The inadequacy of embedded markup for cultural heritage texts

Desmond Schmidt
Information Security Institute,
Queensland University of Technology, Queensland, Australia

Correspondence:
Desmond Schmidt,
Information Security Institute,
126 Margaret Street,
Brisbane, Queensland, 4001,
Australia
E-mail:
schmidda@qut.edu.au

Abstract

Embedded generalized markup, as applied by digital humanists to the recording and studying of our textual cultural heritage, suffers from a number of serious technical drawbacks. As a result of its evolution from early printer control languages, generalized markup can only express a document's 'logical' structure via a repertoire of permissible printed format structures. In addition to the well-researched overlap problem, the embedding of markup codes into texts that never had them when written leads to a number of further difficulties: the inclusion of potentially obsolescent technical and subjective information into texts that are supposed to be archivable for the long term, the manual encoding of information that could be better computed automatically, and the obscuring of the text by highly complex technical data. Many of these problems can be alleviated by asserting a separation between the versions of which many cultural heritage texts are composed, and their content. In this way the complex interconnections between versions can be handled automatically, leaving only simple markup for individual versions to be handled by the user.

1 Introduction

This article is a critique of modern generalized markup systems as applied to the encoding of cultural heritage texts in the humanities. Discussion of markup inadequacy has thus far focused mainly on the problem of 'overlapping hierarchies' (to be defined below), which is mainly a problem for corpus linguists. The more diverse and difficult problems faced by digital humanists have received relatively little attention. There have been a number of recent theoretical discussions on markup adequacy for cultural heritage texts (Raymond *et al.*, 1992; Renear, 1997; Buzzetti, 2002; Neyt, 2006) as well as several papers discussing the problems of encoding specific texts or collections of works (Mah *et al.*, 1997; Vetter and McDonald, 2003; Bart, 2006; Vanhoutte, 2006), but what is so far lacking is a discussion of the technical limitations of markup systems in this context.

This article is divided into four sections:

(1) *The Introduction* defines the terms and limits of the article's discussion.
(2) *The History of Markup* establishes why and how markup was adopted by humanists, and its connection with the print medium.
(3) *The Technical Deficiencies of Markup* establishes the serious flaws in embedded markup as applied to cultural heritage texts: (a) the overlap problem, (b) the embedding of potentially obsolescent technology and interpretations into the text, (c) the manual encoding of information that can be better computed automatically and (d) the fact that markup is a complex textual command language, not a modern graphical user interface.

doi:10.1093/llc/fqq007 Advance Access published on 16 April 2010

(4) *Multi-Version Documents* (MVDs) describes an already working solution to most of the deficiencies of the previous section that uses light embedded markup in combination with a separate layer for versions.

1.1 Definition of cultural heritage texts

'Cultural heritage texts' are historical works that have become an object of study. The subject matter may be of any type including mathematics or science, and the work itself may be in any form, even correspondence or a journal. In spite of this variation in content and form, the study of the text's own composition: for example, evidence of its existence, how it is written, what it is, how it is structured, etc., is normally reckoned as part of the humanities domain. Some of the problems and solutions described in this article may also apply to texts of the present and future.

1.2 Definition of markup

The term 'markup' appears to be a neologism, derived from the 'mark-up' instructions inserted by designers into manuscripts intended for printing (OED). Contrary to this etymology, Coombs *et al.* (1987), Sperberg-McQueen (1991) and Raymond *et al.* (1992) all claim that markup has been with us for centuries in the form of spaces between words and punctuation. By this they appear to mean that spaces and punctuation are a kind of markup distinct from markup in its purely computational sense. In XML, markup is clearly distinguished from the text: everything between and including pairs of angle brackets, and the white space used to format it, constitutes markup, while the rest of the document is content (Bray *et al.,* 2008, Ch. 2.4). But they are also aware of the more formal definition: 'Markup is the use of embedded codes, known as tags, to describe a document's structure, or to embed instructions that can be used by a layout processor or other document management tools.' (Raymond *et al.,* 1992, p. 1). 'By *markup* I mean all the information in a document other than the "contents" of the document itself' (Sperberg-McQueen, 1991, p. 35).

Although Renear appears to reject Coombs's idea that punctuation is a kind of markup, he still sees it as embodied in the formatting information inserted by WYSIWYG word-processors (Renear, 1997, p. 109). But here, too, a distinction must be drawn between the data structures employed by word processor programs, which use text ranges with standoff binary attributes, and explicit markup languages such as HTML, in which the formatting codes are embedded directly in the text. In the early 90s humanists may have preferred a more expanded definition of markup because they needed to overcome their colleagues' resistance to its use, by arguing that it was only a variation on something they already used, such as punctuation and spaces, or word-processors. Since the historical discussion that follows is not bound by this constraint, this article will revert to the original definition of markup, like that provided by the OED, as *embedded* textual codes. References to 'markup' without further qualification also assume that markup is embedded in the text that it describes.

2 History of Markup

The purpose of tracing the history of markup is threefold: first, to investigate why it was adopted by humanists, and why they continue to use it. Secondly, it is to reveal the compromises made on the path to adopting generalized markup systems for cultural heritage texts. Finally, the history will establish the connection between markup and the print medium.

2.1 Markup's early history

Humanists were among the first computer users to devise markup systems. The first works to be converted to digital form were encoded as plain text (Froger, 1968; Gilbert, 1973). By the mid-70s, however, many early encoding schemes had been developed for specific humanities projects whose goal was the digitization of major authors. Ott (1979), for example, describes the insertion of 'codes' for index entries, line numbers, word and line deletion *etc.,* which were introduced into the text via special characters like '&', '!', '%', '+', '???', etc. Another early example of markup in the humanities is the COCOA scheme, which was part of the Oxford Concordance Program (Hockey and Martin, 1988). COCOA was very simple. Each tag was designed to mark a point of reference in the text, for example, a page break or a speaker name in an original document, such as <C Salerio> or <S 1>. Since the tags were unpaired, the values of references could freely overlap. However, this lack of structure meant that COCOA was not very expressive.

Each digitization project used its own markup scheme. Attempts were made to gather the results together, to produce a kind of digital library, for example by the Oxford Text Archive (Burnard, 1988) and Project Gutenberg in 1971 (Lebert, 2008). The former tried to preserve or add markup and was focussed on the needs of scholars, whereas the latter followed the earlier paradigm of plain text files. When there was markup in these early documents it rarely if ever had a formal language structure, that is, one based on hierarchies. Like COCOA, MECS (Huitfeldt, 1992), which was designed originally for the manuscripts of Ludwig Wittgenstein, is a survivor of this early markup model, where the tags are largely independent and allowed to overlap. Huitfeldt explains the reasoning behind it: 'I am not convinced that Wittgenstein's manuscripts are basically hierarchical structures. Potentially, for all I know, any feature may overlap with any other feature. Besides, I do not even know what the hierarchies should consist of, or whether the identification of such hierarchies would be particularly illuminating.' (1995, p. 239).

Even if these markup systems were not as fully developed as the industrial products that arose later, humanists had devised a form of markup that reflected the structure of the texts they knew much better than the IT experts who would succeed them. In spite of these advantages, these early markup schemes were all different. There was little agreement as to which features should be recorded, even less on how to encode them.

2.2 Markup languages

It was at this point in the history of markup, as far as humanists are concerned, that a parallel evolution in the print world promised to provide the desired means of standardization.

It is well known that modern generalized markup systems, such as SGML and XML, evolved from the 'presentational' markup contained in early digital documents intended for printing (Goldfarb, 1996, 1997). XML, whose specification was co-authored by a humanist, Michael Sperberg-McQueen (Bray *et al.*, 1998), is used as a metalanguage to define most of the actual markup languages in use today. XML is derived from SGML, which evolved from GML, developed in 1971 by Charles Goldfarb of IBM, and was one of the world's first 'generic' or 'generalized' markup languages, along with Tunnicliffe's GenCode. Generalized markup is supposed to separate the form of a document from its presentation, although in practice this is only possible to a limited degree (Raymond *et al.*, 1992, p. 16).

GML didn't completely die out with the advent of SGML and later XML; in fact a version of it lived on under a different name, as the most widely used form of digital text on the planet: HTML. The World Wide Web and HTML may have been invented in 1990 by Tim Berners Lee (Berners-Lee and Fischetti, 1999, p. 29), but it was based on an old version of SGML used at CERN (Palmer, 2000). Apart from the addition of the anchor <a> tag in HTML, most of the structure of HTML was taken ultimately from the GML 'Starter' tag set (GML, 1991). GML was used primarily for processing printed documents, but it could also be used for search and retrieval. Unlike SGML, however, you couldn't define your own language, so IBM provided a 'Starter Set', which was a kind of stylesheet called a profile. This connected the generalized tags in the document to the APFs or 'Application Processing Functions' that controlled the printer. The Starter Set profile had seven levels of heading from h0 to h6. HTML has h1 to h6. There are inline and long quotations called q and lq. HTML has q and blockquote. There are ordered and unordered lists called ol and ul, list items called li, and paragraphs divided by the p tag, just as in HTML. There are titles and tables and figures as in HTML. Structurally it is the same, even the tags are often identical. The 'Front Matter' is basically the same idea as the 'head' element in HTML. It has a title element and other bits of meta-text, just like the

HTML 'meta' tag. The following sample of GML markup illustrates this similarity with HTML (GML, 1991, App. 1.1).

```
:fn.
A starter set of GML tags is provided with
the Document Composition Facility to
allow the user to get going.
:efn.
:eol.
:p.A GML tag identifies the associated
text as a particular document element.
For example:
:ul.
:li.A book might have the major divisions:
:ul.
:li.front matter
:li.body
:li.back matter
:eul.
:p.Within those divisions we can have
paragraphs, examples, figures, list
items, and so on.
:li.A memo might have document elements of
addressee, date, sender, address, sub-
ject, and reference, as well as other
types similar to those of a book.
:eul.
:p.You might ask,:q.Why use generalized
markup instead of specific markup:eq.?
```

Despite assertions to the contrary, many of the 'generalized' tags in the Starter Set betray their origins as a printer control language. The title page tag (titlep), the footnotes, and the settings for running headers and footers, for example, imply that you are going to print something. There are even 'Process Specific Controls' to manage the printer directly (GML, 1991, Ch. 2.9). Both The SGML Handbook (Goldfarb, 1990, p. 147) and the latest XML specification (Bray *et al.*, 2008, Ch. 2.6) have similar sections on 'processing instructions', which are intended for the same purpose. The only real advance provided by GML was the deferral of formatting by means of a stylesheet; in structure the text was still an exact replica of a printed document.

This shows that not only HTML, but generalized markup as a whole, evolved from a printed format structure. The very existence of embedded tags (or commands), together with their 'attributes' (or arguments), are the relics of the print control statements from which they have evolved (Goldfarb, 1973). And the tree structure of formal languages added to GML and SGML was originally intended to support the procedural formatting of 'compound structures' in printed and electronic documents (Goldfarb, *ibid.*), not to describe texts born in an analogue medium. Hence, in spite of the generalization provided by its tags, XML can only express a document's 'logical' structure via a repertoire of permissible printed format structures.

2.3 The Text Encoding Initiative

The digital humanists, however, saw the development of SGML as an opportune invention. Its ability to define arbitrary markup languages with a deep structure (and therefore expressiveness) that were human-readable, appeared to answer their need for standardization. This would facilitate document interchange and the development of more standardized software. So in 1987 the Association for Computers and the Humanities convened a conference on text encoding practices and guidelines at Vassar College, Poughkeepsie, New York, to agree on a set of tags to encode cultural heritage texts, among others (Hockey, 1991).

The conference decided on a set of principles, the 'Poughkeepsie Principles', one of which was to 'define a metalanguage for the description of text-encoding schemes' (Sperberg-McQueen and Burnard, 1988). Soon after, a decision was taken to use the recently ratified ISO standard SGML, the 'Standard Generalized Markup Language' as this metalanguage (Goldfarb, 1990). The labour of drawing up the 'Guidelines', as they became known, was delegated to teams of scholars working in specialized areas. It took three years to produce the first draft, called 'P1' (TEI P1, 1990), and then four more years to produce the first printed Guidelines, 'P3' (Sperberg-McQueen and Burnard, 1994). When they were first published a whole volume of Computers and the Humanities (vol. 29, no. 1) was devoted to articles written by various scholars who had contributed to their formulation.

The subsequent versions P4 (Sperberg-McQueen and Burnard, 2002) and P5 (Burnard and Bauman,

2007) expanded the number of SGML structural 'elements'—i.e. pairs of matching 'tags' with intervening content—and introduced technological innovations: P4 replaced SGML with XML (Bray *et al.*, 1998) and P5 introduced various general updates (Bauman and Burnard, 2006). It now numbers 512 specialized elements, each of which may occur in a variety of contexts, and most of which have several attributes.

Although still a coding system of choice, humanists, at least, have begun to question it. The increasing size of the Guidelines, its perceived inadequacies, the strictness and growing complexity of the syntax, all seem to have alienated potential users (Fiormonte, 2003, p. 169; Shillingsburg, 2006, p. 115; D'Iorio, 2007, p. 8; Usdin, 2009). The authors of the TEI-Guidelines themselves admit that they have failed to achieve a close coupling with specific applications: 'This document does not, however, define—at least not explicitly—"sets of coding conventions suited for various applications," since consensus on suitable conventions for different applications proved elusive; this remains a goal for future work' (Sperberg-McQueen and Burnard, 2002, 1.3).

This goal, which first appeared in P3, then in P4, is dropped in the latest P5 Guidelines. Instead they now admit only that the Guidelines are 'a general-purpose encoding scheme which makes it possible to encode different views of text, possibly intended for different applications . . . no predefined encoding scheme can possibly serve all research purposes' (Burnard and Bauman, 2007, p. iv).

There thus appears to be a tension between the desire to create a general standard and the need to build practical applications with specific markup needs (Dipper, 2005, p. 40). The Guidelines are often ignored, e.g. by (D'Iorio, 2007) or adapted by their own editors (Burnard, 2007, Sect. 12), and they were in any case always envisaged to be 'unbounded' (Sperberg-McQueen, 1991, p. 36). This endless chasing after an indefinable standard is part of what McCarty seems to be referring to when he wrote: 'Eventually we realize that the perpetual impetus to construct a permanently elusive whole is the point.' (2005, p. 188).

2.4 The 'OHCO thesis'

In the early days of the TEI era the attractions of industrial tools associated with SGML—products like DynaText (Shillingsburg, 1996, p. 170, Robinson, 1997, p. 153)—led to the deficiencies of markup languages being overlooked or even suppressed. In 1988 Barnard *et al.*, in the first paper to deal with overlap in digital text, reassures the reader that 'SGML can successfully cope with the problem of maintaining multiple structural views' and that the solutions 'can be made practical' (p. 275).

A few years later, also in support of SGML, Renear, Mylonas and Durand presented a famous paper entitled 'Refining our Notion of What Text Really Is: The Problem of Overlapping Hierarchies' at Christ Church College, Oxford in 1992. Their 'thesis' stated that text is an 'Ordered Hierarchy of Content Objects', or 'OHCO' for short. In OHCO the 'ordering' comes from the fact that texts are linear: the objects of which they are composed succeed one another, and the objects themselves are hierarchical because structures like chapters, paragraphs, sentences and prose quotations 'nest inside one another like Chinese boxes'. The key argument that lies behind the thesis, however, appears to be flawed: 'if you treat texts as ordered hierarchies of content objects many practical advantages follow, but not otherwise. Therefore texts are ordered hierarchies of content objects.'

All this says about text is that hierarchical structures are easily processed by computer. This follows in any case from the hierarchical nature of computable formal languages, but is also implied by Church's model of computation from 1936: if all calculation can be modelled on recursion it follows that hierarchically organized data will be readily computable.

The authors of OHCO try to preserve the thesis in light of numerous counter-examples. They posit 'OHCO-2', which says that text is sometimes not a single hierarchy but instead consists of a number of separate 'analytical perspectives', each of which is strictly hierarchical. However, they realize that even multiple perspectives often contain overlapping structures, for example, the division of a play into verses and speakers—often one verse is divided between two (Barnard *et al.*, 1988, p. 266).

'OHCO-3' attempts to repair the damage: 'objects may overlap in a perspective, but if they do then they belong to different sub-perspectives of that perspective'. Even this final refinement admits of many exceptions, including variant readings. However in spite of this they conclude that the fundamental view of the OHCO thesis still stands: text is composed of 'meaning related features' that are 'often hierarchical'.

Just a few years later the authors of the thesis retracted it: 'we now know that the breaking of strict hierarchies is the rule rather than the exception.' (Durand *et al.*, 1996). Renear's retraction, though, is less categorical: 'Whatever may be said for hierarchy as a tendency, it does not seem to be, even in its perspective-contingent form, an *essential* aspect of textual structure.' (Renear, 1997, p. 121). Likewise Buzzetti, in his detailed and more recent discussion of digital text, remarks: 'An OHCO structure is not a model of the text, but a possible model of its expression. ... not only must we agree that "the same document conforms to several overlapping structures," each of which is "strict hierarchical," ... but we must also recognize that textual structures are not usually of this type.' (2002, pp. 71–72). Similarly Huitfeldt asks: 'why on earth should texts by all means be hierarchies?' (1995, p. 240).

Hierarchy *is* detectable in certain texts, e.g. plays and printed novels, but in other cases, such as drafts of modern poetry, it is barely discernible. In documents not authored in the digital medium these fragments of hierarchical structure are rarely expressed rigorously enough throughout the document to enable precise capture by generalized markup systems, and to impose such a structure upon them inevitably results in misrepresentation.

2.5 The 'digital incunable'

Marshall McLuhan drew an analogy for the transition from print to electronic text from the earlier transition of the manuscript codex to printed book (1962, p. 153). The early printed books, called 'incunables' or (books) 'in swaddling clothes', were created in mimicry of the medium they replaced. Having no other model on which to base their designs, the early printers used the form of the manuscript: its characters, its layout, its marginal notes and coloured letters, so that their creations resembled a well-written manuscript. It took until the sixteenth century for the development of graduated types, running headers, footnotes, tables of contents, title pages and other devices to register the 'victory of the punch-cutter over the scribe' (Eisenstein, 1983, p. 21). McLuhan generalizes from this example, among others, to establish his principle that: 'Every technology contrived and outered by man has the power to numb human awareness during the period of its first interiorization.' (1962, p. 153).

The phrase 'digital incunable' appears to have been coined by Dahlström (2000): 'to date, however, the digital SEs [scholarly editions] are very much constructed as though they were print based, trying to imitate the architecture and the subsequent status of print editions. They are, in other words, digital incunables.' Robinson (2003) agrees: 'Almost all we have done, in the first ten years of electronic scholarly editions, is find ways of mimicking on screen elements long present in print and manuscript.'

The idea that modern digital texts in the humanities are really digital incunables has been widely discussed and is generally accepted (Ross 1996; Fiormonte, 2003; Smith, 2004, p. 26; Burnard *et al.*, 2006; Crane *et al.*, 2006; Shillingsburg, 2006, pp. 80–1). If we want to embrace the new medium we have to find ways to purge our digital texts of those characteristics that are adaptations from the print medium. And those very adaptations can only reside in one place. Not in the representation of letters on a page by Unicode characters. For they represent letters from any medium, not only print. The only possible repository of digital incunabularity can be the markup.

2.6 Conclusions

Significant aspects of the print model of text have been copied into our digital representations of cultural heritage texts. Unsurprisingly, markup in these texts suffers from representational difficulties precisely because it is not purely digital.

Digital humanists chose generalized markup as an off-the-shelf industrial tool, not as something that was built according to their specifications. The subsequent attempt to justify the choice with the OHCO thesis failed.

Digital humanists have lived in virtual symbiosis with markup since its invention. They may thus find it discomforting to accept that markup may be inadequate for their purposes, or that they should have to rewrite tools developed over years for processing it.

3. The Technical Deficiencies of Markup for Cultural Heritage Texts

Are the deficiencies of markup sufficient to warrant its replacement or wholesale remoulding to the requirements of humanists? Or should it be conceded that, despite its being an industrial tool, generalized markup is a good enough solution and too useful for humanists to discard? This section explores the tensions and difficulties arising from the adoption of any form of embedded markup, and establishes what are the limitations for textual phenomena that can be represented, and identifies those that essentially cannot.

There are actually several problems with using markup in cultural heritage texts, among which the overlap problem is only the best known. However, the others are nearly as serious, and very little has been written about them. The problems that will be investigated here are:

(1) The overlap problem.
(2) The embedding of potentially obsolescent technological and subjective information into the text.
(3) The manual recording of information via markup that can be automatically computed.
(4) The fact that markup is a textual command language, not a modern graphical user interface.

3.1 Overlap

There are two types of overlap in cultural heritage texts:

(1) Textual variation, and
(2) Individual elements that overlap parts of the document hierarchy

Textual variation was categorized by Renear *et al.* (1993) as one of the forms of overlap to break OHCO-3. Variation involves overlap of the entire text, the content and the markup taken together, not merely in the markup. This is the most serious form of overlap that actually subsumes all other forms (Schmidt and Colomb, 2009).

Examples of overlap between one hierarchy and an individual element include speeches and lines in a play, where one line, normally part of a speech, may be split between two speeches. DeRose (2004) also gives the example of a triple-nested quotation that spans verses in a biblical text. Another case so familiar as to be forgotten is of a page that spans from the middle of one paragraph to the middle of another. We are so used to hacking this as a 'page break' that we forget that it is also a case of overlap.

But what about 'overlapping hierarchies'? As argued in Schmidt and Colomb (2009, p. 498) overlapping hierarchies ought properly to apply to actual hierarchies of elements that partly or completely overlap. This is a problem mostly for corpus linguists, who must resolve the overlap between separate markup analyses generated by tagging tools for natural language. Apart from instances where technologists have proposed overlapping hierarchies as a solution to the overlap problem (Dekhtyar *et al.*, 2006; Chatti *et al.*, 2007; Di Iorio *et al.*, 2009b; Portier and Calabretto, 2009) humanists themselves seem content with one structural encoding of the text. The author's own experience, combined with what Renear (1997), Durand *et al.* (1996), Huitfeldt (1995), and Buzzetti (2002) appear to be saying, suggests that there is no strong tendency towards the formation of even one hierarchy in cultural heritage texts, let alone several.

3.1.1 *The technical cause of overlap*

One possible solution that occurred to Barnard *et al.* (1988) was to allow the start and end-tags of a markup language to overlap. But is this technically feasible? And do markup languages always parse into a tree structure? To answer these questions it will be necessary to briefly investigate the theory of formal languages.

In the 1950s linguists like Noam Chomsky worked on the theory of phrase structure grammars, as a model of natural language (1957). Informally, a phrase structure grammar is a set of grammatical substitution rules such as those in Fig. 1, which describe the syntactical structure of a simple sentence. Each rule is composed of symbols, which represent groups of characters and other symbols, and terminals, which represent tokens of the string being represented. Parsing is the process by which the source text is progressively reduced to a single symbol, usually called S (the start symbol) by the application of the rules. By placing constraints on what kinds of rules are allowed in the grammar, different classes of language could be specified.

Figure 2 shows a modified form of the Chomsky Hierarchy of computer-recognizable languages. At the bottom are the regular languages, the kind that are used in regular expression matching tools such as in grep or perl. Containing these are the context-free languages, of which the SGML and XML-defined languages are examples, and also most modern programming languages (Kilpelainen, 1999). The third type are the context-sensitive languages. These have rules whose application depends on the context. For example the rule AB → Ab substitutes b for B only when an instance of b is preceded by the symbol A.

The minimum standard for a markup language is decidability. A decidable language is one for which there exists a computer program that always halts on any input with a yes or no answer for membership in the language. All context-sensitive languages are decidable. The recursively enumerable languages in the outer ring of Fig. 2 are not necessarily decidable. They may fail to halt on an invalid input but will always terminate on correct input. This is not useful for humanists because the parsing program may just loop forever trying to tell you that your input is invalid.

It is possible to specify a decidable language without using grammars at all, for example, by writing a program. Without a grammar, however, it is difficult to verify that the program terminates on all inputs, that it correctly recognizes all valid inputs and rejects all invalid ones, that the data structures it creates are all those desired, and contrariwise that all

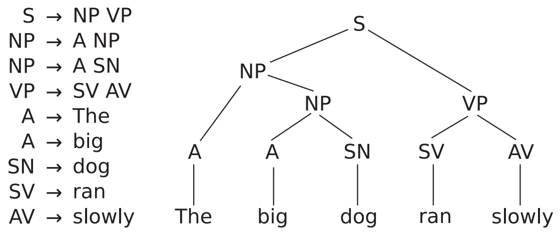

Fig. 1 A simple phrase-structure grammar

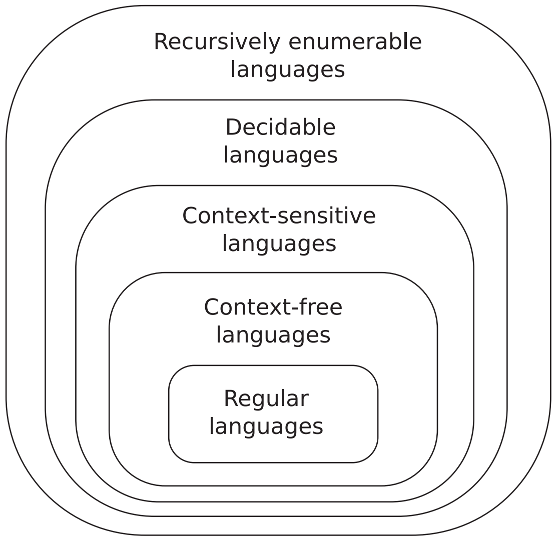

Fig. 2 The Chomsky Hierarchy

desired data structures can be specified in the language.

It has been known since the 1950s that *all* computer-recognizable languages can be specified by a grammar (Davis, 1958, p. 88–100; Chomsky, 1959, p. 143; Hopcroft and Ullman, 1969, p. 111–3). And at least all context-sensitive grammars parse naturally into a tree structure. Although it can't be ruled out, it thus seems unlikely that any language allowing overlapping elements can be decidable.

3.1.2 *Conclusion*

The origin of the overlap problem is simply that humanists are trying to represent what they all agree are non-hierarchical structures using a container whose primary structure is a tree.

This seems to apply to all markup languages that are embedded in the text, not only those based on XML.

3.1.3 *Textual variation*

Textual variation is the main source of overlap in cultural heritage texts. Variation is often introduced in texts that are corrected, usually by the author. Or works may exist in several versions, each of which is understood as a variant expression of the same text. Often in modern manuscripts the work is made up of separate *edited* drafts, and is thus a combination of these two forms of variation. In all these cases the relations between parts of the text that are the same or different imply a kind of data structure that is very difficult to represent accurately via markup.

This implied data structure can be specified in a diagram—the structure that the markup will have to represent (Fig. 3). In a short example taken from the Sibylline Gospel (Schmidt *et al.*, 2008) interrelated segments of the text are underlined and marked with the same numbers. This example has been chosen, not because it is particularly difficult, but because it exhibits in a small space all four editing operations: insertion, deletion, substitution, and transposition:

There is a clear connection here between the 1-variants 'sumpno suscepto', 'somno suscepto' and 'sompno suscepto', as also between the 4-variants 'deinde' and 'tunc'. In addition, one can see the simultaneous transposition *and* substitution of the 5, 6 variants 'mortem sortis' for 'sortem mortis'. Many of the other elements are transposed freely between the versions. The implied structure between the various elements here is not in the least hierarchical, and they could not be recorded in markup without considerable repetition of the overlapping elements.

This example is by no means unique. Many medieval works exhibit similar levels of variation (Bédier, 1970, p. 2 Cerquiglini, 1989; Robinson, 1998, p. 274; O'Donnell, 2005), as do modern manuscripts (Nedo, 1993; Vanhoutte, 2006). Imagine how much more complex the problem would become if, as is often the case, more than three versions of the text exist.

3.1.3.1 *Proposed representations of textual variation.*

For reasons of space the following section applies the two main techniques for representing textual variation in markup to the short example text given above. Some other methods, which don't apply to the example text, will also be described.

3.1.3.2 *Double Endpoint Attachment.*

This method, described in the TEI Guidelines (Burnard and Bauman, 2007, 12.2.2), is a translation to the digital domain of the old critical apparatus method found in printed editions. A single version is required as the 'base text' to which the start and endpoints of each variant *group* are attached. In this case no single base text is really possible; the versions are simply too different. However, for illustrative purposes version 1 can be taken as the base text. The only practical way to represent the three versions using this method is to specify the entire variant section between 'Et' and 'ab inferis regressus ad lucem veniet.' as three long variants:

```
Et <anchor xml:id="sg.1"/>sumpno sus-
cepto tribus diebus morte morietur et
deinde<anchor xml:id="sg.2"/> ab inferis
regressus ad lucem veniet.
<app from="#sg.1" to="#sg.2"><lem
wit="A2">sumpno suscepto tribus diebus
morte morietur et deinde</lem>
```

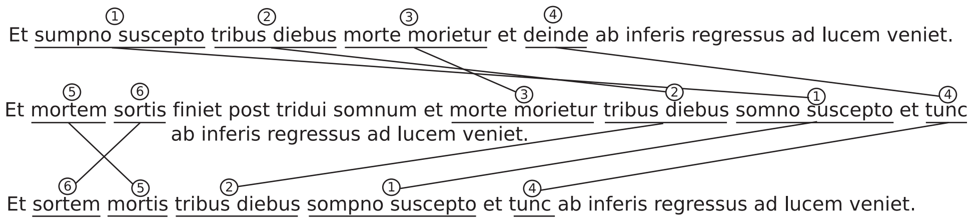
Et sumpno suscepto tribus diebus morte morietur et deinde ab inferis regressus ad lucem veniet.
Et mortem sortis finiet post tridui somnum et morte morietur tribus diebus somno suscepto et tunc ab inferis regressus ad lucem veniet.
Et sortem mortis tribus diebus sompno suscepto et tunc ab inferis regressus ad lucem veniet.
1
2
3
4
5
6

Fig. 3 Structure of the Sibylline Gospel Example

```
<rdg wit="C1,L1ac,L1pc,L1sscr,L2,L3,L5,
L6,L7,L8,Me,Vo">mortem sortis finiet
port tridui somnum et morte morietur
tribus diebus somno suscepto et
tunc </rdg><rdg wit="C4">sortem mortis
tribus diebus sompno suscepto et tunc
</rdg></app>
```

3.1.3.3 *Disadvantages of Double Endpoint Attachment.* Because this method only describes variants, not transpositions, it is very difficult to record very much of the structural detail. If 'sumpno suscepto', 'somno suscepto' and 'sompno suscepto' had been recorded as variants, this would mask the fact that 'sumpno suscepto' is actually transposed in the first version. Similarly, how can one record that 'tribus diebus' is transposed around 'sumpno suscepto' between versions A2 and the rest? Or that 'morte morietur' is transposed around the whole thing?

There is a lot of redundancy in the text of the three versions. Each copied word that is supposed to be identical has to be maintained as such. Every expenditure of effort on the text is then multiplied by the presence of all the copies. Incorrect or inconsistent results are likely to occur when searching an XML document containing such copies.

Variants may overlap with the base text but not with any other version. Trying to describe spans using 'from' and 'to' attributes attached to other versions would inevitably cut across parts of the <app><rdg>...</rdg>...</app> structure itself, and break the well-formedness of the markup, as well as drag in bits of other variants.

3.1.3.4 *Parallel Segmentation.* The second main method, called 'Parallel Segmentation' in the Guidelines (12.2.3) is also used in the Vienna Wittgenstein Edition (Nedo, 1993) and in the form of 'poly-element codes' in MECS, which was originally used for recording multiple variants (Huitfeldt, 1992). Wherever the text diverges into a number of versions these can be recorded as a set of variants (in the TEI Guidelines indicated by an <app>...</app> element). Each alternative in the set can also contain another set of variants. In other words, it is recursive. At first sight this seems to be a big advantage. For example, it allows one to avoid various repetitions that occurred in the Double Endpoint Attachment method:

```
Et <app><rdg wit="A2">sumpno suscepto</
rdg>
<rdg wit="C1,L1ac,L1pc,L1sscr,L2,L3,L5,
L6,L7,L8,Me,Vo">mortem sortis finiet
post tridui somnum et morte morietur
</rdg><rdg wit="C4">sortem mortis
</rdg></app>
tribus diebus
<app><rdg wit="A2">morte morietur et
deinde</rdg>
<rdg wit="C1,L1ac,L1pc,L1sscr,L2,L3,L5,
L6,L7,L8,Me,Vo,C4">
<app><rdg
wit="C1,L1ac,L1pc,L1sscr,L2,L3,L5,L6,
L7,L8,Me,Vo">somno </rdg><rdg wit="C4">
sompno</rdg></app> suscepto et tunc<rdg>
</app>
ab inferis regressus ad lucem veniet.
```

3.1.3.5 *Disadvantages of Parallel Segmentation.* At the cost of making the markup harder to read, this encoding avoids repetition of 'tribus diebus' and 'suscepto et tunc'. Forcing the text to align on 'tribus diebus' avoids some repetition but also falsely encodes 'morte morietur et deinde' as a variant of 'somno/sompno suscepto et tunc', when in fact both variants include transpositions in other versions on either side of 'tribus diebus'.

The ability to nest groups of variants, although used here for the trivial variant 'somno'/'sompno', turns out to be virtually useless: it still cannot avoid repetition of text between versions. In the above example 'suscepto' and 'morte morietur' are both copied. As explained in the TEI Guidelines, if there are more than two versions such copying is inevitable with this technique, and increases with the number of versions (12.2.3). Accuracy also suffers, since the copies interfere with the placement of new variants.

3.1.3.6 *Interlinking.* Vetter and McDonald (2003) try various techniques for representing variation by interlinking segments of text, specifically by adding <seg> or <anchor> elements to denote small sections of text that can be connected to other parts by means of <alt> elements or via attributes such as id and corresp.

3.1.3.7 *Disadvantages of interlinking.*

None of these methods are suitable in a case like the Sibylline Gospel, where it is necessary to record the list of witnesses for each version.

The <anchor>, <seg> and <alt> elements do not record any structural features of the text. They are added purely to provide a means of linking fragments of text via markup.

As Vetter and McDonald (2003, p. 161) themselves note, such methods are at best an unwieldy workaround that is computationally inefficient.

3.1.3.8 *Layering.*

The practice of dividing the corrections to a manuscript into layers is described for example by Gabler *et al.* (1984). Zapf (2006), D'Iorio (2007) and Pierazzo (2007) describe the adaptation of this technique to markup. The method is reminiscent of CONCUR and shares its disadvantages (Goldfarb, 1990, p. 304). In HNML (HyperNietzsche Markup Language) any element may be assigned a lay="N" attribute, where 'N' is the number of the layer. Text assigned to a layer appears in all layers from that point on, i.e. layer 1 text also appears in layers 2, 3 *etc.* This default behaviour can be reversed by enclosing the text in a <str> element with a layer attribute. In this case the text *disappears* from all layers thereafter and including the one assigned to the <str> element (Zapf, 2006, p. 16).

3.1.3.9 *Disadvantages of layering.*

As in CONCUR, there is no way to specify arbitrary combinations of versions. Layers can't represent transpositions or non-hierarchical structures because they are based on markup. Because of the excessive complexity that would result, this technique is generally limited to representing layers of corrections in individual manuscripts, rather than for interrelating variants across physical copies or drafts of the one text.

3.1.3.10 *Forms of variation not covered by markup.*

If markup is part of the text, if it describes real features such as paragraph breaks, and formatting information, then what happens when that very information varies? Smith (1999) makes the point that you really need another level of markup above the ordinary markup to describe such phenomena. Examples include a cancelled underlining. Pierazzo (2007, p. 151) attempts to record this in markup:

```
<del type="underline deletion" time="4">
<hi rend="underline" time="3">si</hi>
</del>
```

In spite of the customized attribute type="underline deletion", the <del> element still necessarily encloses the word 'si', even though this is not deleted.

Another case is joining two paragraphs together—a frequent occurrence in modern manuscripts. What is varying here is the string '</p><p>', which is not permitted as the content of an element in XML. Such variation can't be specified without copying the entire content of the two paragraphs as one alternative, and the other merged paragraph as the other, if well-formedness is to be preserved.

In principle, variation of any feature of the text recorded by markup will run into the same problem. Markup can only describe text not markup itself.

3.1.4 *Type 2 overlap*

This article does not propose a separate solution to type 2 overlap (i.e. individual elements that overlap the document hierarchy, described in section 3.1 above) other than the encoding of alternative hierarchies as versions, as proposed in Schmidt and Colomb (2009, p. 499). Type 2 overlap has been extensively treated in the literature, e.g. by Barnard *et al.* (1988), DeRose (2004) and Huitfeldt (1995). Almost all of the solutions proposed so far, however, take embedded markup as their starting point. As argued above, the intrinsic weaknesses of that form of data representation make a comprehensive solution to type 2 overlap difficult.

It is, however, not hard to imagine what form such a solution might take. The advantage of markup's strict hierarchy of elements and their relations of containment has always been that it allowed sets of properties to be asserted over ranges of text: for example, a word might be part of a line, a speech, a scene, an act and a play at the same time. By collapsing the hierarchy down to a simple set of overlapping properties, and removing the markup from the text, the type 2 overlap problem can be eliminated. This is, more or less, the 'extended string model' of Thaller (1996, 2006) and Neumann (2006), but there is also some resemblance to the standoff annotation systems of EARMARK (Di Iorio *et al.* 2009b) and LORE (Gerber and Hunter, 2009). In these applications annotations are held on an external server and are never embedded in the text, thus allowing multiple interpretations of the same document to be maintained, without disturbing the correctness of the annotated document itself.

However, with current technology it will still not be possible to remove markup from the text entirely. Mathematical formulae and inline graphics are both examples of content that cannot be represented satisfactorily via plain text. However, since the markup in these cases represents actual content, instead of describing other content, problems of overlap should not arise.

3.1.5 *Conclusions*

Overlap is a serious problem in the encoding of cultural heritage texts. The detail of these overlapping structures can only be 'approximated' in markup (Neyt, 2006), and this approximation gets worse the more detail there is to record. The problem does not lie in the TEI Guidelines, nor even with XML. The technical limitations of embedded markup itself are to blame.

3.2 Instability of XML and XML-encoded texts

Even though all digital technologies are subject to change, XML/SGML has been relatively stable for the past 20 years. On the other hand, XML is an industrial tool, whose continued existence depends on industry, not on humanists. And for some years now industry has been complaining about the efficiency of text XML. In modern Service-Oriented Architectures, for example, the parsing of XML messages is a major drain on performance (Davis and Parashar, 2002). XML's verbosity limits human comprehension in ontology languages (Horridge *et al.*, 2006), and seriously impacts on database performance (Nicola and John, 2003). For some of these applications a binary format would be much faster. The ISO have recently ratified the Fast Infoset as a binary format for XML: (ISO-FI, 2007), which is based on the old ASN.1 standard. But there are other competing technologies, such as the W3C's EXI specification (Schneider and Kamiya, 2008). If the attractions of these approaches lead to their supplanting text XML for many applications, then the text form could conceivably fall into disuse.

3.2.1 *Use of customised XML schemas*

But even if it is conceded that XML is a stable enough platform for the archiving of cultural heritage texts, there is another, more serious, problem: the customization of XML schemas. This is common practice in the humanities and is in fact actively encouraged (Burnard and Bauman, 2007, 23.2). And for every customized XML schema there is customized software to utilize it, and *that* is subject to rapid change: 'the software product is embedded in a cultural matrix of applications, users, laws, and machine vehicles. These all change continually, and their changes inexorably force change upon the software product.' (Brooks, 1987). When research projects terminate, their custom software quickly becomes unusable and the archived documents that depend upon it may become inaccessible. And customized XML encodings are not as convertible into new forms as is generally supposed. Marked up transcriptions of cultural heritage texts inevitably contain many hacks to overcome the deficiencies of XML, as described above: duplicated text between variants, customized elements and attributes, extra markup that reflects no features of the original documents, tags and attributes added to record some aspect of the intended output, e.g. concordancing, collation, printing or online presentation. It is almost impossible to keep such information out of what is supposed to be purely generalized markup. For example, modern HTML is full of formatting information. To remove this specific information later requires a precise understanding of the original format or the text may become damaged.

3.2.2 *Markup as interpretation*

Humanists seem united in their view that all markup is interpretation (e.g. Sperberg-McQueen, 1991, p. 35; Fiormonte, 2003, p. 163; Eggert, 2005; McCarty, 2005, p. 27;). If that is so, then why is it embedded in the text? The text is the thing being interpreted, not the markup. Once embedded, markup obscures and biases what a new scholar, who didn't carry out the initial markup, can see. The argument that markup cannot be avoided because carriage-returns, spaces and punctuation are markup (Section 1.2, above) is a purely theoretical stance that shouldn't be used to justify the embedding of any amount or any kind of interpretative markup into the text.

So the big question becomes, is it appropriate to embed the technology and interpretations of today into texts that will be archived for the future? If not, then archives of cultural heritage texts should be in a plain text format. The ASCII standard, which became part of ISO-LATIN-1, which in turn became part of Unicode, is one of the most stable forms of digital data known. If one can predict anything about the software industry it is that texts of the future will be in Unicode or something compatible with it.

3.3 Redundancy of markup to interconnect versions

Another drawback with markup is the redundancy of having to record variation manually. The global differences between a set of versions of the one work are just a set of insertions, deletions, substitutions and transpositions. These can be computed automatically for cultural heritage texts, just as they have been for biological texts (sequences of amino acids or nucleotides) for the past 40 years (Bourdaillet, 2007; Schmidt, 2009a; Di Iorio *et al.*, 2009a).

In speaking of the supposed advantages of standard XML tools what is often forgotten is the human cost of training people to use markup, of getting them to encode it and check the marked up texts against originals that don't contain any markup. Reducing that cost by removing most of the markup and handing its functionality over to the computer will save time and reduce complexity. Although computing the differences between the versions of a work is not yet perfect, and never will be, it is still far more accurate than manual coding because of limitations in the expressiveness of markup. And even though a scholar may know precisely how the text varies, finding out all that detail manually in complex cases is practically impossible, or takes too long.

3.4 Markup as a textual command language

The revolution in human-computer interaction that occurred in the 1960s and 70s, based on key inventions made at Xerox PARC and elsewhere, seems to have passed largely unnoticed by digital humanists. The invention by Engelbart of the mouse in 1968 (Hiltzik, 2000, p. 65), and by Ingalls of the BitBlt in 1974, that allowed blocks of screen image to be rapidly copied, thus creating the illusion of movement (*ibid.*, p. 226), are two important foundation stones of the modern Graphical User Interface or GUI. Before that, humans interacted with the machine via a textual command language, on a 24 × 80 character screen, and a shell that accepted commands with a certain syntax. And markup, like the commandline interface, is a textual command system.

> This style of work [an arcane command syntax] may have been acceptable in the past, but user communities and their expectations are changing. While there are still millions of users of command languages, the development of new command languages has slowed dramatically, due to the emergence of direct-manipulation and menu-selection interfaces. (Shneiderman, 1998, p. 279)
>
> The command-line interface forces an even more expensive excise budget on the user: He must first memorize the commands... The excise of the command-line interface becomes smaller [than the GUI] only after the user has invested significant time and effort learning it. (Cooper and Reimann, 2003, p. 136)

The main advantage of XML or SGML over binary formats is their human readability. But even here there is a conflict between human and machine readability:

computers decode—humans read . . . Nothing kills readability like having to pick your way amongst a forest of tags and fancy character encodings. . . . Computers most easily recognize data and markup when examining data on a byte-by-byte basis—humans most easily recognize data in context. . . . it's easy to see that one might want quite different markup languages for computer use and for human use. (Wilmott, 2002)

Wilmott doesn't consider the possibility of removing markup altogether, and only looks at ways to minimize its effects. But he doesn't contest that markup always has a negative impact on human readability.

As in the case of complex command-line interfaces, complex markup systems divide the user community into two groups. The first consists of a small number of expert users who have made the effort to learn the syntax and the tools associated with it. The second, and far more numerous group, are the casual users who are put off by the complexity of the system and only want something simple to use. It is probably unrealistic to expect that a significant percentage of humanists will ever become power users of a 'very extensive encoding language . . . intended to support very complex encoding of very complex documents' (Burnard and Bauman, 2007, FAQ):

Command languages and lengthier query or programming languages are the domain of expert frequent users, who often derive great satisfaction from mastering a complex set of semantics and syntax. (Shneidermann, 1998, p. 73)

Then came the visible user interface, with its rich use of graphics, consistent behaviour, visually apparent structure, and clear communication. It arose from a culture . . . dedicated to the single task of bringing the power of the computer to people everywhere, instead of concentrating it among a select priesthood . . . Non-computer professionals for the first time gained a sense of competence and control over the computer. (Tognazzini, 1992, p. xiv–xv)

The 'unboundedness' of any comprehensive markup system for cultural heritage texts (Sperberg-McQueen, 1991, p. 36) inevitably leads to a level of complexity that concentrates the skill needed to wield it in the hands of a select few, while depriving the majority of humanists of their share of control. For them, the future of their interaction with the text would appear to lie in the GUI, not in markup.

4 Multi-Version Documents

Generalized markup systems have undeniably been a useful tool for humanists for the past 20 years, and to criticize them without describing a viable alternative would be unjustifiable. This section summarizes a replacement for much of the embedded markup in cultural heritage texts that is already being used in practical projects.

The deficiencies of markup as described in section 3 above are mostly caused by the inadequacy of a software system originally designed to represent printed documents, and later newly-authored electronic documents. When applied to already existing texts on paper, or other physical writing materials, the structure of versions, with their overlapping interconnections, led to a break-down of the OHCO model. The solution, then, can only lie in the creation of a new model of text based on the natural structure of these physically written or printed documents.

The Multi-Version Document or MVD model represents all the versions of a work, whether they arise from corrections to a text or from the copying of one original text into several variant versions, or some combination of the two, as four atomic operations: insertion, deletion, substitution, and transposition (Schmidt and Colomb, 2009).

An MVD can be represented as a directed graph, with one start node and one end-node, as shown in Fig. 4a. Alternatively it can be serialized as a list of paired values, each consisting of a fragment of text and a set of versions to which that fragment belongs

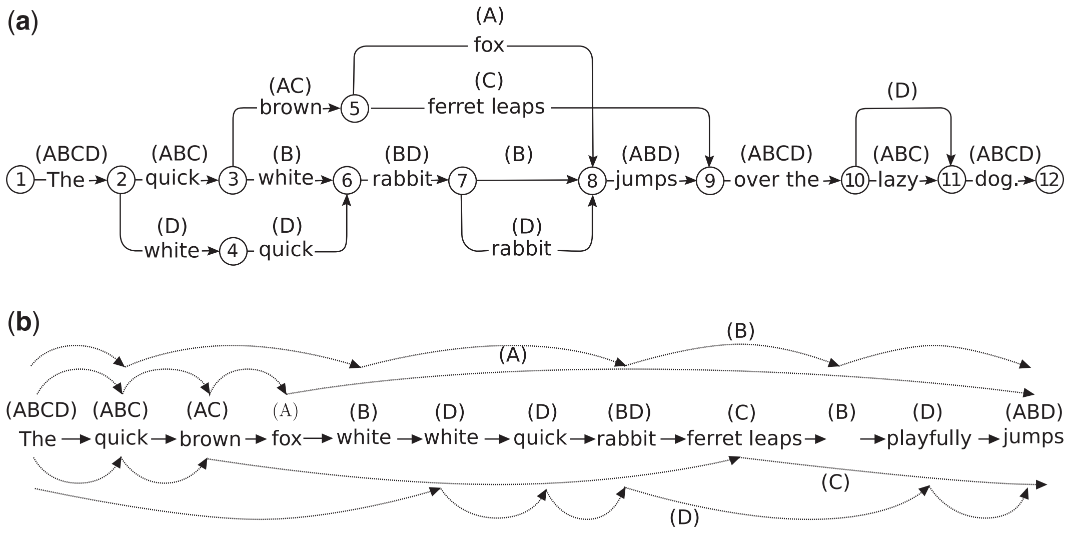

Fig. 4 Multi-Version Document graph and list structures

(Fig. 4b). As the number of versions increases, the number of fragments increases, their size decreases, and the size of their version-sets increases. This provides a good scalability as it trades off complexity for size, something that modern computers are very good at handling. By following a path from the start-node to the end-node any version can be recovered. When reading the list form of the graph, fragments not belonging to the desired version are merely skipped over (Fig. 4b).

The key features of MVDs as far as the humanist is concerned are:

(1) The insertions, deletions, variants and transpositions in a set of versions are computed automatically instead of being manually entered.
(2) An MVD knows nothing about the content format of an individual version. An MVD may be used in conjunction with generalized markup or any other format such as plain text.
(3) An MVD is not a collection of files. It stores only the differences between all the versions of a work as one digital entity, and interrelates them.
(4) Because it encodes all the complex overlapping structures of a set of versions, the markup of an individual version can be much simpler.
(5) An MVD is the format of an application, not a standard.
(6) An MVD has a zero footprint, i.e. it does not require the text of the versions to be changed in any way. You can always get out the text in exactly the same form as it was put in.
(7) An MVD allows the user to efficiently compare, search, edit and list versions.
(8) Type 2 overlap can be represented by markup layers encoded as sub-versions. Alternatively another content-technology that allows type 2 overlap could be used in place of generalized markup.

The earlier example of three versions of one sentence from the Sibylline Gospel can be represented by the following variant graph (Fig. 5).

The structure and contents of this graph were generated *automatically* from the three versions. The nmerge program that produced it (Schmidt, 2009b) detected two transpositions ‘no suscepto’ and ‘tribus diebus’, as well as merging the text in common. The output was in the form of a list of pairs, as in Fig. 4b, which was then manually converted into the graph of Fig. 5. The dotted lines indicate implied links between fragments of

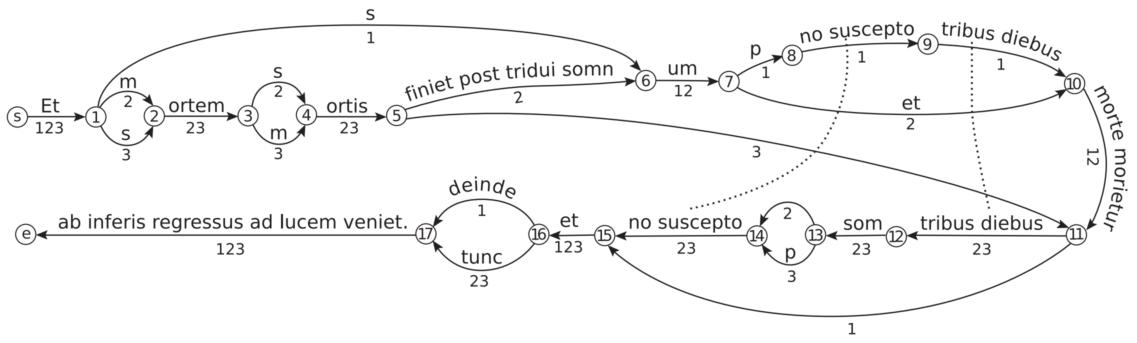

Fig. 5 Variant graph of the Sibylline Gospel example

transposed text. This level of detail, as explained above, could not be represented by *manually* encoding the three versions using markup.

4.1 Standards versus functionality

Standards should not be followed if they lead to mutilation of the data. Although we live in a world of markup, this does not force us to encode every kind of text in it. This article has argued that, because of its flaws, embedded markup should not be used as the core representation of cultural heritage texts. This doesn't necessarily mean that markup need be abandoned entirely. Rather, there can be an Application Programming Interface between the markup universe, which represents text to the user, and interacts with the core text via a prescribed set of commands, as shown in Fig. 6.

All that matters about the core text is that its representation is adequate. Since the current realization of the MVD model still uses light XML to represent the text of individual versions, it can leverage existing tools such as XSLT or XQuery to transform or investigate the text. It can archive the MVD representation as a set of separate files if this is thought to be more compatible. But this also means that the individual versions may still be subject to type 2 overlap (if markup layers are not encoded as versions), and hence it may be desirable to replace the light markup with another content technology that would allow overlap.

However, development of a replacement for the light markup currently in use by the MVD solution

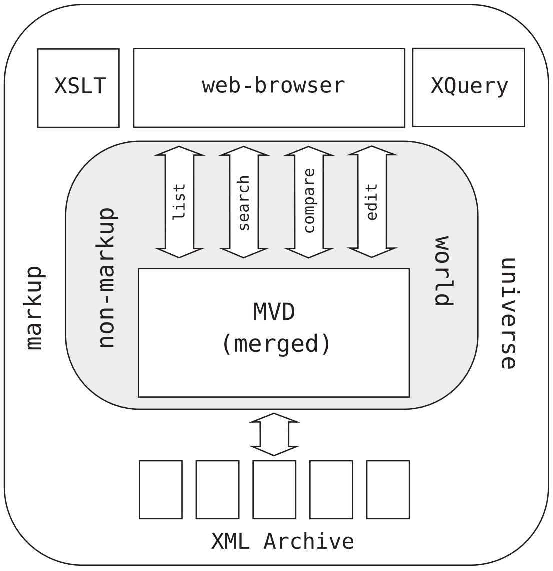

Fig. 6 Interaction between an MVD and the markup universe

is beyond the scope of this article, and is left for future work.

5 Conclusion

The adaptation since the late 1980s of generalized markup systems such as SGML and XML, originally designed for technical documentation, to the encoding of historical manuscripts and printed books, led to the discovery of a number of serious deficiencies, such as the overlap problem. These deficiencies have generally been ignored or underestimated because so far humanists have always supported generalized markup and have even taken an active role in its development. Rather than accept these deficiencies out of a desire for standardization, we can instead choose to store our texts in a non-markup environment that can represent our cultural heritage more accurately. Texts can then be exported from, or imported to that representation, without breaking their dependency on external markup-based forms. One working method that achieves this is the Multi-Version-Document system, whose components are freely available for download and evaluation (Schmidt, 2009b). Although these are early versions, they allow a simplification of the process for the digital recording and editing of historical documents, an increase in accuracy and automation, and promise in the future to broaden the user base to include non-technical people in the processes of text refinement and fruition. This represents a departure from standard practice, but it is a necessary step if these benefits are to be realized.

Funding

This work was supported by the Italian project CoOPERARE (Content Organization, Propagation, Evaluation and Reuse through Active Repositories), Project of Research in the National Interest (PRIN), 2008-09, whose Rome research unit is directed by Domenico Fiormonte.

References

Barnard, D., Hayter, R., Karababa, M., Logan, G., and McFadden, J. (1988). SGML-Based Markup for Literary Texts: Two Problems and Some Solutions. *Computers and the Humanities*, **22**: 265–276.

Bart, P.R. (2006). Experimental markup in a TEI-conformant setting. *Digital Medievalist*, 2.1 http://www.digitalmedievalist.org/article.cfm?RecID=10 (accessed 13 December 2009).

Bauman, S. and Burnard, L. (2006). TEI P5: What's in It for Me. In Gallet-Blanchard, L. (ed.), *Digital Humanities 2006 Conference Abstracts.* CATI, Université Paris-Sorbonne, 5–9 July, pp. 12–13.

Bédier, J. (1970). *La tradition manuscrite du Lai de L'Ombre: réflexions sur l'art d'éditer les anciens textes*. Paris: Champion.

Berners-Lee, T. and Fischetti, M. (1999). *Weaving the Web*. San Francisco: Harper-Collins.

Bourdaillet, J. (2007). *Alignment textuel monolingue avec recherche de déplacements: algorithmique pour la critique génétique*, PhD thesis, Université Paris 6 Pierre et Marie Curie.

Bray, T., Paoli, J., and Sperberg-McQueen, M. (1998). *Extensible Markup Language (XML) 1.0*, http://www.w3.org/TR/1998/REC-xml-19980210 (accessed 13 December 2009).

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, http://www.w3.org/TR/2008/PER-xml-20080205/ (accessed 13 December 2009).

Brooks, F. P. (1987). No silver bullet essence and accidents of software engineering. *IEEE Computer*, **20.4**: 10–9.

Burnard, L. (1988). The Oxford Text Archive: principles and prospects. In Jenet, J.-P. (ed.), *Standardisation et échange des bases de données historiques. Table ronde internationale*. Paris: CNRS, pp. 191–203.

Burnard, L., O'Brien O'Keeffe, K., and Unsworth, J. (2006). Editors' introduction. In Burnard, L., O'Brien O'Keeffe, K., and Unsworth, J. (eds), *Electronic Textual Editing*. New York: Modern Language Association of America, pp. 11–12.

Burnard, L. (2007). *Reference Guide for the British National Corpus (XML Edition)*, http://www.natcorp.ox.ac.uk/XMLedition/URG/ (accessed 13 December 2009).

Burnard, L. and Bauman, S. (eds), (2007). *Text Encoding Initiative: P5 Guidelines*, TEI Consortium: Oxford, Providence, Charlottsville, Nancy. http://www.tei-c.org/Guidelines/P5/ (accessed 13 December 2009).

Buzzetti, D. (2002). Digital Representation and the Text Model. *New Literary History*, **33.1**: 61–88.

Cerquiglini, B. (1989). *Éloge de la Variante. Histoire critique de la philologie*. Paris: Éditions du Seuil.

Chatti, N., Kaouk, S., Calabretto, S., and Pinon, J.-M. (2007). MultiX: an XML based formalism to encode multi-structured documents. *Proceedings of Extreme Markup Languages, 2007.*

http://conferences.idealliance.org/extreme/html/2007/Chatti01/EML2007Chatti01.html (accessed 13 December 2009).

Chomsky, N. (1957). *Syntactic Structures*. The Hague, Paris: Mouton.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, **2**: 137–67.

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, **58.2**: 345–63.

Coombs, J. H., Renear, A. H., and DeRose, S. J. (1987). Markup systems and the future of scholarly text processing. *Communications of the ACM*, **30**(11): 933–47.

Cooper, A. and Reimann, R. (2003). *About Face 2.0 The Essentials of Interaction Design*. Indianapolis: Wiley.

Crane, G., Bamman, D., Cerrato, L., Jones, A., Mimno, D. M., Packel, A., Sculley, D., and Weaver, G. (2006). Beyond digital incunabula: modeling the next generation of digital libraries. *LNCS*, **4172**: 353–66.

Dahlström, M. (2000). Drowning by Versions. *Human IT*, 4. http://www.hb.se/bhs/ith/4-00/md.htm (accessed 13 December 2009).

Davis, M. (1958). *Computability and Unsolvability*. New York: MacGraw-Hill.

Davis, D. and Parashar, M. (2002). Latency performance of SOAP implementations. In *Proceedings of IEEE Cluster Computing and the Grid 2002*. Washington, DC: IEEE Computer Society, pp. 407–12.

Dekhtyar, A., Iacob, I. E., Jaromczyk, J. W., Kiernan, K., Moore, N., and Porter, D. C. (2006). Support for XML markup of image-based electronic editions. *International Journal on Digital Libraries*, **6.1**: 55–69.

DeRose, S. (2004). Markup Overlap: A Review and a Horse. In *Proceedings of Extreme Markup Languages*. http://conferences.idealliance.org/extreme/html/2004/DeRose01/EML2004DeRose01.html (accessed 15 December 2009).

D'Iorio, P. (2007). Nietzsche on new paths. The HyperNietzsche project and open scholarship on the web. In Fornari, M.C. and Franzese, S. (eds), *Edizioni e interpretazioni*. Pisa: ETS, http://www.hypernietzsche.org/doc/files/new-paths.pdf (accessed 15 December 2009).

Di Iorio, A., Schirinzi, M., Vitali, F., and Marchetti, C. (2009a). A natural and multi-layered approach to detect changes in tree-based textual documents. *Lecture Notes in Business Information Processing*, **24**: 90–101.

Di Iorio, A., Peroni, S., and Vitali, F. (2009b). Towards markup support for full GODDAGs and beyond: the EARMARK approach. In *Proceedings of Balisage: The Markup Conference 2009. Balisage Series on Markup Technologies*, vol. 3, doi:10.4242/BalisageVol3.Peroni01.

Dipper, S. (2005). XML-based stand-off representation and exploitation of multi-level linguistic annotation. In Eckstein, R. and Tolksdorf, R. (eds), *Proceedings of Berliner XML Tage (BXML 2005)*. Berlin: Humbolt University, pp. 39–50.

Durand, D. G., Mylonas, E., and DeRose, S. J. (1996). What should markup really be? Applying theories of text to the design of markup systems. In *Proceedings of ALLC/ACH Conference, Bergen*, http://gandalf.aksis.uib.no/allc/durand1.pdf (accessed 15 December 2009).

Eggert, P. (2005). Text encoding, theories of the text, and the 'work-site'. *Literary and Linguistic Computing*, **20**(4): 425–35.

Eisenstein, E. L. (1983). *The Printing Revolution in Early Modern Europe*. Cambridge: Cambridge University Press.

Fiormonte, D. (2003). *Scrittura e filologia nell'era digitale*. Turin: Bollati Boringhieri.

Froger, D. (1968). *La critique des textes et son automatisation*. Paris: Dunod.

Gabler, H. W., Steppe, W., and Melchior, C. (eds), (1984). *Ulysses: A Critical and Synoptic Edition/James Joyce*. New York: Garland.

Gerber, A. and Hunter, J. (2009). LORE: a compound object authoring and publishing tool for literary scholars. In Fraistat, N. and Kirschenbaum, M. (eds), *Digital Humanities 2009 Conference Abstracts, University of Maryland, College Park, June 22-25, 2009*. College Park, MD: MITH, pp. 124–6.

Gilbert, P. (1973). Automatic collation: a technique for medieval texts. *Computers and the Humanities*, **7**(3): 139–45.

GML. (1991). *GML Starter Set User's Guide*, http://publibfp.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/dsm04m00/CONTENTS (accessed 15 December 2009).

Goldfarb, C. (1973). *Design Considerations for Integrated Text Processing Systems*, IBM Cambridge Scientific Center Technical Report No. 320-2094. http://www.sgmlsource.com/history/G320-2094/G320-2094.htm (accessed 14 December 2009).

Goldfarb, C. (1990). *The SGML Handbook*. Oxford: Oxford University Press.

Goldfarb, C. (1996). *The Roots of SGML – A Personal Recollection*, http://www.sgmlsource.com/history/roots.htm (accessed 14 December 2009).

Goldfarb, C. (1997). SGML: the reason why and the first published hint. *Journal of the American Society for Information Science*, **48**(7): 656–61.

Hiltzik, M. A. (2000). *Dealers of Lightning Xerox Parc and the Dawn of the Computer Age*. New York: Harper Collins.

Hockey, S. (1991). *The ACH/ACL/ALLC Text Encoding Initiative: An Overview*, http://www.tei-c.org.uk/Vault/SC/teij16.gml (accessed 15 December 2009).

Hockey, S. and Martin, J. (1988). *The Oxford Concordance Program*. Oxford: Oxford University Computing Service.

Hopcroft, J. E. and Ullman, J. D. (1969). *Formal Languages and their Relation to Automata*. Reading, MS: Addison-Wesley.

Horridge, M., Drummond, N., Goodwin, J., Rector, A. L., Stevens, R., and Wang, H. (2006). The Manchester OWL syntax. In Grau, B.C., Hitzler, P., Shankey, C., and Wallace, E. (eds), *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*, http://ceur-ws.org/Vol-216/submission_9.pdf (accessed 15 December 2009).

Huitfeldt, C. (1992). MECS: a multi-element code system. *Working Papers from the Wittgenstein Archives at the University of Bergen*, 3 http://helmer.hit.uib.no/claus/mecs/mecs.htm (accessed 7 November, 2009).

Huitfeldt, C. (1995). Multi-dimensional texts in a one-dimensional medium. *Computers and the Humanities*, **28**: 235–41.

ISO-FI (2007). *Information Technology – Generic Applications of ASN.1: Fast Infoset*. Geneva: ISO, http://standards.iso.org/ittf/PubliclyAvailable Standards/c041327_ISO_IEC_24824-1_2007(E).zip (accessed 15 December 2009).

Kilpelainen, P. (1999). SGML and XML content models. *Markup Languages: Theory and Practice*, **1**(2): 53–76.

Lebert, M. (2008). *Project Gutenberg (1971–2008)*, http://www.gutenberg.org/etext/27045 (accessed 7 November 2009).

McCarty, W. (2005). *Humanities Computing*. Basingstoke: Palgrave Macmillan.

McLuhan, M. (1962). *The Gutenberg Galaxy: The Making of Typographic Man*. London: Routledge and Kegan Paul.

Mah, C., Flanders, J., and Lavagnino, J. (1997). Some problems of TEI markup and early printed books. *Computers and the Humanities*, **31**: 31–46.

Nedo, M. (1993). *Ludwig Wittgenstein Einführung/Introduction Wiener Ausgabe*. Vienna: Springer.

Neumann, J. (2006). *Ein allgemeiner Datentyp für die implizite Bereitstellung komplexer Texteigenschaften in darauf aufbauender Software*, Masters thesis, University of Cologne.

Neyt, V. (2006). Fretful tags amid the verbiage: issues in the representation of modern manuscript material. *Literary and Linguistic Computing*, **21**: 99–111.

Nicola, M. and John, J. (2003). XML parsing: a threat to database performance. *Proceedings of the Twelfth International Conference on Information and Knowledge Management New Orleans, LA, USA*. New York: ACM, pp. 175–8.

O'Donnell, D. P. (2005). *Caedmon's Hymn: A Multi-media Study, Edition and Archive*. Cambridge: D.S. Brewer.

OED. (2009). Simpson, J. A. and Weiner, E. S. C. (eds), *Oxford English Dictionary Online*. Oxford: Oxford Clarendon Press.

Ott, W. (1979). A text processing system for the preparation of critical editions. *Computers and the Humanities*, **13**: 29–35.

Palmer, S. B. (2000). *The Early History of HTML*, http://infomesh.net/html/history/early/ (accessed 15 December 2009).

Pierazzo, E. (2007). The encoding of time in manuscripts transcription: toward genetic digital editions. In Schmidt, S., Siemens, R, Kumar, A., and Unsworth, J. (eds), *Digital Humanities 2007 Conference Abstracts*. Urbana-Champaign: University of Illinois, pp. 150–2.

Portier, P.-E. and Calabretto, S. (2009). Methodology for the construction of multi-structured documents. *Proceedings of Balisage: The Markup Conference 2009. Balisage Series on Markup Technologies*, vol. 3, doi:10.4242/BalisageVol3.Portier01.

Raymond, D. R., Tompa, F. W., and Wood, D. (1992). Markup reconsidered. *First International Workshop on Principles of Document Processing, Washington DC, October 22-23*, http://db.uwaterloo.ca/~fwtompa/.papers/markup.ps (accessed 7 November 2009).

Renear, A., Mylonas, E., and Durand, D. (1993). *Refining our Notion of What Text Really Is: the Problem of Overlapping Hierarchies*, http://www.stg.brown.edu/resources/stg/monographs/ohco.html (accessed 7 November 2009).

Renear, A. (1997). Out of Praxis: three (meta)theories of textuality. In Sutherland, K. (ed.), *Electronic Text*. Oxford: Clarendon Press, pp. 107–26.

Robinson, P. M. W. (1997). New directions in critical editing. In Sutherland, K. (ed.), *Electronic Text*. Oxford: Clarendon Press, pp. 145–71.

Robinson, P. M. W. (1998). Publishing and electronic textual edition: the case of the wife of Bath's prologue on CD-ROM. *Computers and the Humanities*, **32**: 271–84.

Robinson, P. M. W. (2003). Where we are with electronic scholarly editions, and where we want to be. *Jahrbuch für Computerphilogie*, **5**: 125–46.

Ross, C. (1996). The electronic text and the death of the critical edition. In Finneran, J. (ed.), *The Literary Text in the Digital Age*. Ann Arbor: University of Michigan Press, pp. 225–31.

Schmidt, D., Brocca, N., and Fiormonte, D. (2008). A multi-version Wiki. In Opas-Hänninen, L.L., Jokelainen, M., Juuso, I., and Seppänen, T. (eds), *Proceedings of Digital Humanities 2008, Oulu, Finland, June, 2008*, pp. 187–8.

Schmidt, D. and Colomb, R. (2009). A data structure for representing multi-version texts online. *International Journal of Human-Computer Studies*, **67**(6): 497–514.

Schmidt, D. (2009a). Merging multi-version texts: a generic solution to the overlap problem. In Usdin, B.T. (ed.), *Proceedings of Balisage: The Markup Conference 2009*, doi:10.4242/BalisageVol3.Schmidt01.

Schmidt, D. (2009b). *Multiversiondocs: merge and edit N versions in one document*, http://code.google.com/p/multiversiondocs/ (accessed 15 December 2009).

Shillingsburg, P. (1996). *Scholarly Editing in the Computer Age*. Ann Arbor: The University of Michigan Press.

Shillingsburg, P. (2006). *From Gutenberg to Google*. Cambridge: Cambridge University Press.

Schneider, J. and Kamiya, T. (2008). *Efficient XML Interchange (EXI) Format 1.0*, http://www.w3.org/TR/exi/ (accessed 15 December 2009).

Shneiderman, B. (1998). *Designing the User Interface Third Edition*. Reading, MS: Addison-Wesley.

Smith, D. (1999). Textual variation and version control in the TEI. *Computers and the Humanities*, **33**: 103–12.

Smith, N. (2004). *Digital Publication for Digital Libraries*, http://chs75.harvard.edu/projects/diginc/pdfs/diginc (accessed 15 December 2009).

Sperberg-McQueen, C. M. and Burnard, L. (1988). *Design Principles for Text Encoding Guidelines*, http://www.tei-c.org.uk/Vault/ED/edp01.xml (accessed 15 December 2009).

Sperberg-McQueen, C. M. (1991). Text in the electronic age: textual study and text encoding, with examples from medieval texts. *Literary and Linguistic Computing*, **6**(1): 34–47.

Sperberg-McQueen, C. M. and Burnard, L. (eds), (1994). *Guidelines for Electronic Text Encoding and Interchange: TEI P3*. Oxford: Text Encoding Initiative.

Sperberg-McQueen, C. M. and Burnard, L. (eds), (2001). *TEI P4 Guidelines for Electronic Text Encoding and Interchange XML-compatible edition*. Oxford: Text Encoding Initiative.

TEI P1. (1990). *Guidelines for the Encoding and Interchange of Machine Readable Texts*, http://www.tei-c.org.uk/Vault/GL/teip1.tar.gz (accessed 7 November 2009).

Thaller, M. (1996). Text as a data type. In Lindebjerg, A., Ore, E. S., and Reigem, Ø. (eds), *ALLC-ACH '96, Book of Abstracts*, pp. 252–4.

Thaller, M. (2006). Strings, texts and meaning. In Gallet-Blanchard, L. (ed.), *Digital Humanities 2006 Conference Abstracts, Paris, Sorbonne 5-9 July*, pp. 212–4.

Tognazzini, B. (1992). *TOG on Interface*. Reading, MA: Addison-Wesley.

Usdin, B.T. (2009). Standards considered harmful. *Proceedings of Balisage: The Markup Conference 2009. Balisage Series on Markup Technologies*, vol. 3, doi:10.4242/BalisageVol3.Usdin01.

Vanhoutte, E. (2006). Prose fiction and modern manuscripts: limitations and possibilities of text-encoding for electronic editions. In Burnard, L., O'Brien O'Keeffe, K., and Unsworth, J. (eds), *Electronic Textual Editing*. New York: Modern Language Association of America, pp. 161–80.

Vetter, L. and McDonald, J. (2003). Witnessing Dickinson's witnesses. *Literary and Linguistic Computing*, **18**(2): 151–65.

Wilmott, S. (2002). The dichotomy of markup languages. In *Proceedings of Extreme Markup Languages, 2002*, http://conferences.idealliance.org/extreme/html/2002/Wilmott01/EML2002Wilmott01.html (accessed 15 December 2009).

Zapf, V. (2006). *HNML HyperNietzsche Markup Language*, http://www.hypernietzsche.org/events/sew/post/Slides and Texts_files/HNML.pdf (accessed 15 December 2009).