STANDARDS ARE OUR FRIENDS

Steve Tinney — University of Pennsylvania Museum

History

This is a revised version of a paper delivered to the Workshop on Computerized Cuneiform Databases, Helsinki, 11 September 1995. I am grateful to David Owen for convening the session on computerization, and to Bob Englund for conversation, suggestions and encouragement. My thanks go also to Dr. Jeremy Sabloff, Charles K. Williams, Director of the University of Pennsylvania Museum, who provided a crucial travel stipend. Last but not least, it is a pleasure to acknowledge that the vision offered here owes much to the pioneering hypertext work of Miguel Civil.

Introduction

The following programmatic paper offers an account of the current state of my experimentation, implementation and musings on the broadtopic of the computerization of Assyriology. To a large extent, my comments are driven by the needs of a major multi-decade project, the Pennsylvania Sumerian Dictionary, and only incidentally by the needs of individuals. I do not pretend that the following remarks can be applied *in toto* to the needs of every individual for every project (though I happen to believe that most of the following *is* relevant to the problems we face in our individual labours). Rather than dwelling, therefore, on the specifics of how we each use computers while closeted with the machines in our contemporary scriptoria, the principal concern of the present paper is to turn attention to the deeper methodological and practical problems that beset us in attempting to produce computerized cuneiform databases.

Information Systems

It should be said at the outset that I believe that such databases should eventually constitute only one component of a complete information system for Assyriologists (and others). A fully computerized dictionary, for example, could lead a user to a deeper understanding of a term by including links to representations of objects, ethnographic accounts of practices that are now alien to us, surveys of a thematic nature, like the articles in the *Reallexikon der Assyriologie*, and so on. This has several implications. Firstly, and most importantly, it is not enough to think of unrelated databases of certain *corpora* of texts, even if those *corpora* are divided into groups as large as Sumerian, Akkadian, Hittite, and so on. We must meet the difficult challenge of developing a system that will embrace texts in any language, and allow a degree of orthogonality in the forms in which the texts can be accessed and manipulated. Secondly, it is not enough to think only of the cuneiform textual component of such databases. The posited system must be able to handle multiple translations in multiple languages, commentaries, images of the tablets themselves and more. Further, we must embed within the textual component of any system the information needed to relate —and I use the term in the technical sense of allowing database applications to cross-link information from one database to another— to relate the texts with their archaeological and cultural context. By including rigorously defined findspot information with the tablets we are editing, for example, we could ultimately reconstruct the physical distribution of texts or text types before our very eyes by using Geographic Information Systems (GIS packages) to plot findspot information on maps and plans from various archaeological databases on different sites or even regions.

Longevity

There is a another aspect to these databases and the information system of which they will form part: their longevity. If we are ever to move beyond creating and discarding data sets time and again, our databases must be implemented in such a way that they can perdure for generations: no other goal makes sense. Several criteria must be met in order for this to be realistic. The physical distribution, knowledge and availability of the databases must be great enough to provide adequate incentive to migrate them from one operating system to another, from one machine type to another, as the technology shifts underneath them. They must be organized in such a way that they can be updated easily and centrally, with audit trails and automated propagation of changes. They must be easily adaptable at the local level so that people can, for example, maintain their own variant translations or transliterations without impinging on the canonical versions of texts that derive from the central database.

Structure and Flexibility

The goal, then, should be a comprehensive, lastingly useful group of databases which makes up a complete information system for Assyriologists. How might one achieve such a goal? The key is structure: both the structure of the information, and the structure of the "core form" in which it is represented. I use the term "core form" because I believe that it is unrealistic and unnecessary to expect an information system in a single format to satisfy everybody's needs and desires. A carefully structured core, or deep structure, can be converted to all manner of surface representations for consumption by different audiences, by different programs, for different purposes and in different formats (even print, if that should still be in fashion in a few decades). It is neither a possibility nor a requirement that all of these surface representations be capable of bidirectional conversion, *i.e.*, a conversion which returns the information to exactly the representation from which it started. Since changes to the core system will have to be maintained centrally somehow, it can simply be a requirement that the maintainers of that system work either with the core format directly or with a bidirectionally stable representation of it.

Structure and Standards

Clearly, there is a huge amount of work implied in all of this, which naturally offers an incentive to look for ways of reducing the workload (laziness is the mother of invention). One of the current problems with the computerized representation of information is the plethora of different specifications. We all have character set headaches when we get data from colleagues, but this is only a tiny tip of the information structure iceberg. The real problem lies in our representation of the parts of a text and its ancillary information, for if we are to create the system I am describing we must be able to know with absolute precision exactly which forms of information any given character in a document is contributing. Part of the transliteration of a grapheme? A grapheme boundary? Transliteration or translation? Commentary? Parallel? Duplicate? We could, of course, start from scratch and write specifications for notations that covered every one of these aspects, as well as a mapping of character numbers to their glyphic representations. We would be doubling our work, and there *is* a better solution. Standards.

Standards and Stability

Standards bring with them several benefits. They have usually been designed and written by experienced people who have devoted large amounts of time and thought to getting it right. By contrast, specifications put together to meet specific needs usually need to be tweaked repeatedly each time a new wrinkle in our needs arises. Screen fonts and character sets, for example, always seem to need just one more character to be added to them to be complete. One of the drawbacks of this latter, incremental approach to design is that the documentation rarely keeps pace with the implementation. Another of the great advantages of standards is that they are published, public reference points to which anyone can return at any time. Because of the huge amount of work and negotiation involved in ratifying a standard, they tend to be more stable than, say, the file formats of commercial software, which seem to change constantly under our feet. While each new iteration of Microsoft Word may be able to read its predecessor's files, the fact that incompatibilities creep in gradually, that sometimes a little loss of information may be the cost of upgrading, makes commercial, proprietary software and data formats unacceptable for a system that is designed to survive generations. Furthermore, most commercial software makers are reluctant to invest in products for every operating system or every platform. With standards, any number of different companies can make products that provide conformant implementations of a standard, each for their own niche. Thus, at least for mature, widely accepted, standards, the same files can be manipulated on many different platforms by a wide range of different software. Finally, much of the activity surrounding standards takes place outside of the commercial realm. Everything described in this essay can be done with free software, all of which is also readily available as source code. This provides a further buffer against change, since the software on which a long term project depends can be recompiled for, or ported to, other platforms, and bugs can be fixed easily and quickly, without having to wait for a commercial software house to decide that the bug is affecting enough customers to make a fix financially viable.

Characters

A few words of introduction about character sets may be helpful. To the computer, any character is simply a number. The number of characters in a font or character set is normally limited by the largest number that can be stored in the data type set aside for characters. We call each available slot, or number, a *codepoint.* We call the set of relationships between characters and code points an *encoding*. In the early days of computing, the data type used for characters often had only 6 bits, meaning that only 64 code points were available. For many years, especially on Unix systems, 7 bits was the norm. This gave a set of 128 characters which was standardized many years ago as ASCII, and is fairly stable — local variations mainly concern currency symbols, for example. It was, and is, an eighth bit available for the representation of characters in almost all machines that brought us into anarchy. The eighth bit doubles the number of available code points, and it took no time at all for the extra 128 code points to be used for accented characters and box drawing components. Unfortunately, with no standardization and too few characters to meet everybody's needs, a plethora of different character sets came into use not only at the commercial but also at the private level.

Unicode

This is all about to change. To quote from "About This Book" in the Unicode standard:

> "The Unicode standard had its genesis in early 1988 when a group of information professionals with extensive experience in multilingual computing agreed that no encoding methodology used in their field possessed the elegance and simplicity of ASCII. The Unicode character encoding was established as a fixed-width encoding of 16 bits, which would provide a

> sufficient number of unique codes for the world's scripts and technical symbols in common use, and at the same time promote efficient and flexible system design."

Today, Unicode support is on the increase in operating systems, and applications are gradually following suit. Unicode is not a product of the International Standards Organisation, but there is such a standard under development (ISO DIS 10646), that will be a superset of Unicode extended to allow larger character sets with even more than Unicode's 65,536 code points.

From Unicode to Markup

It is important to understand that Unicode is a character set, and only that. Much confusion is caused by the fact that Unicode contains all the necessary characters for, say, Hebrew and Arabic scripts, each assigned to their own code points, without providing a means for specifying which language any given character represents (though Unicode does reserve code points for controlling the direction of text). This is quite conscious and deliberate, since the specification of the language belongs to a level of information structure above that of the character. We do not expect ASCII to distinguish between American and English, for example. There is, in fact, a Standard, ISO 639/2, which assigns three letter abbreviations to many of the world's ancient and modern languages, and which can be used as a basis for indicating languages in textual markup. Such markup is what concerns us next.

Structural Document Markup

Markup is the name for the information added to document data for various purposes. Markup is commonly procedural, that is, one may add instructions to the effect that a certain word should be italicized, or that a certain amount of space should come before or after a heading, but the value of this kind of markup is restricted to printing out documents, and rarely helps in their analysis. More interesting to us is structural markup, which separates the logical elements of the document. In this kind of markup, one indicates the fact that a certain piece of text is a heading at some level, or that a piece of text is emphasized, without having to specify the effect of the text's status on its appearance.

SGML: A Standard for Structural Markup

SGML, the Standard Generalized Markup Language, owes its origins to a concern with structural, or descriptive, markup, as the following excerpt from Annex A of the SGML standard indicates:

> "It is called 'generalized markup' because it does not restrict documents to a single application, formatting style, or processing system. Generalized markup is based on two novel postulates:

> a) Markup should describe a document's structure and other attributes rather than specify processing to be performed on it, as descriptive markup need be done only once and will suffice for all future processing.
> b) Markup should be rigorous so that the techniques available for processing rigorously defined objects like programs and data bases can be used for processing documents as well".

In other words, SGML's emphasis on descriptive, or structural, markup, coupled with the rigour with which one can define the data structures of a document, makes SGML much more than a generic document description language: it is, in fact, as has become widely realized over the last few years, a way of turning ordinary text files into database objects. It is a language in which we can define the structure of our content without resorting to program-specific or proprietary encoding. It is means of intertwining structure with data such that arbitrary applications —databases, hypertext programs, text formatting programs— can read our work without our constantly having to adjust its organization. It is, in short, the answer.

SGML and Hypertext

Probably the major form of exposure that people have to SGML, without realizing it, is on the World Wide Web, often called simply "the Web". Documents on the Web are primarily written in a language called Hypertext Markup Language, or HTML, which is, in fact, nothing more than an SGML Document Type Definition. The Web is an impressive demonstration of the power and potential of the combination of SGML and hypertext, even if the form overshadows the content in many cases. The Web has to be used to be understood, but the essence of it is that it provides global hypermedia over networks or on local disks, with the physical location of documents being irrelevant from the user's perspective. Text, sound, still and moving images can be used in combination, and links can be made, for example, from maps to images, or maps to video. Remember, though, that HTML and the Web are not the answer: SGML is the answer. HTML, moreover, is only one way of doing hypertext with SGML: there is also a standard, the HyperMedia/Time-Based Structuring Language, or HyTime, which is an extended version of SGML with special constructs for hypermedia.

DSSSL: Standardized Conversion

Finally, the issue of conversion has been touched upon several times in my remarks. There is a very new standard designed to add a conversion layer to SGML, so that DTD's can be converted to other forms in a standardized manner. This standard is entitled Document Style Semantics and Specification Language, or DSSSL to its friends. DSSSL is defined as a language based on the Lisp dialect, Scheme, but has only just been ratified. A DSSSL engine is under development by James Clark.

Implementations

By way of illustration, it may be informative to look briefly at the computerization of the Pennsylvania Sumerian Dictionary. The standards I have just discussed offer us the potential of creating an openly defined dictionary which will no longer be bound by the form in which it is printed. If we view a dictionary article as an outline, we can view the integration of semantic divisions, analysis and references as a composite document in which each element is its own section. Such a document will rely for its examples not on text, statically embedded in the article, but on references to the citations which could then be drawn into the dictionary articles dynamically when viewed or printed. For example, the core form of the online dictionary will store references in the form "Nippur Lament line 1", or "CBS 10000 column 3 line 4", and the actual text and translation for those references will be retrieved from the core text corpus to produce versions of the dictionary article for use on the Web; for printing and so on. Users reading the dictionary articles with a hypertext browser will be able to jump back again from the dictionary to the texts referenced so that they can see the broader context, or read any commentary available for the lines in question. From there, they could access text copies or photographs.

Implementation Requirements

There are three primary issues involved in implementing the system I just outlined. Firstly, a DTD is needed for dictionary articles themselves; secondly, the thorny problem of quoted context must be addressed; thirdly, a rigorously-defined corpus of Sumerian is an absolute requirement. The first of these requirements is the least difficult, and a basic DTD for dictionary articles is illustrated in Fig. 1, with an example in Fig. 2 (see the following page). Note that because these articles must be preprocessed before printing, various ancillary information which is not conventionally part of the printed dictionary could be included, such as a section in which every reference to the word in the secondary literature is listed, or a header field which gives the phonology of the word, at least so far as it can be determined. The second problem, the delineation of quoted context, is avoided at present because the citations are given in full in the body of each reference. What is needed is a special set of markup tags which indicate phrase and clause boundaries, as well as other textual divisions. Such a set of tags need not be dictionary-specific, since these divisions are also of interest for grammatical and syntactical work, and may also provide interesting avenues for the application of discourse analysis.

A DTD for Dictionary Articles

```
ELEMENT word - - (header , unilingual? , bilingual? , lexical? , discussion?)
ELEMENT header - - (term , entry , written , meaning , periods , filename )
ELEMENT term - o (#PCDATA)
ELEMENT entry - o (#PCDATA)
ELEMENT written - o (#PCDATA)
ELEMENT meaning - o (#PCDATA)
ELEMENT periods - o (#PCDATA)
ELEMENT filename - o (#PCDATA)
ELEMENT unilingual - - (ref*)
ELEMENT bilingual - - (ref*)
ELEMENT lexical - - (ref*)
ELEMENT discussion - - (paras*)
ELEMENT ref - - (text , tran , (cite , bibl , note)+)
ELEMENT text o o (#PCDATA)
ELEMENT tran - o (#PCDATA)
ELEMENT cite - o (#PCDATA)
ELEMENT bibl - o (#PCDATA)
ELEMENT note - o (#PCDATA)
ELEMENT paras - - (#PCDATA)
```

Fig. 1. Example of a basic DTD for dictionary articles.

```
‹!DOCTYPE word SYSTEM "psd.dtd"›
‹word›
‹header›
‹term›       a‹sub›2‹/sub›-SIG‹sub›4‹/sub›-da-ri-a
‹entry›      a‹sub›2‹/sub›-SIG‹sub›4‹/sub›-da-ri-a in udu-a‹sub›2‹/sub›
             -SIG‹sub›4‹/sub›-da-ri-a
‹written›
‹meaning›    (a designation of sheep)
‹periods›    Post-OB (Emar)
‹filename›   a2-SIG40
‹/header›
‹lexical›
‹ref›        udu-a‹sub›2‹/sub›-SIG‹sub›4‹/sub›-da-ri-a
‹tran›
‹cite›       Hh XIII Emar 46'
‹bibl›       Emar 6/4, 108
‹note›
‹/ref›
‹/lexical›
‹/word›
```

Fig. 2. Example of a dictionary article (á-sig$_4$-da-ri-a).

A DTD for Cuneiform Text Editions

These context-tags, then, would form part of a DTD for cuneiform texts which in turn would be the basis for the third of the requirements I outlined above, *i.e.*, a rigorously defined corpus of Sumerian. The development of a complete, standardized DTD would be a good topic for another workshop, or a working group, but a few observations may be made here. The basic components are not difficult to list: introductory preamble, a bibliography, a list of sources, composite text, morpho-lexemic transcription, one or more translations (by various authors and in different languages), a score or textual matrix for texts with more than one exemplar, commentary, text copies and photographs of the text. Some of these basic components require careful specification. For example, the sources to a text need to have a variety of fields giving their date, the script they use, their provenience (for links with archaeological databases), their publication places and so on. The problems of specifying joined fragments that have been published in parts in different places must be resolved, since it is important if we are ever to be able to point at a line, or a variant, and say to the computer "show me the copy".The machine must be able to determine the relationship between sources and copies. We need to mark languages, of course, and also the various realia: tags for personal names, divine names and soon. Such tagged names could allow the computer to display, at the click of a key, Reallexikon articles, or genealogical information. A further problem lies with original texts, duplicates and parallels: our present page-oriented conventions for indicating such relationships must be replaced by something much more flexible, for example link tables that give the order and location of all duplicate and parallel passages in such a way that the computer, not the user, can build a convenient list of them at will (that is, the will of the user). Finally, and most fundamentally, we must ensure that our notations provide a unique name for every part of every text, so that jumping from transliteration to translation to copy —or presenting all three simultaneously in response to a search request— can be fully automated.

Concluding Observations

There would be many side-benefits from the production of a corpus of Mesopotamian texts such as has just been discussed. One example is that we could finally produce a new sign-list in which the actual occurrences of sign values were derived from the transliterations, and the palaeography could be studied by reference to the photographs to which those transliterations provide a form of index.

Clearly, much of the preceding relies on using a variety of different pieces of software with a single, carefully-prepared, corpus. This core can be processed on, and transported to, all manner of machines and with all manner of software. The core constitutes a low level dataset which can be manipulated in higher level ways.

This layered approach to information systems is crucial for computerizing the field of Assyriology. I have covered in the foregoing remarks the questions of only the lowest two layers, the character set —for which I propose Unicode as the answer— and the document structure — for which I propose SGML as the answer. Whatever else we are to do, I believe we must begin with those layers in order to establish firm and lasting foundations.