goguen, 2004a] for additional details. These observations, which go back to about 1999, seem to have potential for fascinating new application areas.

4.4 Interaction

Classical semiotics is concerned with static signs; it does not allow for signs that change in response to user input, or that move on their own. This section sketches how algebraic semiotics handles dynamics, by extending its foundation from classical algebra to hidden algebra. As a simple example, consider the problem of designing that part of the Kumo interface that supports browsing proofs. Kumo provides buttons to traverse in the proof author's chosen narrative order, labeled with iconic triangles to indicate forward and backward motion, as well as buttons to return to the homepage, to view the specification, etc. (see Figure 8). Common practice would suggest constructing an automaton with a state for each proof tree node, and a transition label for each traversal button. But this does not allow for the fact that different proofs have different structures, and thus different automata, nor does it account for the different displays that are produced in each state, nor for the variety of possible implementations of transition lookup, e.g., using lists, arrays, or hash tables. An automaton can describe how a single proof instance can be navigated, but it cannot describe the general method which generates proof navigation support for any given proof, nor the way that this method is implemented, nor the quality of the resulting interface.

In fact, despite the formal character of the model itself, the construction and use of transition diagrams (or the corresponding automata) in user interface design is intuitive, and does not provide an adequate basis for a rigorous mathematical analysis of possible designs. In order to address the display, implementation and quality questions raised above, the automaton model must be supplemented in various *ad hoc* ways, whereas hidden algebra can handle all of these within a single unified framework. Another example of dynamics in Kumo that would be difficult to handle with traditional user interface modeling techniques is the facility to execute the proof script for a proof part by downloading it to a BOBJ proof server and then viewing the result on a local browser as it executes.

This is not the place for details (see [Goguen, 2003] for that), but we can say that hidden algebra provides a precise way to handle both the display and implementation aspects of examples like that described above, and the corresponding extension of semiotic morphisms gives a precise basis for comparing the quality of interface designs realizing the desired dynamics, without bias towards any particular implementation. The dynamics of a window with a scrollbar is discussed in considerable detail in [Goguen, 2003].

5 Summary, Future Research, and Social Implications

This paper has presented theory and case studies to support the claim that algebraic semiotics is a promising foundation for virtual world design, in both theory and practice. The case studies on information visualization, proof presentation, metaphor, humor, and interaction are encouraging, and suggest that design problems can be successfully confronted directly, without unreliable *ad hoc* methods and assumptions, such as analyses based on prior systems that are only remotely related, or expensive, time-consuming methods of experimental psychology and usability testing. These studies also confirm our views that taking account of key social and cognitive factors is crucial for success, and that formal methods can play a very helpful role, if applied pragmatically rather than dogmatically. However, much more work is still needed, such as:

- Combining Gibsonian affordances [Gibson, 1977] with algebraic semiotics, to provide a socio-cognitive dimension for the interaction formalism discussed in Section 4.4.
- Studying immersion in virtual worlds, e.g., how closure and embodiment relate to representational coherence, image schemas, affordances, choice of media, etc.
- More work on social foundations and the processes of semiosis.
- More work on narrative structure, including flashbacks and flashforwards.
- More work on how to choose quality orderings on representations that are appropriate to their actual use.
- More case studies, done more thoroughly.

Only the second of these is specific to virtuality, though all are related. We hope that others may find some benefit to the algebraic semiotic approach, and will contribute to its further development.

I close this article with some words of warning, along lines perhaps most closely associated with Jean Baudrillard [Baudrillard, 1994], who wrote:

> Simulation is no longer that of a territory, a referential being, or a substance. It is the generation by models of a real without origin or reality. ... By crossing into a space... no longer that of the real, nor that of truth, the era of simulation is inaugurated by a liquidation of all referentials – worse: with their artificial resurrection in the systems of signs, a material more malleable than meaning, in that it lends itself to all systems of equivalences, to all binary oppositions, to all combinatory algebra. It is no longer a question of imitation, nor duplication, not even parody. It is a question of substituting the signs of the real for the real, that is to say of an operation of deterring every real process via its operational double, a programmatic, metastable, perfectly descriptive machine that offers all the signs of the real and short-circuits all its vicissitudes. Never again will the real have the chance to produce itself – such is the vital function of the model in a system of death ...

If we translate this out of the stylistic conventions of recent French intellectualism, the danger is that the virtual can replace the real in our affections, so that we lose touch with our communities, our values, even the very living quality of our lives. Baudrillard claims that exactly such alienation is already characteristic of the contemporary world, and that it is growing like a cancer. He does not offer any solution to this dilemma, but I would like to suggest that compassion [Goguen, 2004b] is one way out of an enervating absorption in virtuality. A sympathetic feeling for the suffering of others, and action on their behalf, can generate positive emotionality and re-engagement with real experience. And, contrary to Baudrillard, it seems quite possible that technology, including virtual world technology, can assist with such projects.

References

[Aristotle, 1997] Aristotle (1997). *Poetics*. Dover. Translation by S.H. Butcher; original from approximately 330 B.C.

[Barthes, 1968] Barthes, R. (1968). *Elements of Semiology*. Hill and Wang. Trans. Annette Lavers and Colin Smith.

[Barwise and Perry, 1983] Barwise, J. and Perry, J. (1983). *Situations and Attitudes*. MIT (Bradford).

[Baudrillard, 1994] Baudrillard, J. (1994). *Simulacra and Simulation*. Michigan. Translated by Sheila Faria Glaser.

[Campbell, 1973] Campbell, J. (1973). *The Hero with a Thousand Faces*. Princeton. Bollingen series.

[Carroll, 1982] Carroll, J. (1982). Learning, using, and designing filenames and command paradigms. *Behavior and Information Technology*, 1(4):327–246.

[Fauconnier, 1985] Fauconnier, G. (1985). *Mental Spaces: Aspects of Meaning Construction in Natural Language*. Bradford: MIT.

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

[Fauconnier and Turner, 2002] Fauconnier, G. and Turner, M. (2002). *The Way We Think*. Basic.

[Gentner, 1983] Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170.

[Gibson, 1977] Gibson, J. (1977). The theory of affordances. In Shaw, R. and Bransford, J., editors, *Perceiving, Acting and Knowing: Toward an Ecological Psychology*. Erlbaum.

[Goguen, 1994] Goguen, J. (1994). Requirements engineering as the reconciliation of social and technical issues. In Jirotka, M. and Goguen, J., editors, *Requirements Engineering: Social and Technical Issues*, pages 165–200. Academic.

[Goguen, 1997] Goguen, J. (1997). Towards a social, ethical theory of information. In Bowker, G., Star, L., Turner, W., and Gasser, L., editors, *Social Science, Technical Systems and Cooperative Work: Beyond the Great Divide*, pages 27–56. Erlbaum.

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

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

[Goguen, 2003] Goguen, J. (2003). Semiotic morphisms, representations, and blending for interface design. In *Proceedings, AMAST Workshop on Algebraic Methods in Language Processing*, pages 1–15. AMAST Press. Conference held in Verona, Italy, 25–27 August, 2003.

[Goguen, 2004a] Goguen, J. (2004a). CSE 275 homepage: Social issues in science and technology. `www.cs.ucsd.edu/users/goguen/courses/275/`.

[Goguen, 2004b] Goguen, J. (2004b). Groundlessness, compassion and ethics in management and design. In Boland, R. and Callopy, F., editors, *Managing as Designing*. Stanford.

[Goguen and Harrell, 2003] Goguen, J. and Harrell, F. (2003). Information visualization and semiotic morphisms. In Malcolm, G., editor, *Visual Representations and Interpretations*. Elsevier. Proceedings of a workshop held in Liverpool, UK.

[Goguen and Harrell, 2004] Goguen, J. and Harrell, F. (2004). Foundations for active multimedia narrative: Semiotic spaces and structural blending. To appear in **Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems**.

[Goguen et al., 2000] Goguen, J., Lin, K., Roşu, G., Mori, A., and Warinschi, B. (2000). An overview of the Tatami project. In Futatsugi, K., Nakagawa, A., and Tamai, T., editors, *Cafe: An Industrial-Strength Algebraic Formal Method*, pages 61–78. Elsevier.

[Goguen and Linde, 1984] Goguen, J. and Linde, C. (1984). Optimal structures for multi-media instruction. Technical report, SRI International. To Office of Naval Research, Psychological Sciences Division.

[Goguen and Malcolm, 1996] Goguen, J. and Malcolm, G. (1996). *Algebraic Semantics of Imperative Programs*. MIT.

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

[Goguen et al., 1978] Goguen, J., Thatcher, J., and Wagner, E. (1978). An initial algebra approach to the specification, correctness and implementation of abstract data types. In Yeh, R., editor, *Current Trends in Programming Methodology, IV*, pages 80–149. Prentice-Hall.

[Grady et al., 1999] Grady, J., Oakley, T., and Coulson, S. (1999). Blending and metaphor. In Gibbs, R. and Steen, G., editors, *Metaphor in Cognitive Linguistics*. Benjamins.

[Labov, 1972] Labov, W. (1972). The transformation of experience in narrative syntax. In *Language in the Inner City*, pages 354–396. University of Pennsylvania.

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

[Lakoff and Johnson, 1980] Lakoff, G. and Johnson, M. (1980). *Metaphors We Live By*. Chicago.

[Linde, 1981] Linde, C. (1981). The organization of discourse. In Shopen, T. and Williams, J. M., editors, *Style and Variables in English*, pages 84–114. Winthrop.

[Linde, 1993] Linde, C. (1993). *Life Stories: the Creation of Coherence*. Oxford.

[Livingston, 1987] Livingston, E. (1987). *The Ethnomethodology of Mathematics*. Routledge & Kegan Paul.

[Mac Lane, 1998] Mac Lane, S. (1998). *Categories for the Working Mathematician*. Springer. Second Edition.

[Miller, 1956] Miller, G. A. (1956). The magic number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Science*, 63:81–97.

[Peirce, 1965] Peirce, C. S. (1965). *Collected Papers*. Harvard. In 6 volumes; see especially Volume 2: Elements of Logic.

[Pierce, 1990] Pierce, B. C. (1990). A taste of category theory for computer scientists. Technical Report CMU-CS-90-113, Carnegie-Mellon University.

[Saussure, 1976] Saussure, F. (1976). *Course in General Linguistics*. Duckworth. Translated by Roy Harris.

[Shneiderman, 1997] Shneiderman, B. (1997). *Designing the User Interface*. Addison Wesley. Second edition.

[Shneiderman, 1998] Shneiderman, B. (1998). *Designing the User Interface*. Addison Wesley. Third edition.

[Tufte, 1983] Tufte, E. (1983). *The Visual Display of Quantitative Information*. Graphics Press.

[Turner, 1997] Turner, M. (1997). *The Literary Mind*. Oxford.

[Zbikowski, 2002] Zbikowski, L. (2002). *Conceptualizing Music*. Oxford.

Contents