Knowledge Representation in Humanities Computing

by

John Unsworth

Best viewed with Internet Explorer 5 or higher

The assertion of this paper is that the methodology known as knowledge representation has profound implications for humanities computing, and through humanities computing, has the potential to change the way humanities scholarship is done, to change the nature of graduate education in the humanities, and to change the relationship between the humanities and other professions, let alone other disciplines. I believe that knowledge representation has already produced important new research, and will, in the future, bring us new insights into what we know about the human record, and how we know it. Anyone who looks, broadly, around the field of humanities research, or who knows deeply the history of research paradigms within a particular discipline, will recognize this as a moment in which disciplines are dithering: in many fields, in many areas, there is no compelling methodology, no lens that seems to focus our attention on important new questions to be asked and answered, and important new ways of asking and answering. For humanities computing, knowledge representation is a compelling, revelatory, and productive way of doing humanities research--and in many ways, it is what humanities computing has been doing, implicitly, for years.

Having made those claims, I must also say that the computer, itself, is not the guru in this picture. If anything, representing the human record in computable ways teaches one the truth of Emerson's dictum, that "a foolish consistency is the hobgoblin of little minds"—indeed, the computer **has** a very "little mind." But even so, there's an enormous return on the exercise of foolish consistency. To put it another way, there's definitely something to be learned from the discipline of expressing oneself within the limitations of computability. Without a doubt, something's lost in that exercise as well, and I wouldn't argue that the methodology I'll be discussing in what follows is the best or the only method—but it is certainly an illuminating one in many cases.

The approach that I want to discuss today—the particular engagement with humanities material on the one hand and computers on the other—is called *knowledge representation*, a sub-discipline in the field of artificial intelligence, but also an interdisciplinary methodology that combines logic and ontology to produce models of human understanding that are tractable to computation. I'm interested in knowledge representation because it offers a very exact description of what we've been groping toward (with varying degrees of intentionality and varying degrees of success) in the various projects at the Institute for Advanced Technology in the Humanities, and because it is a methodology that I believe will be at the core of the University's new MA in digital humanities that I've also been working on. I don't claim expertise in this field, and I'm grappling now with things I have never mastered, or had to master, in the past—formal logic, mathematical constructs, philosophy—but I'm aware as I grapple that I already know some of this, that some of the premises of this method are lessons I've learned from experience.

So, let's proceed to **Figure 1,** and a definition of terms. "What is a knowledge representation?" These MIT AI researchers argue "that the notion can best be understood in terms of five distinct roles it plays, each crucial to the task at hand:"

1. A knowledge representation (KR) is most fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e., by reasoning

about the world rather than taking action in it.

With respect to the texts and contexts I work in and around, knowledge representation usually takes the form of markup (in extensible markup language, XML, or in Standard Generalized Markup Language, SGML), or it takes the form of databases—flat tables or relational structures. We'll see examples of each in a moment: both are clearly "substitutes for the thing itself:" the entity that uses those surrogates to determine consequences is, in one sense, a piece of software (a search engine, for example), but in another sense it is a human being, who comes to understand the consequences of his or her own assumptions and beliefs by making them explicit and then putting them in play with one another and with a text.

2. It is a set of ontological commitments, i.e., an answer to the question: In what terms should I think about the world?

This is very important to understanding what one does when producing these surrogates. There is a logical component, but it is very basic and content-neutral—just some Boolean operators, some rules of logic at that level. The more complicated stuff happens when you start to name the predicates for your predicate calculus—the things of which these logical statements are true, or false. The names you use imply a perspective, a purpose, and more than that, an understanding of the constituent elements of the object of your attention, and their relations to one another in that object. We'll see a simple example in the context of the Salem project in a moment.

3. It is a fragmentary theory of intelligent reasoning, expressed in terms of three components: (i) the representation's fundamental conception of intelligent reasoning; (ii) the set of inferences the representation sanctions; and (iii) the set of inferences it recommends.

Intelligent reasoning about a literary text, for example, may take many forms: a materialist critique, a philological study, a critical edition, a cultural history. The "fundamental conception of intelligent reasoning" differs somewhat in each case, as do the inferences sanctioned and the inferences recommended under each model.

4. It is a medium for pragmatically efficient computation, i.e., the computational environment in which thinking is accomplished. One contribution to this pragmatic efficiency is supplied by the guidance a representation provides for organizing information so as to facilitate making the recommended inferences.

This is very important—as Sowa says (next slide), the requirement of computability is what separates knowledge representation from pure philosophy, and it is also what brings me, and the projects at IATH, in contact with knowledge representation. If you want a computer to be able to process the materials you work on, whether for search and retrieval, analysis, or transformation—then those materials have to be constructed according to some explicit rules, and with an explicit model of their ontology in view.

5. It is a medium of human expression, i.e., a language in which we say things about the world.

It may not look like it, when you look at markup, but that is a language that both humans and computers can understand, and it is a language in which we say things about the world.

Figure 2 is a kind of counterpoint, between an email message distributed on March 19th, 2001 through the humanities computing mailing list called Humanist, and the preface of the New Big Book on Knowledge Representation (called Knowledge Representation) that just came out from John Sowa, a well-known researcher in this area. The email message was written by David Halsted, a comparative literature Ph.D. with interests in history, democracy, and eastern Europe—and until recently, a project manager at H-Net, "an international interdisciplinary organization of scholars and teachers dedicated to developing the enormous educational potential of the Internet and the World Wide Web" (based at

Michigan State University). I'm interested in this email message because it testifies to a connection between philosophy and computing that is pivotal to the rest of what I want to talk about here, and does so in what are, for me, fairly concrete terms.

For what it matters, I picked up _Truth and Method_ recently after a long absence and a career change. Computing got me back to hermeneutics because the process of defining what people want a computer to do is so hard. I started reading some of Terry Winograd's books on design, and that got me back into questions I hadn't really thought about much since grad school. Somewhere between Winograd talking about people's inability to articulate fully what they do every day (yeah, he got that idea from Heidegger) and Frederick Brooks talking about the process of programming as "debugging the spec" in _Mythical Man Month_ I found myself right back at questions of interpretation and the articulation of an unspoken lifeworld.

Socrates said he was the midwife to his listeners, i.e., he made them reflect better concerning that which they already knew and become better conscious of it.

Then I picked up Gadamer again, and then I got a new job a month ago and haven't had time to think, but a lot of time to see what happens when unspoken assumptions don't get written out into a formal description by people who can't visualize what a program is (again, Brooks).

Like Socrates, knowledge engineers and systems analysts play the role of midwife in bringing knowledge forth and making it explicit. They display the implicit knowledge about a subject in a form that programmers can encode in algorithms and data structures.

Anyway, that's how I got back to Gadamer from computing. The implementation of a spec (where a spec can be quite informal) in a program appears as a kind of critique/interpretation of the text of the spec . . . . So I'd say interpretation, and the classic hermeneutic pattern of working between the part and the whole toward an understanding, are central to whatever it is we're doing when we try to create working programs for carbon-based life forms capable of linguistic misunderstanding.

"The implementation of a spec in a program appears as a kind of critique of the text of the spec"—which is to say, when you sketch what you think you understand, and then try to turn that sketch into a series of instructions a computer can execute, you find out where there are holes in your sketch, where it breaks down. Similarly, when you design an information structure and then try to fit the actual information into it, you find out where the structure doesn't fit. That's the back and forth between part and whole, rule and instance.

And before we leave this Figure, I'd like to focus attention on the tripartite structure that Sowa lays out for knowledge representation: it consists of **logic, ontology, and computation**. Logic disciplines the representation, but is content-neutral. Ontology expresses what one knows about the nature of the subject matter, and does so within the discipline of logic's rules. Computability puts logic and ontology to the test, by producing a second-order representation that validates and parses the ontology and the logic of the knowledge representation.

Figure 3 shows a database table from Ben Ray's Salem Witchcraft project—a "tuple." A tuple is a simple (but powerful) data structure containing two or more components—for example, a row (with two or more fields) in a table in a database. A tuple is a structure that can be repeated any number of times, always with exactly the same parts—a set with a certain number of elements. The simplest tuple would be a matched pair—out of which, properly deployed, some very complicated things can be built.

Logic: The logic here is that of the tuple—and, and, and. Of course, some of these elements can be null, and some cannot be null, so it's not the case that every row in the database will have every field filled out.

Ontology: With respect to ontology, the first thing to be said is that the database schema can enforce some data-typing (which is a level of ontology in itself, with respect to the data) But at a higher level, the table at the top left in Figure 3 expresses a certain ontological view of marriage, a demographic view, but also a western, judeo-christian view. It says that marriage has a begin-date and an end-date, a place, a unique husband and a unique wife, and so on. Admittedly, these are pretty basic ontological commitments with respect to marriage, but they still express a perspective and imply a purpose.

The right-hand table in Figure 3 shows the set of tables in this database, and suggests the larger ontology of Salem that's in play here, with a focus on family relationships, corporate entities, property, sources (manifestations) and events.

Computation: The animation at the bottom shows one way—visualization—in which this knowledge representation of Salem, its individual residents and the events in which they are involved, is computed. In order for Ben Ray to provide us with this visualization of accusations played out over space and time, and across households and townships, the knowledge representation according to which his data is prepared must capture all these ontological features ("husband" and "wife" for example) according to some pre-defined logical rules ("every husband has one and only one wife; every wife has one and only one husband; etc.). And then software has to compute this all—in this case, a Flash presentation of a map, with timeline, on which data from the Salem database is plotted.

Interestingly, in talking with Ben Ray recently about this project and how it changed his research, I heard him say two things (unprompted by me) that resonate with earlier descriptions and discussions of knowledge representations:

1. Research in such a framework constantly requires you to test your particular assumptions against your general model--you continually validate the particular with respect to the general, and vice versa.

Research in such a framework discourages you from accepting and relying on knowledge representations produced by others—in the form of existing indices, for example. You want instead to generate those things on the basis of your data, your ontology, etc..

Figure 4 shows a knowledge representation from the Blake Archive, in the form of an SGML DTD designed to represent the illuminated books of William Blake, and then shows that knowledge representation applied to and computed with respect to an instance, Blake's Songs of Innocence and Experience—in this example, Copy C, plate 9 of that work, which presents the poem "The Lamb."

The Document Type Definition (DTD), visualized at the top in a piece of software called Near and Far, specifies the possible logical and ontological representations and relationships within particular instances marked up with this DTD. A DTD, as you can see from this example, is a tree structure with conditional branching, a more complex data structure than a database table.

Logic here would be found in the operators that govern the elements in the document type definition--the set of rules that says what tags can occur in the document, where they can occur, what they can contain, and what can contain them, whether they are repeatable, and so forth. Here's the relevant section of the Blake Archive DTD:

```
<!ELEMENT illusdesc             - -        (head?, illustration+)        >

<!ATTLIST illusdesc        %global; >

<!ELEMENT illustration          - -  (head?, (illusobjdesc,

                                                        ((characteristic |

                                                        transcription)+  |

                                                        component+))) >
```

```
<!ATTLIST illustration
                                        %global;
                              type (interlinear | plate) #IMPLIED
                              location  CDATA  #IMPLIED >
<!ELEMENT illusobjdesc              - -  (%m.phrase; | cit | note | quote)+ >
<!ATTLIST illusobjdesc
                                        %global;  >
<!ELEMENT component                 - -  ((characteristic | illusobjdesc |
                                                        transcription)*, component*)>
<!ATTLIST component
                               %global;
                     type (figure | structure | animal | object |
                                        vegetation | text ) object
                              location  CDATA  #IMPLIED  >
<!ELEMENT characteristic            - -  (%datanote;) >
<!ATTLIST characteristic
                                        %global;  >
<!ELEMENT transcription             - -  (lg)* >
<!ATTLIST transcription %global; >
```

In this example, the following standard SGML notation conventions are in use:

Sequence:

Sequential elements, in order specified = , (comma)

Elements are alternatives (OR), no particular order = | (vertical bar)

Elements are included but in no particular sequence (AND) = &

Occurrence:

Optional element, single occurrence if present = ?

Optional element, may occur more than once = * (asterisk)

Required element, single occurrence = unsigned element name

Required element, must occur at least once, can occur many times = +

Terms like “must”, “or”, “at least one” and so on are logical conditions or operators.

Ontology here, as in Salem, is found in the naming of parts and the specification of their relationships. The names for the elements and attributes are made up by the DTD designer—they are intended to name or suggest ontological characteristics or aspects of the object to which they refer. The computer doesn’t “read” these names in any semantic sense, but it does enforce consistency in the use of names, and it enforces compliance with the rules laid out in the DTD to govern the way these elements and attributes are deployed in an instance of marked-up text. In this fragmentary example, we see that an illustration element may have a head element within it, and it must have an illustration object description and that description must be followed by at least one characteristic or transcription or at least one component:

```
<!ELEMENT illustration          - -  (head?, (illusobjdesc,

                                                    ((characteristic |

                                                    transcription)+  |

                                                    component+))) >
```

The specification that an illustration consists of these things is an *ontological commitment*, based on (and producing) a certain view of the object in question, in this case, one of Blake’s plates.

The particular instance, at lower left in Figure 4, applies the Blake Archive knowledge representation, in the form of markup, to a particular case: Songs, Copy C, Plate 9. This illustration description says that there is an illustration of type “plate” applying to location E (i.e., the whole plate), and gives a description of the objects within the whole illustration, then individually lists illustration components and locates them according to a grid system in which E is the whole, A is the upper left quadrant, B the upper right quadrant, C the lower left quadrant, and D the lower right quadrant—so, component of type “figure” is to be found at location D, or in other words the lower right-hand quadrant, of the plate, and that component has the characteristics “male, child, nude, standing, contrapposto, etc..

Computation: The lower-right window in figure 4 shows the computation of the knowledge representation applied to this instance—computation, in this case, being the search for the term “contrapposto” (the position of a human figure in painting or sculpture in which the hips and legs are turned in a different direction from that of the shoulders and head; the twisting of a figure on its own vertical axis), and the retrieval of plate descriptions that use that term to name characteristics of components in illustrations on plates. There are a couple of other computational steps in this process, too, not shown in Figure 4.

First, there’s a Java program that reads the Blake Archive SGML markup, and on the basis of that markup, finds the image file for a particular plate, opens that file and gets its dimensions, determines which quadrants of the image have descriptive textual material associated with them, calculates the x,y coordinates of those quadrants, and writes out a separate XML data structure, the overlay file for Inote —the software that presents the image at the end of the search process—with those coordinates and descriptions.

Second, there’s a stylesheet being applied by software (Dynaweb—the SGML-aware Web Server through which the Blake Archive is published) to the instance that matches our search criterion. The stylesheet is its own knowledge representation, but one whose ontology is concerned with the presentation, rather than the structure and content, of the Blake Archive information—but it is still informed, very distinctly, by a perspective on the content, by ontological commitments regarding the nature, value, and relations of the material being presented, and so on.

The fact that the computer can find, on request, human figures twisted on their vertical axis is not a result of artificial intelligence in the computer, but it is a result of some highly trained humans having given formal expression to what they know about the ontology of Blake's works, and then having a software process parse that formal expression to find (first textually, then graphically and spatially) the representation that matches our search criterion.

I think the Blake scholars involved in this project, Joe Viscomi, Morris Eaves, Bob Essick, were they here, would testify to **both** the frustration provoked by the 'foolish consistency' that the computer requires of them, in order to comply with the knowledge representation we have collectively developed to make their understandings of Blake computable, **and** the surprising illuminations they have come upon in the midst of their frustration.

For an example of the kind of reflexive illumination that can be produced by forcing humanities scholars to endure the frustration of working with computers that require absolute logical consistency, see Figure 5. In this example, we are discussing concrete intellectual issues of representation provoked by the literal-mindedness of a certain piece of software—called EBT (for Electronic Book Technology) then, now Dynaweb.

Figure 6 shows a representation of the data and structure of Jerome McGann's Rossetti Archive, produced by a computer program (Graphviz)—a visualization of the actual content of the Rossetti Archive and the connections made explicit in the course of systematically applying a knowledge representation to representations of Dante Gabriel Rossetti's writings and pictures: ergo, a representation of a representation. This visualization shows that the data structure of the Rossetti Archive is more of a lattice than a tree: it has many cross-connections, few dead ends.

This graphic also demonstrates that you can generate complexity by applying some fairly simple rules to a large amount of information, especially when the information is itself originally produced by a human being.

And, finally, Figure 6 is an example of the analytical and expressive power of knowledge representations, and at least one kind of answer to the question "why would you do this?" The computer can deal with *far more* information than you can, and even though it can't (yet) reason, it can show you opportunities for reasoning you would never find without it.

In short, the intellectual outcomes in all of these examples are as follows:

- developing a knowledge representation requires you to make explicit what you know about your material,
- applying a knowledge representation shows you the ways in which your material exceeds or escapes your representation,
- computing a representation of large amounts of material that has been encoded and processed according to a rigorous, well thought-out knowledge representation affords opportunities for perceiving and analyzing patterns, conjunctions, connections, and absences that a human being, unaided by the computer, would not be likely to find.

The process that one goes through in order to develop, apply, and compute these knowledge representations is unlike anything that humanities scholars, outside of philosophy, have ever been required to do. This method, or perhaps we should call it a heuristic, discovers a new horizon for humanities scholarship, a paradigm as powerful as New Criticism, New Historicism, or Deconstruction—indeed, very likely more powerful, because the rigor it requires will bring to our attention undocumented features of our own ideation, and coupled with enormous storage capacity and computational throughput, this method will present us with patterns and connections in the human record that we would otherwise never have found or examined.