Extreme Markup Languages 2002

Montréal, Québec
August 6-9, 2002

Human and Machine Sign Systems

Wendell Piez
Mulberry Technologies, Inc.

Abstract

A schema's role is to mediate and adjudicate between human and machine semantics; recognizing this can help us manage our schemas better. Some practitioners work solely with an operational semantics, according to which the meaning of a tag is what we want it to cause the processing software to do with the data. A better understanding is reached if we adopt the structuralist view that a sign is the (arbitrary) relation between a signifier and a signified. In metalanguages (including schema languages) the signified is itself a sign; in some languages the signifier may likewise be a sign. Proper understanding of the relationship among sign, signifier, signified, metalanguage, and connotative system will allow us to layer our systems more effectively and to obtain useful results even in fluid systems where our understanding of the underlying reality cannot, or should not, be fixed.

Human and Machine Sign Systems

Table of Contents

Human and Machine Sign Systems

Wendell Piez

§ 1 From schemas to the “semantics” problem

This consideration began in an impression that there is a deep connection between two ongoing discussions in the markup languages community. The first is the debate on schema languages. Pursued both in public (in the public lists and publications) and in private (in classrooms and design sessions), this is a collective consideration (a collective thought process, as it were) that ranges from highly theoretical and abstract arguments about the role of schemas in systems, to the most intricate details of the various features of schema technologies, real and hypothetical. The second discussion considers a problem that may seem to be both more obscure and more abstruse: namely, what are “semantics” anyway? This paper will argue that these two issues are intimately linked, and moreover that the second problem — despite our habit of treating it in a much more cavalier, less rigorous way[1] — is the more fundamental. That is, if we can gain better insight into what “semantics” are and how markup languages express them, various of our design problems, including what we should consider to be the proper applications and functional requirements for schemas (or, more contentiously, for a standard schema language), may become more tractable. Because it is focused on semantics as a fundamental theoretical issue, I have called this paper *Human and Machine Sign Systems*. To give a hint of where I am going, a subtitle (after Freud) might be *Formalism and its Discontents*.

Interestingly, and significantly for my argument, when the issue of the semantics of markup languages or markup technology applications has come up in any formal or systematic treatment, it has almost always been in order to postpone it. The most salient example is that of the “Semantic Web”. Here is a case where an implicit promise is made: the term “semantics” suggests we will benefit from some kind of sensitivity to, even intelligence regarding, what human users intend to “mean” by terms or words we deploy. Yet when you get into the closer discussions about the way the Semantic Web will be built, it always turns out the specific promises are, cannily, more modest.[2] The tactic, almost inevitably it seems, is to make a distinction between what I am calling “machine” and “human” semantics — that is, first, the kind of semantics or meaning that can be implemented or expressed, at least in principle, in some kind of automated process, as opposed (secondly) to some kind of broader sense of what human beings mean when we use language or any system of signification. Merely by setting up this distinction, the evident problem of whether a machine can “know” what we “mean” is set aside.

However we may suspect this is avoiding the issue, this distinction between what human beings mean, and what machines mean (or as I will suggest, how machines can be used to channel, mediate and express meanings), is nevertheless a useful one, and one I will build on. In part, it serves to isolate in a very helpful way the difference between what our machines actually do, and fuzzier notions regarding what they might be supposed to do by someone uninformed about what’s actually going on in the mysterious inner workings of an automated data processing system. Even if it would seem to be absurd to claim that a machine can know (whatever we mean by that) what a human user means, nonetheless machines are evidently doing *something* — and given that information-processing systems are built specifically to deploy, manage and manipulate symbols (things that mean other things), that something apparently has something to do with meaning.

To be more explicit, an elementary example might be the use of an element called `<emph>`. Now this `<emph>` element might have the simple consequence, when a document is processed (let’s say it is being converted into some other format for print or screen display), of providing that its content be displayed in an italic type face. (Maybe its content is rendered in italics unless the surrounding context is italic, in which case it is “toggled” the other way, to roman.) Now this mere behavior may not be supposed, in the intention of the designers of the system or the user of the tag, to be the entire

meaning of <emph>. For example, in a TEI system, italics might also be a consequence of other elements, such as <foreign> or <hi>, leaving <emph> for a certain (analytical) subset of elements expressed in italics: "emphasized" text.[3] (Actually, as we will see this problem arises to begin with since <emph> is not supposed to be a sign for italics, but rather the reverse; or more particularly, italics are sometimes taken to be a sign for something of which the element <emph> is also a sign.) Nonetheless, we can say that in general, presentation in italics comprises at least part of the semantics of <emph>; indeed, if our system happens to do nothing but display our data (like some document-oriented systems), there might be nothing more to it than this: <emph> *means* "italics".

This is what I will call the "operational view" of the tag's semantics. We can also call this the "machine semantics" of the tag (or labelled element type), while noting that this is the kind of semantics to which arguments for "semantic processing" must necessarily limit themselves. So much is tautological, or a matter of definition: machine semantics is that kind of semantics (that set of "meanings") that is effected operationally through some functionality in an automated system. "Human semantics", in contrast, are not so limited. This distinction will be useful even if we posit that the border between the two is constantly shifting, as machines get more and more sophisticated (and perhaps as human beings get more or less so). I hope it is also evident that there is great utility and power in the operational view.

But we cannot reduce the problem of semantics in general by saying that operational semantics are the only semantics that exist, or the only ones that matter. This is to explain the problem away by begging the question. In particular, to say that operational semantics are the only semantics leaves us helpless to explain why, or even to recognize when, our machines go wrong, as clearly they often do. A rather refined case of this is how, in markup systems, it leaves us with no way to assess when a particular usage of an element or attribute type is "tag abuse".[4] More broadly, the operational view gives us no basis on which to understand when a system is well-fitted to the problem it seeks to solve or the power it seeks to provide. It does what it does. "The proof of the pudding is in the eating", but in the absence of an understanding of what tastes good or not, we can't prove anything. The web shows us this every day: it is a strange world in which successful searching requires a skill in imagining how pages present themselves *despite* their actual content, where the earnest and literal is all mixed up with the partial, the satirical, the fraudulent. Inevitably, we have to turn to the wider context — the wider world — to see whether something works or not, and to understand why it works the way it does.

§ 2 Structuralism, signs and layered sign systems

In trying to fathom these issues, I decided that a systematic approach was called for. I looked at information theory, only to discover (to my surprise and interest), it was deliberate in not providing that approach. On the contrary, classic information theory,[5] in taking the position that information (which in its view is understood in the context of the problem of transmission) reduces itself to the problem of how to identify a particular message among a given set of possible messages.[6] This is a consistent position, and perfectly defensible in its context; but it leaves us effectively in the same place as the notion that only operational semantics are worth considering. Operational semantics themselves are those semantics that are expressible through the differentiation of messages in a system that already knows what to do with them (what they "mean"). The position does not address the hovering problem of how to create or understand new meanings — how to consider what messages we have in the light of "higher levels" of understanding, or in view of wider contexts. It was precisely what information theory itself sets aside as not significant for its problem, that I found I was trying to grapple with.

Fortunately, there is another discipline that attempts to deal with this problem systematically — and in fact, provided a critical intellectual foundation to information theory itself (and to the theory of formal grammars which relates to it). Semiotics, or the study of signs, and more generally the roots of late-twentieth-century Semiotics in Structuralist Linguistics, presents a number of observations

that can provide us with clues to solving this puzzle. While my argument is not especially informed by a good century of work in literary criticism, linguistics, psychology, anthropology and related fields that have reflected, both at length and in depth, on sign systems — sometimes taking Structuralism to extremes where even this paper might hesitate to follow — nonetheless even a cursory examination of the fundamental concepts of the theory demonstrates its relevance for the problem at hand.[7]

Structuralist Linguistics was founded about a century ago, by Ferdinand de Saussure, a Swiss linguist who studied in Germany and Paris and taught at the University of Geneva; his *Cours de Linguistique Générale* was first published by his students in 1915. More recently, his ideas were picked up, summarized and developed by (among others) the French post-structuralist theorist Roland Barthes, whose work (along with Structuralism and Post-structuralism altogether) has been very influential over the last thirty years in academic humanities in the West.[8]

For our purposes, there are three points essential to Structuralism, which reveal it to be highly significant to the design of markup languages and the development of their applications:

- Signs are constituted not as essential objects of any kind, but rather as phenomena in relation. The primary relation that constitutes a sign is that between a **signifier** (we might sometimes say a "token", the external expression of the sign) and what it signifies. This relation is *arbitrary*, not determined by nature but rather only within the context of the system as a whole.
- Signifiers in relation to one another together constitute more complex signifiers. A sequence of words builds into a sentence; a combination of elements, `<day>`, `<month>`, `<year>`, can be assembled into a `<date>` element. The systematic aspect of these relations between tokens, even apart from any putative relation between signifiers and what they signify, links Structuralism with information theory, which begins by considering exclusively the relations within the layer of signifiers. (Note that the arbitrariness of any given signifier is essential to this capability; if significations were "wired" into individual tokens, they could not be substituted and recombined freely but could only be deployed in predetermined ways.)
- The capability of recombination between elements, tokens or signifiers, enables them to go beyond the simple conventional relation of signifier to signified, to constitute higher-order significations. It is not only that "tiger" signifies a large striped feline that lives in the rain forest; a "man-eating tiger" is something more from the coincidence of the word "tiger" with the words "man" and "eating". When Emily Dickinson writes a "tiger eased / By but a crumb of blood, fasts scarlet / Till he meet a man / Dainty, adorned with veins and tissues", that means something more again. Thus the entire system of relations across the layer of signifiers can be reflected in an analogous system in the layer of what they signify. In other words, they constitute a *layered system* — which is a salient feature of markup technologies as well.

The following sections spell out these principles in more detail, before turning to a consideration of their implications.

2.1 *The sign according to structuralism*

It is possibly easier to understand structuralism and its implications by considering diagrams or illustrations of its core concepts. Figure 1 copies Saussure's diagrams indicating the dual nature of the sign from his *Course in General Linguistics* [Saussure 1915]. Figure 2 provides an example to show the arbitrary nature of the sign. This essential feature of language is the source of much of its power; though not all world-views allow for it, preferring to pretend that meanings should be fixed and "literal" (and though even those of us who do not consciously ascribe to such a world-view, still keep falling into treating signs as if their meanings were fixed), it is something that poets, clowns

and children teach us to recognize and even revel in. Figure 3 is a depiction of how signs *in combination* imply systematic relations between the structures of language (or any sign system) as it occurs phenomenologically, and the implied structures of conception which it expresses — and which arguably motivate it to begin with.

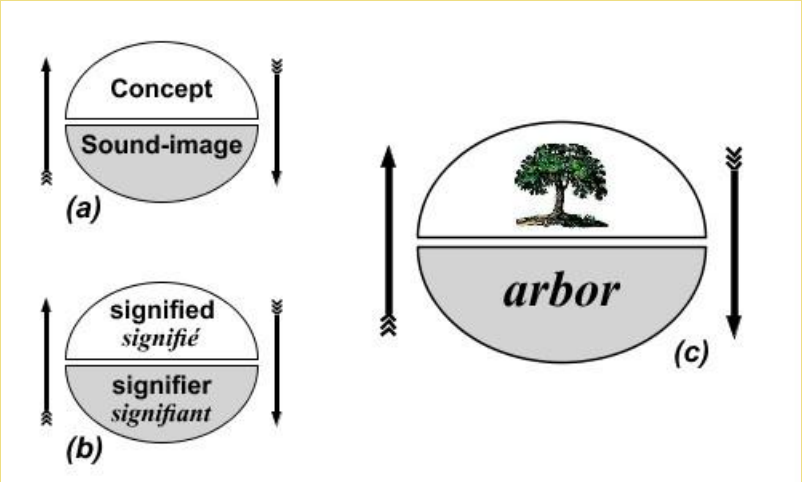


Figure 1 Saussure's conception of the sign

In this figure, (a) shows Saussure's first effort to depict the constitution of a sign in its two parts. A sound-image (and Saussure stresses it is not a sound or an image *per se* that he means to indicate, but rather a sound or an image *as distinguished as a sign* apart from other sounds or images) serves to represent an abstract mental conception (which in turn may stand for some "thing" in the world). Diagram (b) shows the same relation, except using the technical terminology Saussure introduces. The word "sign" is not useful to indicate the thing that does the representing, as it is the sign itself that is made of two parts: nothing is a signifier unless it is bound to something else signified, and it is this binding that constitutes the sign. Diagram (c) shows an example: the Latin word *arbor* is a signifier in the Latin language for the concept of a tree.

Alert readers will observe that the diagram, of course, shows only a picture of a tree. But you get the idea.

Figure 2 What it means to be "open"

A sign is not determined merely by a signifier; it is constituted by a signifier *in a relation to what it signifies*. While this relation is commonly habitual and unconscious, the arbitrary relations between signifiers and what they represent makes for an ever-present possibility of slippage: ambiguity, jokes, mistakes and multiple meanings. Disambiguation is ordinarily performed with the help of a sign's context.

For example, if you took the sign depicted here on the right to mean "open for business", you might find yourself in trouble with the store management.

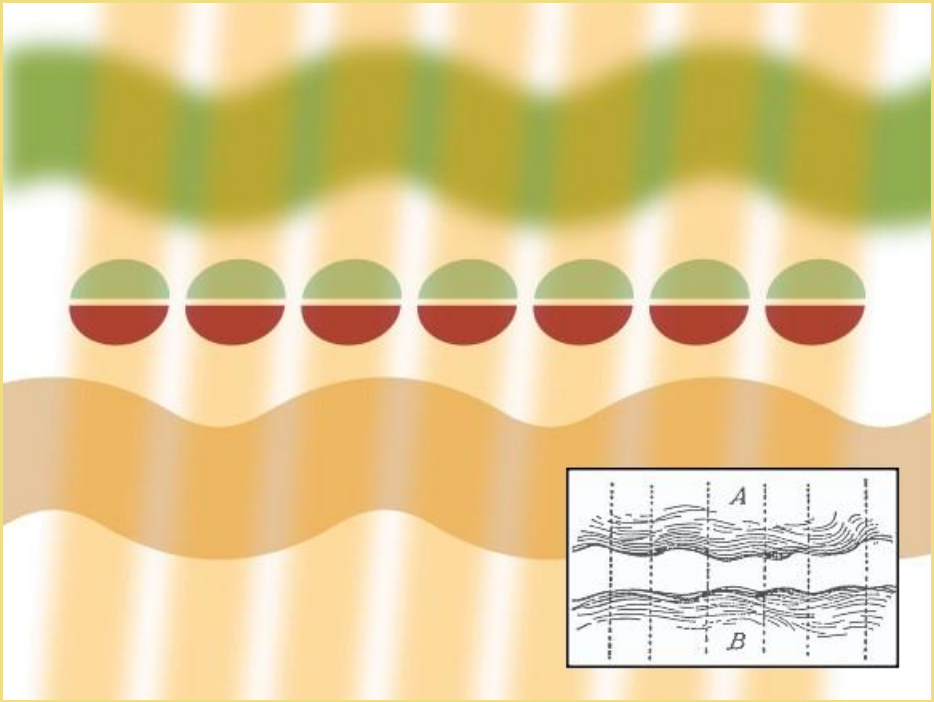


Figure 3 Signs in context imply correspondences between realms

Saussure pointed out how we make sense of signs by locating them in context, either of an ordered progression (as in words in a sentence) or in a visual relation, or in a social situation. Thus arbitrary tokens (signifiers) support higher-level signification, not *despite*, but *in combination with* their arbitrariness, since an ordered arrangement of signs in combination allows a correspondence (albeit still slippery) between entire "planes" of signifiers and signified. Thus an external language, as a sequence (and system) of utterances or expressions, reflects a conceptual model of the world. Moreover, each plane exists only by virtue of, and in relation to, the other.

The inset graphic is a copy of the depiction of this concept as it appears in Saussure's *Course*. "The linguistic fact can therefore be pictured in its totality — i.e. language — as a series of contiguous subdivisions marked off on both the indefinite plane of jumbled ideas (A) and the equally vague plane of sounds (B)." [Saussure 1915]

Following this basic conception of the nature of the sign, it is a relatively straightforward exercise to apply these notions to markup technologies. In any language (or for that matter for any API, macro or process) we can discern various levels of signification. Taking for a moment a single sign in one language or another, we can distinguish between its signifying part, and what it signifies — what we "know" in case we encounter it. In order to establish a careful exposition of these concepts, however, it is useful to alter the visual notation somewhat, as illustrated in Figure 4.

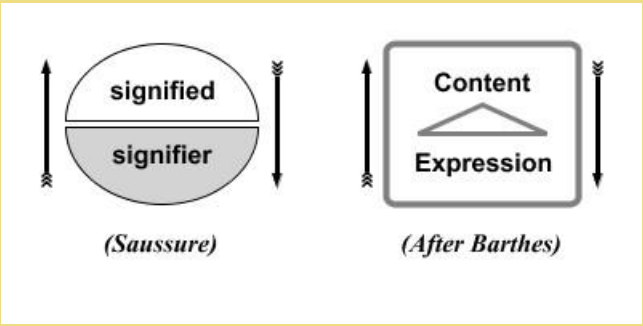


Figure 4 Alternative notations and terms

Saussure uses the terms “signifier” and “signified”. Roland Barthes (or rather his translators) suggest “expression” and “content”. We can speak of “signifier”, “expression” or “representation”, on the one hand, “signified”, “content” or even “concept” on the other. There is a reason not to fix onto a particular terminology.

Such a diagram as the one on the right can easily be read as “**x** is a sign for **y**” where **x** is the signifier (expression), **y** is the signified (content).

2.2 Metalanguages and connotative systems

Given these basic insights, however, semiology goes considerably further, by recognizing ways in which sign systems themselves are both objects of knowledge, and devices for further signification. Roland Barthes, following other students of the problem, describes two forms of higher-order system [Barthes 1964]. Given the distinction between its two planes, a sign system can *recurse* in either of two ways (depicted in Figure 5), when a sign (which combines signifier with signified, or expression with content) plays the role either of signifier or signified in a “staggered”(i.e. multi-leveled) system.

When a set of signifiers is used not to refer to elemental objects (whether they be concepts or anything else) but rather *signs*, Barthes sees fit to call this a “metalanguage”. As should become evident, the coincidence between this terminology and the use of the same term in our own field is not insignificant.

Perhaps more complicatedly, it is also possible to recurse in the opposite direction, taking a sign system not as an object of signification (a **signified**), but rather as a signifier for a projected or “connoted” signification of some sort. Barthes calls this a “connotative system”, proposing literature as an example of such a system. Not only does a reader of literature construe signs into meanings; also those immediate meanings themselves are constructed into higher-order meanings. This also happens in ordinary conversation, in its allusion and ironies. To take just a simple example, we might describe a colleague as a “Don Quixote”, taking the *sign* Don Quixote (a character in a novel) as a signifier in a metaphorical description of our friend as someone who is liable to “tilt at windmills”. We don’t mean to say that he is literally a Spanish knight, or that he is a figure in a novel, or that he literally attacks windmills; rather, the sign “Don Quixote” serves a signifier in relation to a signified that is understood to be a metaphorical description of this person’s chivalrous, idealistic temperament. (Indeed, given the nuances of Cervantes’ novel and of the notions of knights, tilting and windmills, we may find it difficult to describe this aspect of our friend’s character without such a compressed and economical expression as “Don Quixote” with all its connotations.) Likewise, entire media cultures work in such a way: television is rife with examples, in the form of generic conventions in situation comedies or cop shows, which work to build an odd species of suspense in their gags or plot twists by virtue of their manifold implicit references to the conventions of sitcoms or cop shows.

Suggestively, Roland Barthes argues that in connotative systems, “signifieds have a very close communication with culture, knowledge, history, and it is through them, so to speak, that the environmental world invades the system”. Following Hjelmslev, Barthes suggests the term “ideology”

for the signified (content) of such sign systems, “rhetoric” for the signifiers or expressions (see [Barthes 1964]).

As we apply these ideas to markup technologies, it’s important to keep in mind one aspect of these diagrams. It may be tempting to fall into old habits and read them as if they were Venn diagrams describing subset-superset relations. They aren’t. Each box in a diagram designates a sign or (more broadly) a generalization of the nature of a sign in a particular kind of sign system; what appears on either side of the dividing bar (depicted as an arrowhead pointing from signifier to signified[9]) within the box only serves to show how the two components relate to one another: which serves as signifier (external expression), and which as signified (what the sign refers to, its “content”).

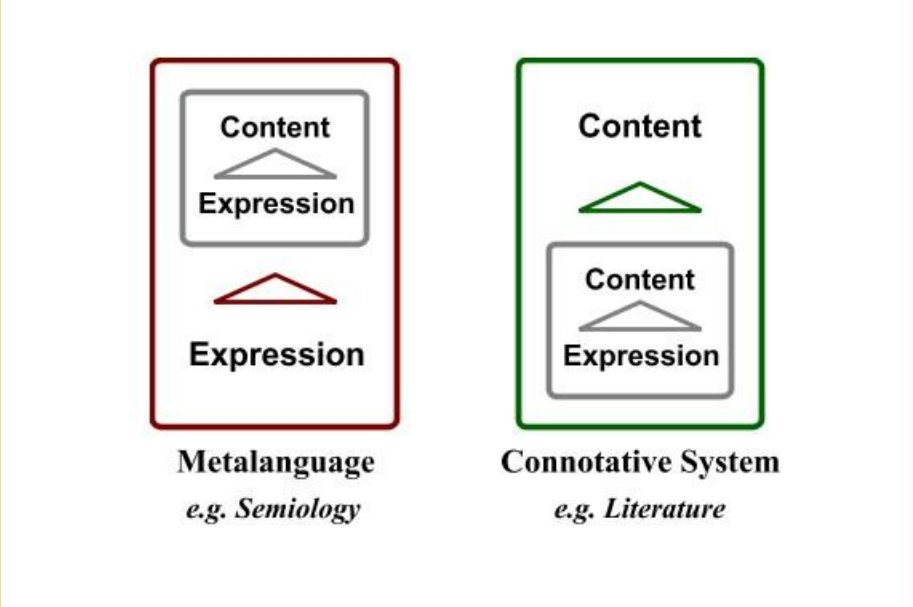


Figure 5 Compound sign systems

Barthes shows how sign systems can be constructed reflexively, with a sign system playing the role of either signifier or signified.

Semiology (the study of signs) is a metalanguage since it deploys terms to describe signs themselves, thereby posing them for analysis. Other examples include literary criticism and historiography, and any computer language from Assembler on up. The specification of virtual machines shows how even here there can be a slippage between layers: a virtual machine implements its signified as a sign system, while allowing the possibility of other implementations of the same signified (and so the possibility of exchange of its signifiers with such an alternative implementation).

A connotative system is possibly a more difficult notion. Literature constitutes a connotative system since, for example, its tropes (figures), characters and generic conventions all work on at least two levels at once, both conveying senses in their immediate context, and fitting into larger systems in which those signs themselves signify by their relation (allusion) to other signs. Popular culture is also rife with connotative systems, for example television genres such as cop shows and sitcoms, or music or fashion trends.

§ 3 Markup technologies as sign systems

A series of diagrams (see Figure 6 through Figure 14) can illustrate how markup technologies work as sign systems. Note that in these diagrams, an element `<title>` (or a rough HTML equivalent in `<h1>`) is taken as an example; but this is only to illustrate each sign system by way of an intelligible example. The reader needs to extrapolate from the example given, in the understanding that no sign works in isolation from other signs.

3.1 *Simple sign systems*

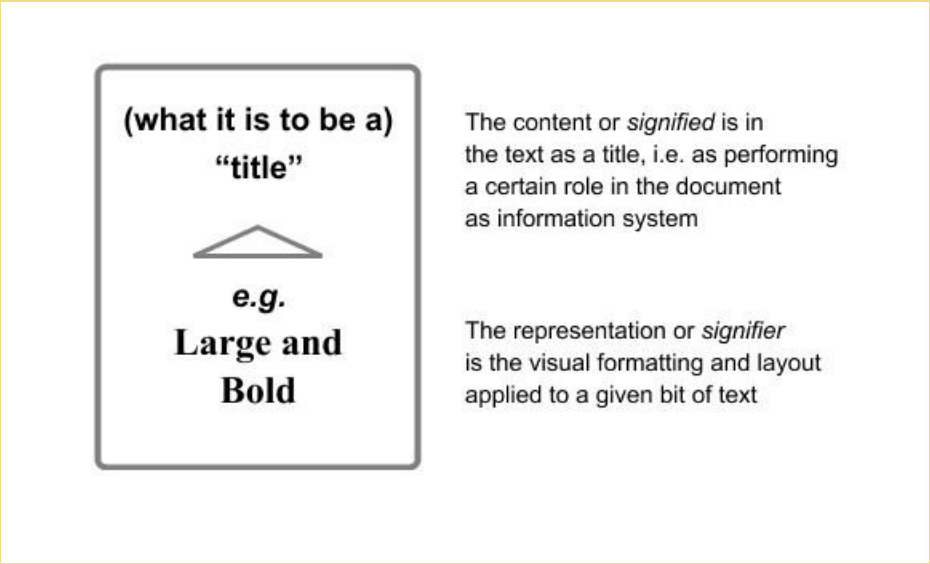
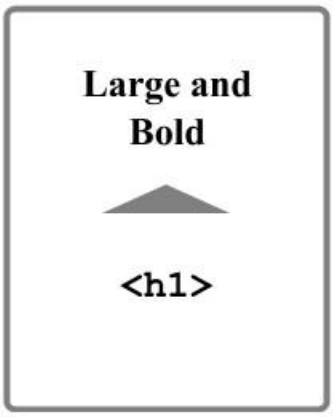


Figure 6 Signification of a title to a reader

Both the presentation of a given snippet of text, and its context (for example, its position) are usually necessary to indicate that it is a title.

As such, it functions within a larger system of relations which constitute the entire sign system of the document layout and handling.

One useful aspect of these diagrams is that they allow us, finally, to make a practical distinction between machine or operational semantics on the one hand, and human semantics on the other. HTML is a ready example, in part because our experience with it has alerted us to how really arbitrary, in practice, is the signification that its elements provide. In view of the actual practices of readers, we have to posit only provisionally that a large-and-bold presentation "means" that a given text is a title (Figure 6). While this may generally be the case, we can also distinguish how other contextual factors come into play, for example the placement of a title at the start of a document. The rule is only a very loose one. Yet because HTML's `<h1>` element markup has the operational semantics of making the included text large and bold, it may make it an appropriate way to mark up a title (by an implicit reliance on both signifying steps occurring in a kind of chain).[10]



The content or *signified* is in the processing applied to the element (if this is not reasonably consistent across implementations, the system breaks down)

The representation or *signifier* is the markup `<h1>...</h1>`

Figure 7 HTML: operational semantics (naïve view)

In a more sophisticated view (say, some XHTML applications) the tags only serve to denote an abstract element, which in turn receives processing. In this view, the tag `<h1>` is not simply a signifier; what it signifies (the element) is a signifier in a wider system.

Both these critiques are merely to restate what we have already identified: that signification is arbitrary, and that it commonly occurs in layers.

The content or *signified* is in the recognition that the text is a title — both for processing, and ultimately to an end-user

The representation or *signifier* is the markup `<title>...</title>`

Figure 8 XML in theory (generic markup)

According to the theory of generic (descriptive) markup, the tagging of data as a `<title>` should be a transparent and dependable signifier for its "titleness".

This is easy enough to *say*... (thus reminding us how powerful human sign systems are).

According to the theory of generic markup (Figure 8), in XML, element types may serve directly to signify logical constructs not merely for machine processing, but in an "objective" ontological sense. That is, because something in the text should only be marked up `<title>` if it actually "is" a "title" (in whatever senses of that term has been postulated by the markup designer), the tag or element type name `<title>` can serve as a dependable label for that thing in the world, a title. In such a system, operational semantics are relegated to the role of providing appropriate expressions and applications of that ontology.

If we don't buy into that theory, however, we can still fall back on operational semantics. (This is depicted in Figure 9; a careful reader will see that in practice, the same reservations apply to this view of XML as apply to the description of HTML's operational semantics.) In fact, regardless of the philosophical position we take, it is evident in practice that good designers will always be ready to reduce or limit their view, if only pragmatically, of what an element "is", to this (what the tag or element actually does for us), conceding that at some point theory may have to yield to practice.[11] Even if the operation of a title element (along with, and in accordance with, other elements and constructs) is not all we have, they are all we can *see*. Whether we can reliably step from this simpler signification to the more ambitious claim that what we have is "actually" a title, is perhaps something that is up to the recipient of the element as processed (again, Figure 6), or indeterminable, rather than something up to the markup designer or even the user who intends to "make a title" by deploying the markup for it.[12]

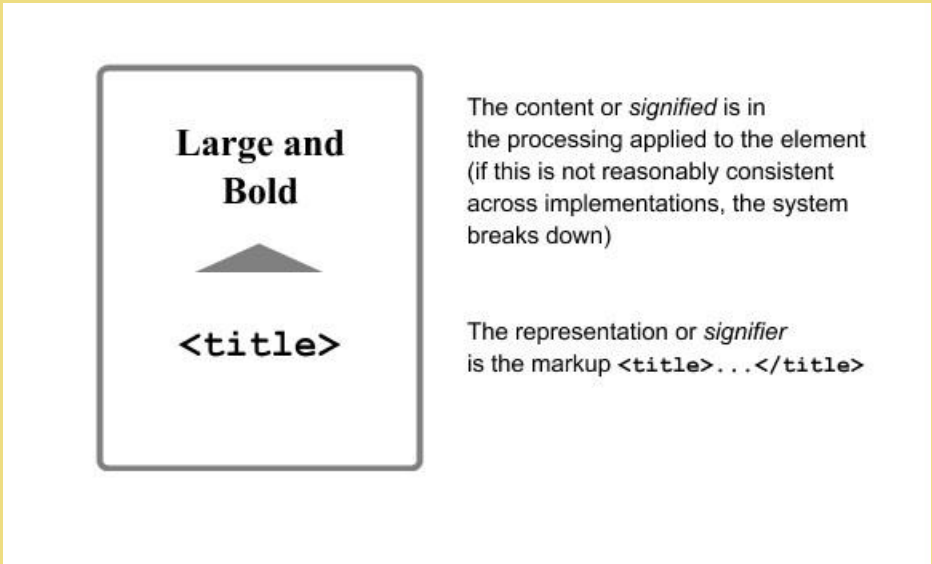


Figure 9 XML: operational semantics (naïve view)

To an author who uses an XML system, a tag `<title>` "means" to display (or otherwise process) the data content in a way appropriate to titles. (A very naïve author may believe that that way is fixed, unaware that it is up to an implementation or stylesheet to effect that processing.)

Note also here how examples are given both of markup as putative signs, according to the theory of generic (descriptive) markup, and according to a competing world view, which reduces the signification of tags or element types to their operational semantics. HTML elements have virtually no semantics at all apart from their operation (notwithstanding the apparent origin of HTML in the Gencode Standard [Gencode 1983] and its occasional use of ostensibly generic elements such as `<em>` or `<blockquote>`); in XML, however, the principle of "separation of content from presentation" argues that good design practice is for elements to signify directly the abstractions

(Figure 8) that, in turn, their presentation will presumably serve to indicate to the reader (Figure 6). Despite this theoretical point, however, in a working system an XML element has an operational semantics, to which (in the other view at least) its wider semantics will have a tendency to reduce.

3.2 *Complex sign systems*

As already indicated, however, these simple signifier/signified relations, when examined more closely, apparently require more discrimination. This is where the observation that signs can participate in higher-order sign systems — metalanguages and connotative systems — becomes really useful.

Due to limitations of space, it is necessary here to compress the discussion of metalanguages to a minimal presentation of the features of such metalanguages as we are familiar with: schemas and stylesheets (processing specifications). The reader is encouraged to ponder the implications of the concept, however, for markup systems and information systems in general, inasmuch as the capability we have to describe our sign systems recursively, through other signs, whether these descriptions be for purposes of constraint or of processing (schemas or stylesheets), is central to the automation of information processing.[13]

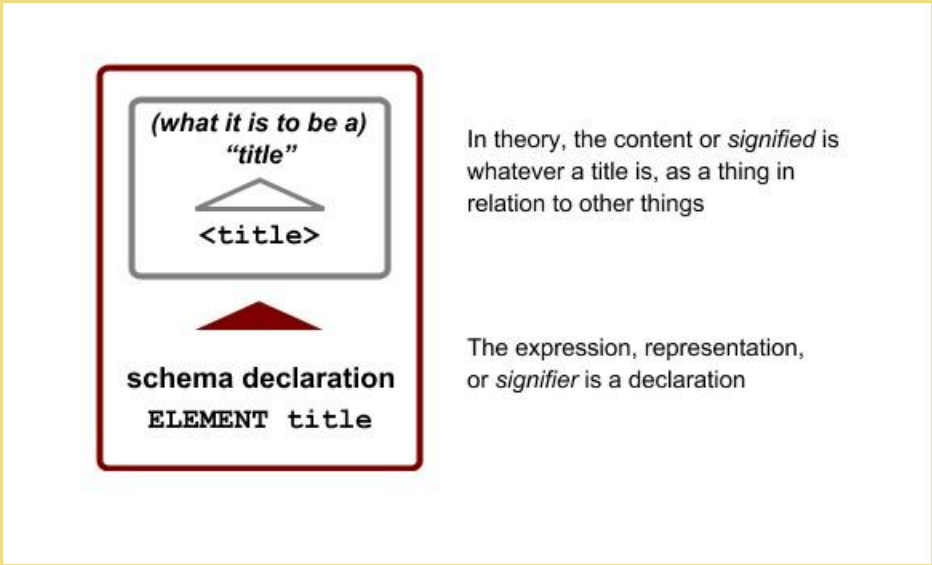


Figure 10 Schema as metalanguage (theory of generic markup)

In a system implementing generic (descriptive) markup, the role of a schema is to describe and constrain what it is to be a title as distinct from other data elements. The classic theory (cf. SGML, ISO 8879) recognizes that this cannot be formalized in principle, so argues that the formal declarations only constitute *part* of a document type definition.

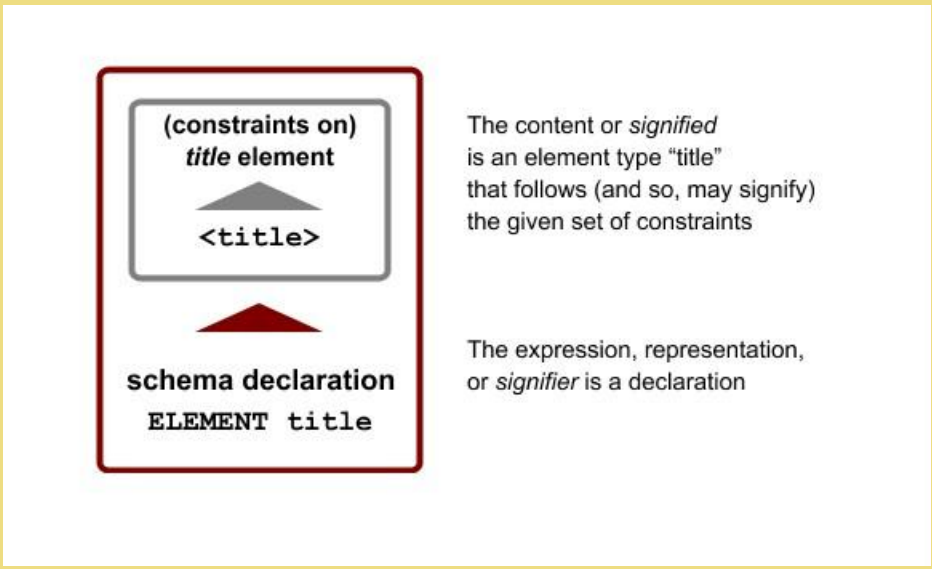


Figure 11 Schema as metalanguage (operational semantics)

Since the context (especially its position in an order relative to other content) of an element's appearance is a critical signifier in document (layout) systems, document-oriented markup applications commonly constrain this positioning. This kind of constraint is sometimes superfluous in data-oriented applications; yet even there, constraints on an element (for example its datatype) may be necessary to assure the integrity of processing downstream.

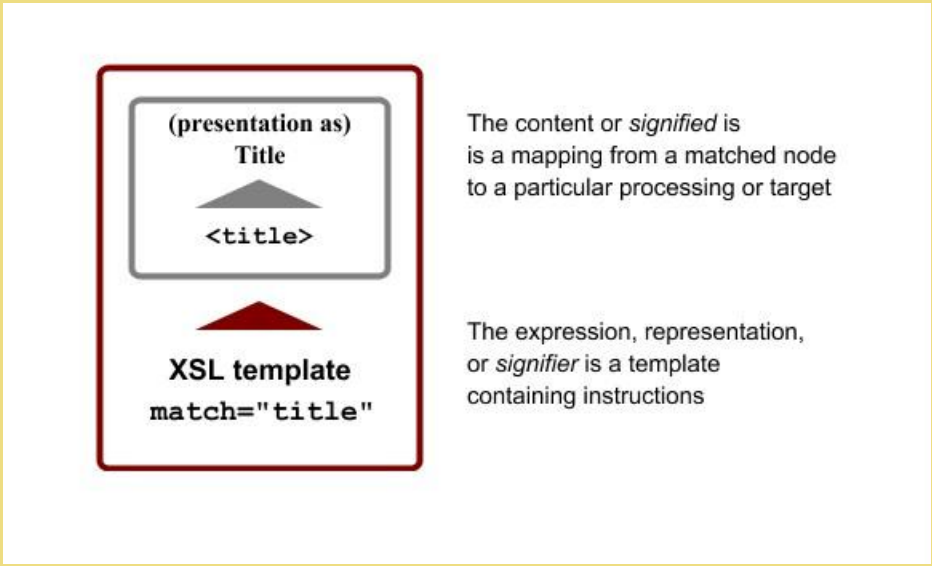


Figure 12 Stylesheet as metalanguage

XSLT, as a declarative language, serves as a sign system of a higher order than would a procedural program that did the same task (transformation or reformatting). In Perl or Omnimark, the signifiers would represent abstractions in the programming environment: variables, data objects, operations and so forth. In XSLT, a template (in its simplest form) signifies a mapping from a node type in the source to a target node type. Since its signified is also a sign (i.e., a signifying relation in which the source node may be taken to signify the target or vice versa), the stylesheet is also a metalanguage.

But it is when we consider connotative systems that things really get interesting. The concept of the connotative system, in which a sign (signifier/signified) can in turn be taken as a signifier of a higher-order signified, is critically important, and useful, since it is how we are able to bridge all the gaps we introduce through our deployment of signs. Where a metalanguage is an instrument of description and control, a connotative system is where we can see that control must finally be ceded to the user — ultimately the "end-user" — of any system. The ever-present potential for a connotative system, for all our sign systems to be taken as merely signifiers of something else, is both what allows our systems to work in the first place (what gives them meaning), and what prevents us from "putting a cap on them", constraining signification to any one thing.

In other words, even in the face of a reduction of our system to operational semantics, a connotative system is how we get back to titles (or other things that matter to us). We do not have even to say what titles are, to see that connotative systems can do this for us: without being able to define a title, a user can say "I know it when I see it", and that is sufficient. We merely need to present the user (the reader or consumer of information) with a useful *signifier*, that is, for signification to occur. In fact, it is up to them, not up to us, what they will take that signifier to mean in any case.

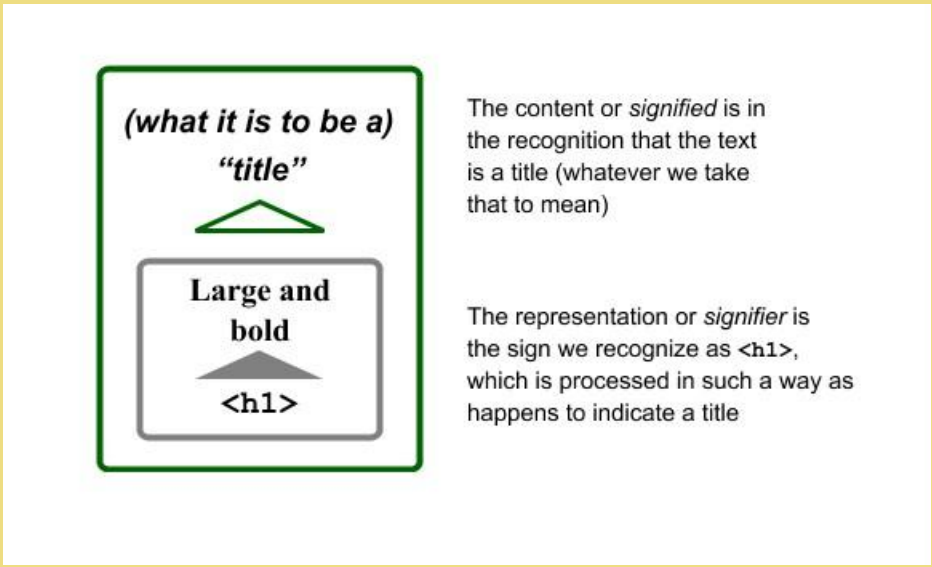


Figure 13 HTML as connotative system

The markup `<h1>` as a signifier for the operational semantics associated with the element type allow the deployment of such elements in HTML to signify titles to readers.

Of course, since there is slippage here between signifier and signified (not all large and bold texts are titles), this system can be rather unstable. `<h1>` by itself is not sufficient (or even necessary) to indicate a given text is a title, so other signifiers (for example constraints on where a title may appear in a text) are needed for disambiguation.

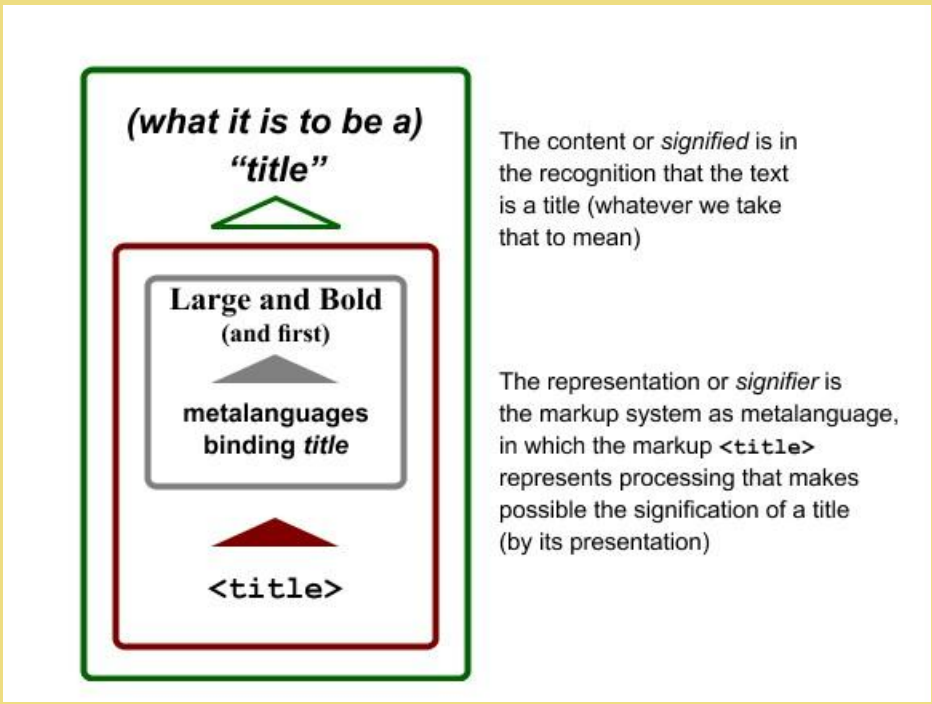


Figure 14 XML as a sign system

As a sign system, XML is a compound connotative system, in which the signifying role is played by a metalanguage. If you care to, you can even go deeper here; for example, depending on the implementation, there may be further signification intervening (e.g. through HTML) between a `<title>` element and its presentation. Yet since the constituent parts of this metalanguage complex (schema, transform) are formalized and automated, all a reader has to grapple with is its end result (what the metalanguage signifies), namely the formatting or other processing.

Note a key point about this arrangement, however complex we consider it to be: there is no way to rise from whatever *expression* of "titleness" we choose to settle on, to "titleness itself", *except by the perception of titleness as such*. That is, whatever formalisms we introduce, and whatever transformations we effect, all we can do is manipulate the sign that represents to us that we have a title. The arbitrary relation between signifier and signified remains a gap; as such, it is always slippery, since the human signification of a title does not finally depend on any formalisms that may happen to govern the operational semantics of the title's signifier (whether it be tag, formal declaration, or presentation) as a sign.

It can be stressed that it is by virtue of its operational semantics, although potentially it is never limited by them, that "descriptive" or generic markup in XML works. That is, it is because of the mediation of the conventional signification of a title to a reader (Figure 6) that the operational semantics of an XML element (Figure 9) can serve to signify its higher-order significance (Figure 8). Yet XML itself, as we have indicated, works by way of the application of one or more metalanguages. This combination of combinations is depicted in Figure 14, which apparently is a restatement of Figure 8 in view of the complex systems depicted in Figure 11 and Figure 12.

§ 4 From machine to meaning

4.1 *The role of formalism*

I have already described what operational semantics, or machine semantics, should be taken to be. Let me re-emphasize here how useful and important this view is, and how being able to constrain ourselves, pragmatically, to this view, is an important skill in design. Many design problems result from not being able to see into the stresses that can arise between operational semantics and a wider sense of meaning, particularly because it is not practical (in fact, in the nature of things it ultimately

will be impossible, for reasons we will see) to build an operational expression for every wider meaning, through an actual functionality in the system. Although operational semantics will never be all there is to meaning, they are all there will be to an automated system as far as its operations are concerned.

But an important distinction remains to be made. We can posit that a tag, say `<title>`, stands for or represents a given set of constraints and/or a given kind of processing on a given data element. Likewise, we can see how schema declarations, stylesheet templates or procedural code can signify how a `<title>` element is to be manipulated. In this respect, there is an ambiguity in our understanding of what is signifying what, as element types on the one hand, and (declarations of) constraints and processing over them on the other, may each be taken to signify or represent the other. For example, a declaration in a DTD may assert that certain constraints should be placed on the element type **title** (say it may appear inside a **section** and must be the first of its children), while the element type itself (or the `<title>` tag that signifies it) in a document instance, indicates that these same constraints are, or should be, in effect. Apparently, the reference or signification goes both ways; what's worse, since each of these is itself a compound sign, it becomes even more in question what is signifying what. An existential title? The big and bold processing? The constraints on placement? How do we account for this reversibility?

In fact, this reversability of signification is of the essence when considering operational semantics. It is a simple property of the fact that these semantics are effected through automated processes, and thereby are expressions of the operations of whatever formalisms we have introduced in order to achieve this automation.

The word "formalism" in this context requires some explanation. *Not* "definition", I hasten to add, since what is at issue is in one sense precisely whether something can be (or should be) "formally defined" — or rather, whether what we have when we have something defined formally, is sufficient to the purposes of signification. It would be an irony if by providing ourselves with a formal definition of the term "formalism", we thereby prevented ourselves from registering precisely the phenomena, or problem, in question.

Be that as may be, it is interesting to note that the term "formalism" can be found in a range of different disciplines, including mathematics, computer science, and philosophy, through religious studies (the first uses of the term in English apparently reflect a religious application) into literary criticism and folklore studies, where "Formalism" identifies a set of methods associated particularly with a particular Russian school of thought that goes by the name. And notwithstanding the unlikeliness that any or all these usages of the term should refer to any other, there does seem to be some commonality to the concept wherever we find it. Briefly, a formalism is a set of conventions or rules that can be applied, or whose application can be observed, irrespective of the particular local terms to which they are applied. That is, a formalism is the application of a given set of logical operations to a set of tokens without reference to what those tokens may be taken to stand for.

To bring this back into the context of semiotics, formalism thus designates whatever manipulations we can make of the system at the level of the signifier (consider again Figure 3). These operations are made in conformance with a set of abstract rules, namely logical operations that determine when we can substitute, when we can reverse, etc. For example, if a title is the first child of any section, then the first child of any section is a title — unless the title in the section is optional. Sometimes we can substitute: if a title is to be processed as large and bold, and the first child element of a section is a title, then the first child element of a section is processed as large and bold. And so forth. (Developers of markup applications can extrapolate from these crude examples to real cases.)

It is these rules that make formalisms so powerful when we deploy them as metalanguages: for one thing, it is by virtue of them that we can account for the "inversion" between Figure 14 and the several diagrams depicting metalanguages (Figure 11, Figure 12); in the former, `<title>` is a signifier for element operationally; in the latter, the metalanguage assures that operability by signifying

that *sign* in a further relation. Since this signification is formalized, the metalanguage as sign can in effect be turned "inside out", its operations taken as a given — just part of the system — by a user, who simply uses a tag to get results.

Such rules of equivalence, or the more complex rules of conditional operations that can be used to qualify them (possibly a first element in a section *may* be a title, but if any title is in a section, it must be first), must reduce at some point to formalisms (even if only *ad hoc* processing directives) if they are to be automatable. But the same thing is not true of sign systems in general. Formal rules of syntax may apply; but they are not in themselves the end of the story. Some formal operations are legal, but do not result in signification, such as (famously) "Colorless green ideas sleep furiously".[14] So formalism is necessary, but not sufficient, for the operation of sign systems that can generate new significations. To recapitulate earlier arguments, another necessary component is the relation, arbitrary but given (by convention or itself by some set of rules), between signifiers and signified: signifiers must be not merely tokens to be manipulated; they must also serve as signs. If not, these manipulations, however elaborate, are essentially sterile, "merely formalisms" in a derogatory sense. And yet even when the tokens we manipulate do rise to the status of signs, of course — the machine knows nothing about it.

4.2 Operational and emergent semantics

Yet we can know something about it. Despite the relative impoverishment of machine semantics compared to what human beings can do with signs in the very complex and nuanced ways we use them, evidently machines mean something; or we mean something by them. Given a better understanding of the way these layered systems work, it is now possible to extrapolate a more general relation between these kinds of semantic systems (diagrammed in Figure 15).

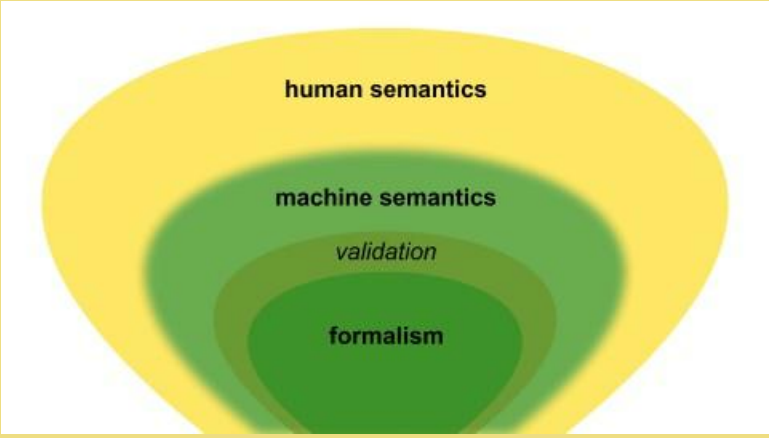


Figure 15 Machine semantics as a special type of semantics

Though in a working system, we may be oblivious to the specific operations carried out by the machine, nonetheless its semantics are a subset of human semantics: just as we mean the large, bold, centered text at the top to be taken as the title, so also we mean the element tagged `<title>` to be given such a format. That this intention is expressed through the machine does not make it any less human. The automatability (and thus, peculiarly, the obliviousness) of this transformation of signifiers is made possible by the formalisms at the core.

A key recognition is that there is a crucial resemblance between machine and human meanings as systems, in that in each, "meaning" is determined within and by the system as a whole, by its own internal rules, including both the formal operations of syntax and the signifier/signified relations that happen to obtain in the system's signs. This is a critical characteristic of these systems, and the source of their capabilities to generate new meanings. On the one hand, I can write a new paper, give it a title, and communicate something when I say "the title of my paper is *My Summer Vacation*" (although

in this particular case it may not be an especially new meaning). On the other, we can take that string of characters, `My Summer Vacation`, and display it in an appropriate font, just by virtue of having labelled it a `<title>`.

If the relation between these two types of naming is itself arbitrary in some sense (as we can see it is, since a reader can determine that the title of a school essay, as displayed or printed, is *My Summer Vacation* without knowing how it happens to be tagged in its source — or even that there is an encoded source file as opposed to marks on a page), nonetheless in a real system it is formally instantiated by the constraints and bindings of our processing. This is where our formalisms come in, which are of course deployed by means of our metalanguages. In the diagrams, the operation of these formalisms is depicted as a core within a capsule of machine semantics, itself within the more general field of human semantics.

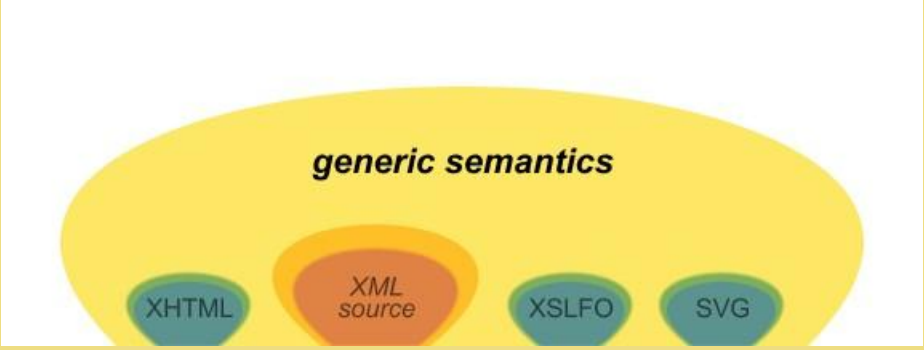


Figure 16 Machine expressions of a more general semantic system

If you were to change your perspective to look at these markup applications “down” from the top, they might resolve into the classic “hub-and-spokes” model of single-source generic markup. The source code is developed and maintained in a generic format; other formats can be spun off automatically. If the formalisms (schemas and conversion routines) are sufficient to constrain those expressions of the human semantics as well as they do the original source, data can potentially even be “round-tripped” back.

Figure 16 accordingly illustrates how several different markup regimens (markup languages as expressions of particular kinds of machine semantics) can separately express a single human semantics. The stress here is on the special advantages of a custom-designed XML source, constrained by appropriate validation, which reflects a set of abstractions that characterize the general semantic field (which in turn is typically conditioned by our perceptions and requirements). In effect, this diagram indicates how “generic” XML markup can provide for the familiar hub-and-spoke model of an XML application suite, in which other applications and media (as expressed by languages and formats that integrate them into the system) can provide the XML with their own kinds of expressiveness. In this context, it is worth contemplating how in principle it should be possible to move data freely among all these applications, if only we have unifying formalisms (again, validation in particular) that can assure the preservation of the integrity of the wider semantics of the data set *across* its various applications.

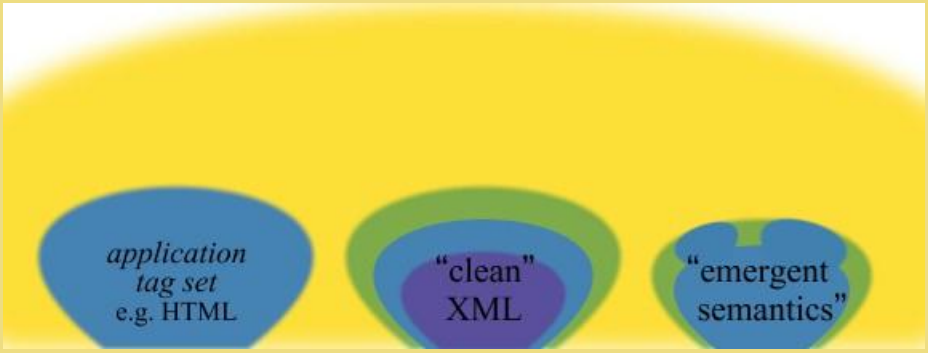


Figure 17 Machine semantics evading formalisms

Some markup languages may be nicely constrained by abstract models, while others (notably HTML) function effectively as control systems for browsers or viewers, and thus have no semantics apart from their behavior. Some are purely functional or media-specific; some are more generic. On the other hand may be a system where conventions, usages, and voluntary patterns of constraint *over and above* the way the markup is originally designed for, come into play — whether validated or not — giving a new "twist" to an old tag name, or going in an entirely new direction.

Also worth considering is what happens when a particular markup application is not bound by formalisms. Historically, for example, HTML has been like this: there has never seemed to be sufficient reason to validate HTML according to any formal definition, since it was all about how HTML *behaved* in the browser (and in other tools). Precisely because the browsers varied in behavior, however, the semantics of the tag set ended up reducing to some greatest common factor between what the "standards" stipulated and what the browsers implemented (and how). Consequently, HTML has been prone to abuse not only by users (who can get all kinds of effects out of tags despite the tags' purported meanings), but also in the hands of implementors and tools developers with their extensions and creative adaptations.[15]

At the other extreme are markup languages that are misapplied to domains outside their scope, or which are simply not designed well to begin with. In this case, what tends to happen is an adaptation of available operational semantics to express semantics even apart from any explicit arrangement made in the formalism (document type) for these semantics. When a tag set or element type hierarchy does not make provision for something to be performed or expressed (that needs to be expressed or performed), a workaround is often conceived; if this workaround takes the form of actually reinterpreting the available elements (with their given functionalities) to the purpose at hand, we have something designers and system engineers may call "tag abuse". So, because HTML lacked proper markup for page layout, table markup (which incidentally provides much layout information, since tables have to do this too) was adapted to serve in its place. Likewise, if HTML developers want to, they are (now) free to introduce a more benign (even "generic") semantics in the form of `class` attributes. Both these examples might be called "emergent semantics", since they show how functionalities are adapted and creatively deployed even apart from the semantics explicitly provided for by the tag set.

On the positive side," emergent semantics" can be viewed merely as a fair description of the development of patterns and best practices as a markup application evolves. Local customizations or subsets of broader standards, if done carefully and considerately, allow emergent semantics to be not only a way of building more expressive media, but also the engine of future kinds of applications and functionalities.

§ 5 Conclusions: knowing what we know

The foregoing consideration should be suggestive rather than prescriptive: if there is any usefulness in the view proposed, it will probably be an accidental outcome. Nonetheless, if there is any point to it, it is fair to wonder what conclusions we might draw from it.

It may be that while a sign system needs to exhibit formal features, that a fully worked out formalism is not necessary. Gödel's theorem already suggests that any formal system will manifest propositions that are "true", but that cannot be proven — in the case of a markup language, this may make for properties that are useful, though not instituted formally or instituted for ostensibly some other purpose. A reasonably complex sign system may be expected, if it is used widely, to exhibit emergent behaviors (whether these be "tag abuse" or not) as it is applied in different ways, and it is fair to expect that some of the more expressive and useful of these (especially those with the smallest negative impact) may come to be "best practice". In fact, this evolution is natural for any medium; there is no reason to suspect on line media would be any different. Conversely, it may be the case that much of what constitutes human semantics is often not determined by formalisms either explicit or unstated, but as much by habit (which is possibly to say by formalisms of a different sort), tradition, style and simple memorization. Then too, it is very clear that no system works only at one level — and that the very capability to refigure any given signifier — improvising signification — is what enables the construction of layered or "staggered" systems, metalanguages or connotations that either enable or undermine automated processes. Because these same principles of formal operability (operational semantics) in combination with the arbitrariness of the sign (which always implies the potential at least of broader semantics) work at many levels, moreover, the situation is indefinitely complex, self-referential and self-generative.

Beyond these fundamentals, however, there are also some implications of this view, and not only as it applies to the question with which I began my paper — the design and roles of schema technologies. Understanding something more about the nature of "semantics" as such also has implications for how we consider we should go about the design process for new applications. Finally, it may carry some warning about what kind of expectations we bring to technology development, and how our significations — human semantics — may always be one step ahead of what the machine does.

5.1 *Formal models, validation and maintenance*

Bringing to mind the problem of schema languages, their purposes and applications, one might well return to requirements. Tag sets, after all, are intended to provide solutions to the perennial problems of interoperability and interchange — of semantic integrity, one might say. In light of what we have observed about the fuzzy line between machine and human semantics, it seems reasonable to infer that a standard, abstract API[16] is not, perhaps, likely to be a comprehensive approach to the problem. An API is, by conception and application (and whether governed by a reference implementation or by a more abstract formalism), as close to the heart of machine semantics as a sign system can be. As such, it is an extremely useful thing: it provides the core of a machine semantic system, something that human semantics can successfully be layered on; but it does not specify the boundaries. Consequently an API shouldn't be made itself to carry the burden of all our wider meanings (since it is inevitable that sooner or later we introduce something new and unpredicted). It might be best to regard it as one formalism among many.

The problem here is not in any particular formalism, but in formalism itself. It is precisely because formalism is at the heart of the machine's capacity, that it cannot be relied on to mediate between the machine and wider meanings. Certainly there are cases where one formalism can express itself in another; and we can always try to unify on a particular formalism. We can also continue to deploy formalisms as metalanguages constraining metalanguages. But at the same time, there will never be one particular formalism that will always and fully express any other arbitrary (and hitherto unknown) formalism.

So if not through a formalism, how to do this? Perhaps the answer is in conceding that we must always have one foot, at least, in human semantics. Coming out of the recognition that although a semiotic system needs to manifest formal features, or regularity[17], those features need not ever have been explicitly formalized,[18] we might regard it as a positive opportunity to supplement our formalisms with other kinds of expression. Using a kind of literate programming approach, the specification of constraints on instances of a given document type would be stated through an open-ended range of expressive forms, not limited to notations that lend themselves to automated processing (be that a DTD, an RNG schema, a stylesheet, whatever), but also including natural language and other non-systematic (or at any rate, non-automatable) expressions. In the end, what we may need is a kind of prose specification with formal features, rather than the reverse.

While this comes at the price, one might think, of putting work into something that isn't itself amenable to automated processing — far better to have just the formalism! we think, then at least we can process it, and the formalisms will always be important touchstones for contract-making and interchange, for example — actually to fall back on natural language is the only approach that can preserve the virtues of descriptive or generic encoding as we have always conceived of it, as transcending any particular, everyday *application* of text encoding or markup. Our prose descriptions of those features that might constitute conformity to given abstract models are able to reach well beyond where an automated processor can go (can *yet* go), asserting, even while they cannot demonstrate in any processable way, how meanings may be interpolated into markup. Even as our formalisms improve, getting more and more sophisticated and powerful, our concepts of how our markup may be used and what outlandish applications of it there may yet be, can always stay a step ahead. And when our formalisms fail us, at least we have something to which we can return, to provide some direction at least towards working something out.[19]

Significantly, this line of argument has been anticipated in functioning systems. For example, the Text Encoding Initiative's editors have maintained both formal and prose specifications for TEI element semantics in a system called ODD. This amounts to a literate programming approach to schema development and maintenance [20] — with a key distinction. To whatever extent the intended constraints on element types cannot, or cannot yet, be reflected in formalisms, they can still be described in prose, allowing validation by hand or ultimately by means not feasible when the schema is first implemented. Other work worth citing in this context include an interesting proposal made in 2000 by Vorthmann and Robie [Vorthmann 2000] and the promising DSDL effort (Document Schema Definition Language [DSDL 2002]).[21]

So maybe what I'm talking about is not so mysterious. It has long been a truism in the markup industry — though lately it may have been forgotten in the flush of having new tools with which we can validate in new ways — that the definition of a document type is not complete without the human part, the prose description. A schema unaugmented with any indication of the "intentions" the tags might carry, isn't really enough to do the job. [22]

Even more deeply, maybe this is just an expression of a lesson we have already learned many times, a deeper truth that many computer programmers have discovered even if they haven't articulated it. The version of it I know, naming this particular formal expression after the person from whom I heard it, we can call Lapeyre's law: "If you can't explain it to me in words, I can't write it for you in code".

5.2 *Design strategies, standards development, chickens and eggs*

Close to many of us is a related problem, namely what strategy to adopt or recommend in the development of new tag sets, and not only how but when. This of course is a very practical problem, both within the established markup industry and wherever markup technologies are being applied for the first time. The basic question seems to be an expression of a fundamental stress. On the one hand, the technologies show a great capacity to be sensitive to local requirements, i.e., local semantics. In this respect their great flexibility recommends them over many alternatives, whether they be

relational databases — suitable for some types of information processing but not others — or other established applications that do one or another kind of data management, processing or formatting task. On the other hand, the promise of markup technologies in great part is that they ease interchange. This makes it a judgment call, sometimes a close one, whether and when a given enterprise waits for a standard, or whether it moves to develop local applications in defiance of interchange as a potential benefit. Of course in some respects this is a chicken-egg problem, since developing a standard is an undertaking best informed, properly, by local experience.[23]

The perspective offered by this paper cannot provide a solution to this: on the contrary, it suggests that it may not have an ultimate solution. Even a technical standard backed by a formalism will be subject to evolutionary pressures in its application. In effect, formalisms notwithstanding, one implication of considering markup technologies as sign systems is that they may share more with human languages than with their machine cousins: that is, they may be more like English, Latin or Javanese than like Perl, LISP or Java. In computer languages, meaning is effectively limited to machine semantics; to the extent that human semantics are layered on top (for example in careful naming of variables to provide for human maintainability of code), these are completely subordinate to the operation of the language once it is compiled or interpreted. This is not the case with markup languages, in part because their operation may be distributed much more widely (there is not a single implementation — even a single data model — to which tag sets in general must refer) and so even their machine semantics may vary across implementations, and in part because the combination of the layering of operational systems (metalanguages such as schemas and stylesheets) with the arbitrariness of the signifier provides for an ever-present potential for the semantics of the language to break out of what is known and understood, creating new meanings and applications (albeit not to the extent natural languages do). Again, we have seen this happen with HTML, where the `<table>` element's operational semantics — to lay out data in a tabular grid — proved to be essential, at one point in the web's history (1994-1995?), to providing a functionality that HTML desperately needed, namely page layout. But this came at the cost of ultimately preventing HTML table markup from ever being used for actual tables (data grids) in ways suitable to the online medium, such as providing scrolling tables under fixed table heads when a data set gets large. While it may be provable (for all I know) in general that for any added functionality provided by a case of using a tag in a new or particular way, there is at least one potential functionality lost, I suggest that an inherent property of any tag set that is more than trivial in its design is that, even given a fairly conservative expression (stylesheet), it is capable of being applied to unplanned uses; what is more, one person's tag abuse is always someone else's *ad hoc*, emergent semantics. Essentially it comes down to a question of authority; the calculus will vary considerably across communities of interest, domains and institutions how to balance between a top-down governance, and a bottom-up Darwinian model. Yet human beings adapt meanings to their own purposes regardless of what authorities say.

In effect, the particular problem faced by an enterprise or community over whether to borrow or build, and to what extent local processes should be adapted to the semantics of a standard, or to what extent the standard should be adapted to local requirements — or maybe the whole thing can be put off and someone else can decide — can be seen as just an instance of the more general, ongoing challenge in using markup technologies. There is not likely to be, in short, an answer to when the right time is for investment in markup technologies, since this question cannot be isolated from general strategic concerns about the means and methods of such an investment altogether, and in a given case there may not be a single right time, or even any.

Yet this dark cloud of unknowability does have a silver lining. However important standard tag sets for given domains prove to be, the greatest contribution of XML may be in its data models and capabilities for transformation — that is, in those formalisms closer to XML's core, its syntax and the semantics not of element and attribute types, but of elements and attributes *per se*, as logical constructs. Interestingly, the more unreliable given tag sets may come to be, the more valuable having a unified syntax will be at the bottom of it all. XML's logical model makes a staged approach to

development, in which investments are made gradually and strategically with an eye to returns both in the short and long terms, more feasible than ever. This suggests that at least to the extent that there fails to be a recognition of the general semantic "slipperiness" of markup, the emphasis on the tag set as such is perhaps misplaced, and should be balanced by a concentration on transformation technologies, conversion strategies, and sophisticated analytic and validation regimens, as a way both of mitigating the impact of local variability and "dialects", and of providing interchange across these boundaries despite their semantic mismatch.

Another encouraging thought is that if markup languages become more like the living things human languages are (maybe as unpredictable, maybe as upsetting), they may come to accommodate an analogous kind of expressiveness. Not all markup languages are written, or used, clumsily. To use them well, maybe, calls for having one's head into more than the immediate application — or nothing more, depending on whether it is the language and the potential wrapped up in whatever gracefulness it manages, or the application, that is the work of art. Either way, there are some strategies that should help. Understanding the way languages and metalanguages are deployed in the system, the way they are layered, the connections between points, above all the *audience* (or *audiences* as they are layered) — these will be the same whether designing or applying, whether using an installed system or building a new one.

Just so, designing languages with certain kinds of flexibility in advance can (again, depending on what the language is for) ease the problem of unexpected "repurposing" of structures and element types by identifying stress points and allowing for "stretch".[24] For example, it has always made sense to allow free subclassing of elements used for certain expected applications, be they database mapping or formatting (e.g. have a `CDATA` attribute named `type` or `class` if users are going to be developing their own stylesheets). Even without a schema, some developers and integrators are going to move forward with mappings or enhancement of tag sets, for some kinds of data; other systems will be more rigid (more schema-dependent), so will require explicit provision for these sorts of things. XML Namespaces (if and as namespace-sensitive validation starts getting commoner) may eventually have an influence on these problems as well.

5.3 *Signification and deception: the Trickster element*

But the least tractable, because the root problem, remains. For all the systems we build and metalanguages we deploy, we still cannot automate signification itself. What do the operational semantics of our tag sets do? They relate terms to one another; possibly they go so far as to do very complex things, like arranging type on a page. Yet when we get to signification itself, a leap is always required. Figure 13 and Figure 14 might depict this, across the gap between their signifier and signified: in order to get to what a title is, we have to create something that our intended audience will recognize as a title. There is no way around it. And while it may seem like a trivial point, it is easy enough to forget when we are taken up with the enthusiasm of automating some process or another. Yet no matter how long our chains of cause and effect, our machines will never intend "something be a title" in the same way a person might (or a person might by means of a machine), at least not without a radical leap not in machine power, but machine *participation* in our meanings.[25]

Some of the most persistent problems of the application of markup languages — why, in effect, their design and use is more challenging than earlier kinds of information processing systems regarded as works of engineering — are nothing but the flip side of what makes them powerful. Markup technologies are enabled by formalisms; but just as the slipperiness, the arbitrariness of our names cannot be confined by any formalism, so also the systems we build out of layers of metalanguage and connotative system will have ways of getting away from us.

Any time you have signification, the very fact that signifiers are slippery makes for the possibility of *deception*: deception is a metalinguistic relation to (manipulation of) a sign system: exploiting an assumption that a signifier/signified relation is dependable, it voids the signifier by deploying it misleadingly or fraudulently — the signifier without what it signifies. Between the truth-teller and

the con artist is the huckster, the spin-master, the fanatic; and maybe as often as not, we are deceiving ourselves. Whenever we get caught up in thinking about the anticipated powers of a new technology, therefore, we should give some consideration to what the hucksters, spin-masters and fanatics will do with it — not to mention what might be done with it by positively inimical interests.

Related to the huckster is the fortune-teller. Not all are carnival barkers, bringing us in; some may be prophets of doom. In between are ones who recognize that markup languages haven't just invented a way to communicate; they are inventing ways of *miscommunicating*.[26] This can go either way.

A growth industry for markup technologies ought to be auditing of all kinds. XML can expose a system, but can also obscure it. XML alone cannot guarantee that numbers are right. Data content can be entirely fantastic, while the markup gives it a machine-like air of consistency and completeness. Nonetheless, markup itself can be audited; and properly understood, it can shed light on whatever it reflects. And in principle, there is no reason why a properly authenticated and validatable data set could not aid greatly in transparency. Related to auditing is observation; there is no reason why markup should not flourish in data-collecting studies of all kinds: archeology, linguistics, economics, life and earth sciences.

It is when things get reflexive, complicated, arcane, that the Trickster starts getting involved. This pattern has been associated not only with both fraud and the arts of interpretation, but also with anything that inverts or confounds polarities such as human and machine.[27] Such a polarity is implicit, for example, in our ascribing "intention" and motive to the one (the human) but seeing the other (the machine) as mindless and unintentional, and in some way thereby *innocent*, even pure, free of guilt or responsibility but enormously powerful and capable of doing good. We blind ourselves to how the unintentioning agency can be set to the purposes not only of its creators, but also of other later users — and they may find something else to do with it, not at all what the creators intended.

Purpose can't be hard-wired. Wondering "what semantics means" finally engages us in questions of purpose, another meaning of the word "meaning". Purpose is not an internal state of some kind, some kind of organization of elements that can be modeled, whose operation can be replicated in one or another virtual kind of way; it comes from a relation to a context, the "why" as well as the what. *Context* is the part left out of discussions of machine semantics — odd how even the most enthusiastic proponents of glitzy technologies can only come up with mundane applications like knowing when you want the lights turned out at home, or getting you directions to a good restaurant — because context is the relation not only of a signifier to other signifiers (other codes) but more importantly to the vast realm of the signified (to say nothing of the unsignified). The purpose of a signifier is to relay a bit of information; the signified is what it relays. Operational semantics enact purposes because they bring about certain effects: they may, we say, serve certain functional requirements. But whether those effects serve their own purposes is something the system cannot govern. This is a gap at the foundation of any sign system, that ultimately its meanings and purposes are (and will always be) as much a matter of context as of content. No metalanguage can constrain it; the very fact that we can take an entire system including its metalanguages, and signify something *by* it as in a connotative system, always provides us a capability to evade. In the end, we may be able only to describe it, reflecting it as it reflects us.

Notes

1. By the time I have done with it here, I hope it becomes apparent why this discussion lacks rigor, and yet why it continues to be so tantalizing. Semantics, "meaning", like school, is an issue which matters to everyone, but in which no one recognizes an authority, since we are all experts.
2. Going straight to the source: Tim Berners-Lee, in *Weaving the Web*: "...we can program a computer to do simple things, like make a bank payment, and then we loosely say it

understands' an electronic check. Alternatively, a computer could complete the process by following links on the Semantic Web that tell it how to convert each term in a document it doesn't understand into a term it does understand. I use the word **semantic** for this sort of machine-processible [sic] relative form of meaning'. The Semantic Web is the web of connections between different forms of data that allow a machine to do something it wasn't able to do directly" [Berners-Lee 1999].

Two things happen here: Berners-Lee concedes that the senses of "understanding" and "meaning" that he imputes to machines (or to the Web itself) are metaphorical, at least in relation to the usual senses of those terms (as he says, "we loosely say"); then he appeals to a functional argument ("allow a machine to do something it wasn't able to do directly" — as if these machines do anything at all "directly") to assert that this is close enough. In effect, he sets aside the problem of what meaning and understanding actually are, in favor of something he can envision functioning. But how can we set aside the question of what meaning and understanding actually are, if the whole point of the Web is to facilitate human communication (meaning and understanding)? The unfounded assertion that "this sort of machine-processible relative form of meaning'" will be good enough — for something — is the hub of his argument, even if he has already allowed, in effect, that such machine meaning is relatively impoverished compared to the real thing.

3. The way the *TEI Guidelines* put it: "`<emph>` (i.e. emphasized) marks words or phrases which are stressed or emphasized for linguistic or rhetorical effect" [TEI P4].

4. Or only a very convoluted, circular way, based on the consequences of misapplying the tag in a different application from the one at hand, maybe an application that doesn't even exist. As an example, one vivid case of tag abuse that was once described to me was where users had, in a publishing system, taken to using a `<ship>` element to get text to appear in italics (i.e. to mark up rhetorical emphasis or some other feature commonly expressed through italics). Without being able to say "this is wrong because the content doesn't name a ship" (an appeal to a wider semantics than the machine can recognize), the only argument against this usage would be something along the lines of "this is wrong because if you ever made an index of ships, the list would be screwed up" (i.e. an argument based on a hypothetical use case for the element that might never exist).

   This problem will come up again when we consider the potential in markup systems for "emergent semantics" (Section 4.2).

5. Of course the locus classicus for this position is [Shannon 1948]. A very sophisticated treatment, with a candid consideration of the theory's strengths and limitations, may be found in [Ruelle 1991].

6. As Shannon puts it immediately in his second paragraph, "The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design." [Shannon 1948].

7. A simple search on the web, on the term "semiotic" or "semiology", is enough to indicate the great range and depth of semiotic studies of every kind. Several academic journals are devoted to the subject, some having been published for decades.

8. See an excellent survey of the development of the study of Semiotics in [Chandler 2001]. An English translation of Saussure's *Course* is in print; see [Saussure 1915]. As for Roland Barthes, see [Barthes 1964].

9. Also, looking forward to later: there is a distinction between a solid arrowhead, which indicates signification enforced or enacted by a formalism, and an outline arrowhead, which indicates signification through only partially-formalized means, typically "human assisted".

10. A very sophisticated (maybe too sophisticated) critic of the design and practice of HTML might assert that the operational semantics of HTML's `<h1>` element only "happen to be" that text in an `<h1>` will come out large and bold. Likewise, in view of the way in which markup systems are generally implemented, one might quibble that it is not the `<h1>` *markup* (the start and end tags) that make for the text to be large and bold, but rather that they merely serve to designate (signify) an abstract "h1" *element* in an object model, which receives the given operational semantics. Thus the machine semantics of a tag as tag (parsing the markup syntax) and of the element type as element type (processing the document model) are distinguished.

11. Yet I would argue that the reverse is likewise true, that designers who think only in functional and operational terms are less able to design well than those who are also able to think more abstractly about the ontology or "world" their element types imply.

12. The stress between these two views is at the heart of the long-standing discussion within the markup technologies community on "descriptive" and "procedural" markup. See [Renear 2000] and [Piez 2001], and works cited there, for more.

13. In passing, however, it is worth noting that one metalanguage is not diagrammed here: XML itself. What XML describes is a syntax (an arrangement of characters) and a language for describing a grammar for that syntax: well-formed XML and DTD syntax, respectively. If we were to diagram this, on the signifier side would be the XML Recommendation itself; what is signified is the sign system constituted by the relation between XML as a format (lexical or physical arrangement of entities) as signifier, and XML as a logical structure (elements and attributes), as signified. All the debate over infosets and object models constitutes our ongoing discussion of what that signified can dependably be taken to be — an important issue because this thing, whatever it is, is in turn going to serve to provide signifiers in other sign systems (XML applications).

14. A sentence coined by Noam Chomsky to demonstrate how syntax can operate even apart from meaning. Interestingly, the words in this sentence can be reconstrued in such a way as to make sense: "colorless"means "undistinguished, without quality"; "green " means "fresh, new, untried" etc. The reconfigurability of the semantics of these tokens so as to adapt themselves in this way, however, to the collective demands of their context, only goes to demonstrate the general point. See **http://www.emich.edu/~linguist/issues/2/2-457.html** for more along this line.

15. It is an open question, however, whether HTML generally would have benefited through the earlier stages of its evolution from non-proprietary validation regimens. Such overhead and coordination would have required resources to adopt and maintain, and HTML benefitted from being relatively lightweight. Moreover, there was almost certainly a benefit in the discovery of masses of new developers of the virtues of simplicity, external specification and validation, all earned the "hard way", as well as allowing hothouses-full of new tools to thrive.

16. Such as DOM AS (abstract schema). See [DOM AS 2002]. I do not mean to imply, by citing it in this context, that DOM AS characterizes itself as an effort to solve the problem

of schema coordination or unification: far from it. But it is natural to assume that some implementors will turn to it for these purposes, if only because systems based on the W3C DOM will find it to be a lowest common denominator among candidate formalisms.

17. See [Barwise 1997], especially section 1.2, *Regularity in Distributed Systems*.

18. And more than this, that formalisms are emergent properties of the systems themselves, not only by design but by evolution (or more strictly, by evolution of design), and therefore can be recognized *ex post facto*, not having been "placed there" by anyone.

19. The law, literary and academic genres, and manufacturing systems all serve as examples of systems based on prose (albeit never based entirely on prose but on other machineries as well), so we know systems like this can work, given time and energy.

20. See [Rahtz 2002].

21. "The objective of developing the Document Schema Definition Language is to create a framework within which multiple validation tasks of different types can be applied to an XML document in order to achieve more complete validation results than just the application of a single technology." See [DSDL 2002].

22. "Part of a document type definition can be specified by an SGML document type declaration. Other parts, such as the semantics of element and attributes, or any application conventions, cannot be expressed formally in SGML. Comments can be used, however, to express them informally".[ISO 8859 4.105].

23. This dilemma is portrayed at length and in some detail in the April 2002 report of the U.S. Government GAO (General Accounting Office) on "Electronic Government" [GAO 2002].

24. This is what Liam Quin's1996 advice amounts to (see [Quin 1996]).

25. Accordingly I suspect we won't see an artificial intelligence we can relate to until we solve the problem of the machine's sensorium and bring it into the same world we inhabit. A machine that knows what it means to us for the lights to be out, will be a different kind of creature from one that merely has a `lightsout` flag set somewhere. While a net of sensitivities might be engineered, and might be a useful thing, even without being able to put it all in context (that is being able to see what we see and feel what we feel), still it stops short of intelligence. Adaptability would help, but then too, the machine's I/O is still very limited. We might get insect intelligence out of it. Until the sensorium improves and the machine starts to have analogous experience to ours.

26. Worth careful consideration, not simple dismissal, is Walter Perry's argument vis-a-vis the dangers of standard vocabularies [Perry 2002].

27. See a fascinating book by George Hansen [Hansen 2001]. To say that these polarities may be inverted is not to argue that they are natural or a normal order of things.

Bibliography

[Barthes 1964] Barthes, Roland. Elements of Semiology. 1964. Annette Lavers and Colin Smith, translators. New York: Hill and Wang, 1967.

[Barwise 1997] Barwise, Jon, and Jerry Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge Tracts in Theoretical Computer Science 44. Cambridge: Cambridge University Press, 1997.

[Berners-Lee 1999] Berners-Lee, Tim. *Weaving the Web: the Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. New York: HarperCollins, 1999.

[Chandler 2001] Chandler, Daniel. Semiotics for Beginners. **www.aber.ac.uk/media/Documents/S4B/sem02.html**.

[DOM AS 2002] *Document Object Model (DOM) Level 3: Abstract Schemas and Load and Save Specification*. Version 1.0. W3C Working Draft 14 January 2002. **http://www.w3.org/TR/2002/WD-DOM-Level-3-ASLS-20020114/**

[DSDL 2002] Holman, G. Ken. *ISO/IEC 19757 - DSDL Document Schema Definition Language*. At **http://dsdl.org** (April 1 2002).

[GAO 2002] United States General Acccounting Office. Report to the Chairman, Committee on Governmental Affairs, U.S. Senate. *Electronic Government: Challenges to the Effective Adoption of the Extensible Markup Language*. GAO-02-327. April, 2002.

[Gencode 1983] Graphic Communications Association. *GCA Standard 101-1983: Document Markup Metalanguage; Gencode and the Standard Generalized Markup Language* Arlington, Virginia: 1983.

[Hansen 2001] George P. Hansen. *The Trickster and the Paranormal*. XLibris, 2001. **http://www.tricksterbook.com**

[ISO 8859 4.105] ISO 8859: Standard Generalized Markup Language (SGML). Note to section 4.105. As printed in Charles Goldfarb, *The SGML Handbook*. Oxford: Clarendon Press, 1990: 126, 264.

[Perry 2002] *Standard Data Vocabularies Unquestionably Harmful*. XML.com, May 2002. http://**www.xml.com/pub/a/2002/05/29/perry.html**

[Piez 2001] *Beyond the descriptive vs. procedural distinction*. Extreme Markup Languages 2001 (Montreal). Reprinted in *Markup Languages: Theory and Practice*, 3, no. 2 (Spring, 2001).

[Quin 1996] Quin, Liam. November 1996. *Suggestive Markup: Explicit Relationships in Descriptive and Prescriptive DTDs*. SGML'96 Conference Proceedings. Graphic Communications Association.

[Rahtz 2002] Rahtz, Sebastian. *Converting to schema: the TEI and RelaxNG*. XML Europe (Barcelona), 2002.

[Renear 2000] Renear, Allen. *The descriptive/procedural distinction is flawed*. Extreme Markup Languages 2000. Reprinted in *Markup Languages: Theory and Practice*, 2, no. 4 (Fall, 2000).

[Ruelle 1991] Ruelle, David. *Chance and Chaos*. Princeton Science Library. Princeton: Princeton University Press, 1991.

[Saussure 1915] De Saussure, Ferdinand. *Course in General Linguistics*. Trans. Wade Baskin. The Philosophical Library, 1955. Reprint New York: McGraw-Hill, 1966

[Shannon 1948] Shannon, Claude F. *A Mathematical Theory of Communication*. *Bell System Technical Journal*, July and October, 1948. Reprint available on line at **http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html**.

[TEI P4] Sperberg-McQueen, C.M., and Lou Burnard. *Guidelines for Electronic Text Encoding and Interchange*. Chicago, Oxford: 1990-1994. Revised Reprint, Oxford: 1999.

[Vorthmann 2000] Vorthmann, Scott, and Jonathan Robie. *Beyond Schemas: Schema adjuncts and the outside world*. *Extreme Markup Languages 2000* (Montréal): 249-255.

The Author

Wendell Piez
Mulberry Technologies, Inc.
17 West Jefferson St.
Suite 207
Rockville
MD
20850
wapiez@mulberrytech.com
tel: (301) 315-9635
http://www.mulberrytech.com

In school, Wendell Piez studied Classical Literature, Poetics and Rhetoric, receiving a Ph.D. in English from Rutgers University in 1991. His work with markup languages dates from 1994, when he first became involved in Humanities applications of SGML. A few of his experiments in English verse form may be read on the web at **http://www.piez.org/wendell/poetry.htm**.

Extreme Markup Languages 2002
Montréal, Québec, August 6-9, 2002
This paper was formatted from XML source via XSL
Mulberry Technologies, Inc., August 2002