Unicode, XML, TEI, Ω and Scholarly Documents

Yannis Haralambous

Unicode, XML, TEI, Ω and Scholarly Documents

Yannis Haralambous
Atelier Fluxus Virus
187 rue Nationale
59800 Lille, France
yannis@fluxus-virus.com

J'en étais là de mes pensées,
lorque, sans que rien en eût
décelé les approches, le
printemps entra subitement dans
le monde. [Ara, p. 11]

1 Introduction

Scholarly documents, like bilingual parallel texts or critical editions, involving any kind of classical or Oriental language are a challenge to computers because of their intrinsic complexity in form and structure. In this paper we will try to discuss some of the issues involved in their processing, with particular emphasis to issues related to the Unicode encoding.

To be stored or communicated in the most efficient way, a scholarly document has to be *encoded* and *structured*. By the former we mean the representation of the textual data in Unicode; by the latter we mean the markup of the document in a markup language, like the Extensible Markup Language (XML). On diagram 1 the reader can see a schematic representation of the process:

1. data, which can be of any origin (keyboarded text, OCR, legacy data,...) is encoded and structured as to obtain a valid XML document, encoded in Unicode;
2. this document, to be processed, needs a certain amount of linguistic background information and transformations (for example: rules for upper- and lowercasing, hyphenation, sorting order, etc. which depend on language and dialect);
3. finally the document can be processed in several ways: the most traditional being of course the printed book, but other being there as well: the electronic document, a database, voice synthesis, etc.

All processes involved in this diagram are clear and relatively trivial when the document contents is plain English text and its structure is relatively simple; but in the case of scholarly documents involving classical or oriental languages there are many open questions.

The first question that arises is to what extent Unicode should be used and when one should rather use XML entities: it is clear that in French, using the Unicode character U+00E9 LATIN SMALL LETTER E WITH ACUTE is a more elegant choice than using the entity é, but on the other hand, entities may be a very efficient way to avoid Unicode ambiguities, like the treatment of the Greek mute iota or the Arabic hamza.

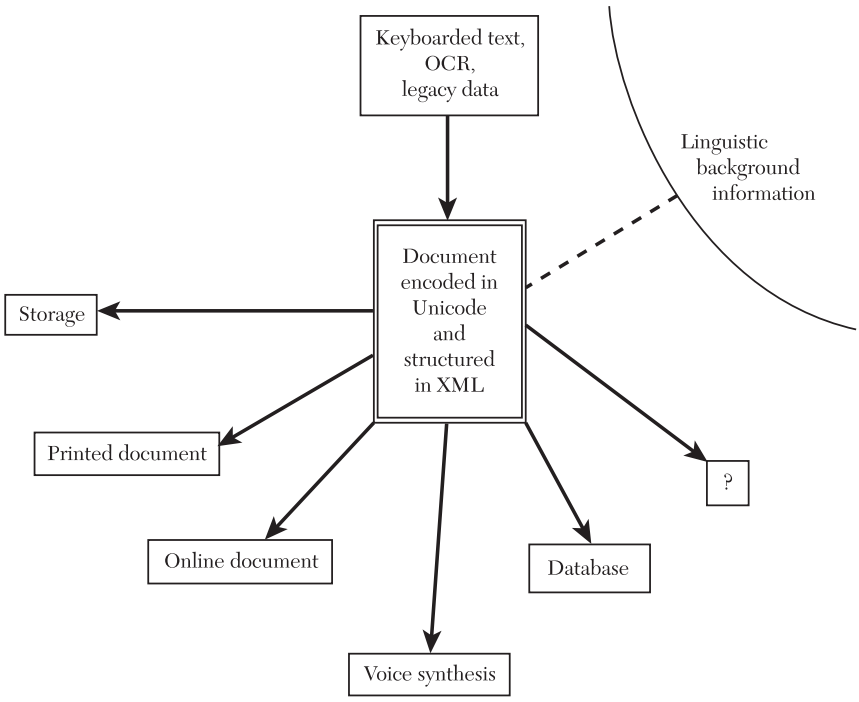

Figure 1: The various transformations of a XML document.

A second question is about *how* to use Unicode: should pre-combined characters or combining diacritics be prefered? Should the Arabic hamza be used graphically or grammatically? Should the accents and breathings in front of capital Greek letters be used as standalone characters or as combining diacritics?

Before we address these questions we will give a short introduction to tools we will be mentioning: XML, TEI and Ω.

2 XML

The Extensible Markup Language XML is a document processing standard proposed by the World Wide Web Consortium (W3C). It is actually a simplified form of the Standard Generalized Markup Language SGML, ISO 8879.

XML uses tags to markup data. These tags can be of several forms, the most important being the following:

1. *element tags:* these are of three kinds: *opening tags*, like `<title>`, *closing tags* like `</title>`, and *stand-alone tags* like `<pause/>`;
2. *entity tags:* `&eacute;`, `&CompanyLogo;`, etc.;
3. *processing instruction tags:* `<?tex \rule{20pt}{.5pt} ?>`, etc.

An *element* is a part of an XML document; it may be either a stand-alone element tag, or a part of the document starting with an opening tag and finishing with a closing tag. Whatever is between the opening and closing tags is called the *contents* of the element: this may be text or other elements, provided they are *properly nested* (the last opened element must be the first closed). Elements are the

building blocs of XML documents: the main body of the document is already an element; it is also the root of the hierarchical tree of nested elements of the document. The leaves of this tree are either text blocks or *empty* elements (elements whose contents are empty, often —but not necessarily— denoted by stand-alone tags).

Element tags carry additional information, which is not part of the contents of the element: this information is given in the form of *attributes* and their *values*. For example the element

```
<footnote xml:lang="en" num="2" para>...</footnote>
```

has three attributes: `xml:lang` whose value is `en`, `num` whose value is 2 and finally `para` which has no value.

An XML document consists of three (largely unequal in length) parts:

1. the *XML declaration*, which is a single tag with a certain number of arguments (one of which is the encoding of the document);
2. the *Document Type Definition* (DTD), which is a very precise declaration of all elements and attributes used in the document, as well as their properties: which elements can be contained in other elements, in what order, which ones can contain text, etc.
3. the *main body* of the document, the elements of which must obey to the rules given in the DTD.

A very essential property of XML documents is the fact that they can be parsed by software and *validated*. A valid XML document is one whose tags are written correctly and whose elements obey to all rules given in the DTD. Validity of a document is totally independent of its actual contents: it is a formal property which ensures that the document will be treated correctly by XML-compliant software.

Elements provide us with the *logical structure* of a document. An equally important concept, which allows an XML document to have a very flexible *physical structure* is the one of *entity*. Like elements, entities are also *declared* in the DTD; in the main body we use *entity references* (whose tags are of the form `&hamza;`). The idea is very simple: when a document is parsed or processed, entity references are replaced by their contents as declared in the DTD. The contents of an entity can be a text block (including other entities, as long as there is no infinite loop), an external XML file (given by an URL) or an external file in a different format (for example an image in GIF format, or a TEX file) in which case the processing software must know how to handle the file's format.

Entities are often used like "macros" in the sense that something which is unclear or temporary at the time of writing, is declared in the DTD, and the corresponding entity reference is used in the document. A change in the entity's declaration will then have repercussions to the whole document, wherever the specific entity references have been used. For example let us suppose that you are editing a 16th century text in which "&c." is sometimes used instead of "etc." You are not certain yet if the publisher would like to keep that spelling but you do wish to keep the information in the source file; in that case using an entity `&etc;` would be an efficient solution: in the DTD you can then decide if `&etc;` should expand as "&c." or as "etc."

Entities are very important in the scope of this paper because they have been a natural substitute to Unicode encoding: if you are writing an HTML document (HTML is a special case of XML: in fact it is XML with a specific pre-defined DTD) which instead of being encoded in Unicode, is, for example, encoded in ISO 8859-7 (Greek) and need a French 'é' then you have no other choice than using the entity `&eacute;`. Of course once you decide to encode your document in Unicode, you have the choice between the actual Unicode character `U+00E9 LATIN SMALL LETTER E WITH ACUTE` and the entity `&acute;`.

You have even a third (intermediate) choice: the *character entity reference* which is an entity reference explicitly giving the code position of a character in the Unicode table; using hexadecimal notation the French 'é' is then written `&#x00E9;` (where `&` stands for "entity," `#` specifies the fact that it is a character entity, `x` stands for hexadecimal notation, `00E9` is the hexadecimal code of 'é' in the Unicode table and `;` closes the tag).

2.1 XML and Unicode

Thru the attribute `encoding` one can define the encoding of an XML document, in the XML declaration. To specify Unicode is not enough: one must define the Unicode transformation format, for example UTF-8 ([Uni, §2.2]). So, for example, an XML document encoded in Unicode would have the XML declaration:[1]

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

It is interesting to note that according to XML specifications, Unicode happens to be the default encoding for XML documents.

Also note that the encoding is not only used for the textual contents but also for the tag names. When the document encoding is Unicode, then tag names can be written using any Unicode character having the "letter property," ([Uni, §4.5]). Under these circumstances an XML document like the following ([Kor]) can be perfectly valid:

```
<τίτλος ἀρίθμηση="ὅχι" ἐπίπεδο="1">Πρόλογος</τίτλος>
<π><κεφαλαία>᾽Αδαμαντίου Κοραῆ</κεφαλαία> &μεγάληπαύλα ; τοῦ ἐν τῇ περὶ ἅπαν μαθήσει ἀριπρεποῦς, καὶ περὶ τὴν τῆς φιλολογίας ἐν γένει, ᾽Αδαμαντίου, λέγω, τοῦ Κοραῆ ἡ πολύκροτος φήμη ἐνηχεῖ θαυμασίως εἰς ὅλων τὰς ἀκοάς.</π>
<π>Πρῶτον βιογραφικὸν περὶ αὐτοῦ ἄρθρον κατεχωρίσθη ἐν τῇ <γαλλικά>Biographie nouvelle des contemporains. Paris 1822, tome V, pag. 52-55</γαλλικά></π>
```

Nevertheless most of nowadays available XML parsers are not able of handling such XML documents; an interesting exception is *XML Spy* ([wSpy]) which claims to be fully Unicode-compliant (at least under Windows 2000).

2.2 References for further reading on XML

A search on `www.amazon.com` returned 255 (!) books on XML. The books we would suggest are: Robert Eckstein's *XML Pocket Reference* (a $4\frac{1}{4}'' \times 7''$ 100 pages booklet, very clear and well organized) [Eck], Neil Bradley's *XML Companion* (quite complete) [Bra] and Paul Spencer's *XML Design and Implementation* (the most pedagogical) [Spe].

For French language readers we also recommend the proceedings of the 1999 GUTenberg Conference: *Actes du colloque GUT'99, seconde partie : XML* [Gut] and [wGut] (which also contains the XML specifications in French).

The XML specifications as well as a lot of material around XML can be found on the OASIS Web site [wOas].

[1] Unfortunately the XML declaration does not allow us to specify the version of the Unicode standard used, so maybe a comment of the type

```
<!-- Unicode v. 2, as described in ISBN 0-201-48345-9 -->
```

would fill that gap.

An interesting document on XML and Unicode is also the (for the moment, proposed draft) Unicode Technical Report #20 *Unicode in XML and other Markup Languages* by Martin Dürst, Mark Davis, Hideki Hiura and Asmus Freytag [wUtr]. Finally we strongly recommend the (W3C working draft) document *Character Model for the World Wide Web* by Martin Dürst and François Yergeau [wCmw].

3 TEI

The Text Encoding Initiative[2] (TEI) is a Document Type Definition (DTD) for SGML documents, SGML being the ancestor of XML; in fact there is already an XML version of TEI [wPiz]. In the following we will not mention any fact which is exclusively SGML-related, so everything can be considered as being true for XML-TEI, even if the latter has not yet been officially released.

It should be noted that the term "encoding" in TEI name and documentation is used in the sense of "structuring," (and not in the sense of "character encoding" as is Unicode). TEI is a DTD specialized in documents used in the humanities, scholarly documents, etc. Here is an excerpt from the TEI documentation [wTdc]:

> *[TEI is] the result of over five years' effort by members of the research and academic community within the framework of an international cooperative project called the Text Encoding Initiative (TEI), established in 1987 under the joint sponsorship of the Association for Computers and the Humanities, the Association for Computational Linguistics, and the Association for Literary and Linguistic Computing.*
> *The impetus for the project came from the humanities computing community, which sought a common encoding scheme for complex textual structures in order to reduce the diversity of existing encoding practices, simplify processing by machine, and encourage the sharing of electronic texts. It soon became apparent that a sufficiently flexible scheme could provide solutions for text encoding problems generally. The scope of the TEI was therefore broadened to meet the varied encoding requirements of any discipline or application. Thus, the TEI became the only systematized attempt to develop a fully general text encoding model and set of encoding conventions based upon it, suitable for processing and analysis of any type of text, in any language, and intended to serve the increasing range of existing (and potential) applications and use.*
> *What is published here is a major milestone in this effort. It provides a single, coherent framework for all kinds of text encoding which is hardware-, software- and application-independent. Within this framework, it specifies encoding conventions for a number of key text types and features. The ongoing work of the TEI is to extend the scheme presented here to cover additional text types and features, as well as to continue to refine its encoding recommendations on the basis of extensive experience with their actual application and use.*

As TEI tries to be as complete and general as possible, it has become very large: there are more than 500 different elements defined and the printed documentation is around 1,300 pages. To minimize resources, it has a modular structure: users can include in their documents only the modules they need; there is also the possibility of defining new elements and entities.

The basic modules of the TEI DTD are the following:

1. the core tag set, declaring elements available in all TEI documents;
2. the header tag set, declaring elements used in document headers: every TEI document must have a header which describes the file, the encoding and the text profile and gives the revision history of the document;
3. the prose tag set: as its name says, it is the set of tags used for prose text;

[2] And not "Total Employee Involvement," as defined by Japan Human Relations Association…

Figure 2: A sample critical edition ([Chr]).

Διὸ παραινῶ φεύγειν αὐτῶν τὴν μανίαν· μανίαν γὰρ ἔγωγέ φημι εἶναι ἐσχάτην τὸ φιλονεικεῖν εἰδέναι τί τὴν οὐσίαν ἐστὶν ὁ Θεός. Καὶ ἵνα μάθῃς ὅτι μανίας ἐσχάτης τοῦτο, ἀπὸ τῶν προφητῶν ὑμῖν τοῦτο ποιήσω φανερόν· οἱ γὰρ προφῆται οὐ μόνον τί τὴν οὐσίαν ἐστὶν ἀγνοοῦντες φαίνονται, ἀλλὰ καὶ περὶ τῆς σοφίας αὐτοῦ πόση τίς ἐστιν ἀποροῦσι· καίτοι γε οὐχ ἡ οὐσία ἀπὸ τῆς σοφίας, ἀλλ' ἡ σοφία ἐκ τῆς οὐσίας. Ὅταν δὲ μηδὲ ταύτην δύνωνται καταλαμβάνειν οἱ προφῆται μετὰ ἀκριβείας, πόσης ἂν εἴη μανίας τὸ τὴν οὐσίαν αὐτὴν νομίζειν δύνασθαι τοῖς οἰκείοις ὑποβάλλειν λογισμοῖς; Ἀκούσωμεν τοίνυν τί φησιν ὁ προφήτης περὶ αὐτῆς· «Ἐθαυμαστώθη ἡ γνῶσίς σου ἐξ ἐμοῦ.» Μᾶλλον δὲ ἀνωτέρω τὸν λόγον ἀγάγωμεν· «Ἐξομολογήσομαί σοι, ὅτι φοβερῶς ἐθαυμαστώθης.» Τί ἐστι «φοβερῶς»; Πολλὰ θαυμάζομεν μόνον, ἀλλ' οὐ μετὰ φόβου, οἷον κιόνων κάλλος, τοίχων ζωγραφίαν, ἄνθη σωμάτων· θαυμάζομεν πάλιν τῆς θαλάσσης τὸ

1 μανίαν[2] : μανίας EL DG OVX || ἔγωγε om. DG || 2 φημι B : transp. post. ἐσχάτης E O om. cett. || εἶναι : οἶμαι VX om. DG || ἐσχάτης EL DG OVX om. B || τὸ om. DG || 3 μανίας ἐσχάτης : μανίαν ἐσχάτην AC om. B || 8 μετ' corr. Duc || 14 μόνον : νῦν E om. CL G VX || 14-15 κάλλη τύπων ζωγραφίας Duc e cod. Paris. 777.

a. Ps. 138, 6.
b. Ps. 138, 14.

1. Le mot προφήτης a des nuances diverses dans l'Ancien et le Nouveau Testament et à l'intérieur de chacun d'eux, mais ces sens divers ont un caractère commun : le prophète est l'homme qui parle sous l'inspiration de l'Esprit. Cf. *hom.*, III, li. 150 s. C'est ainsi que Jean va citer successivement des textes du Psalmiste, d'Isaïe et de Paul.

2. On remarquera que Jean utilise très fréquemment la tournure οὐ μόνον... ἀλλὰ καί. C'est sans doute une habitude de style enseignée par la rhétorique mais elle correspond chez lui, croyons-nous, à une tendance profonde. Son dynamisme naturel ne se contente pas d'une constatation pure et simple ; il la renforce soit dans un sens positif, soit dans un sens négatif. Ici, la tournure marque un nouveau point gagné sur l'adversaire, grâce au témoi-

Je vous exhorte donc à éviter leur folie, car c'est le comble de la folie que s'acharner à connaître Dieu dans son essence. Et pour que vous compreniez que c'est bien en effet le comble de la folie, je vous le montrerai à l'évidence par le témoignage des écrivains sacrés[1] : non seulement ceux-ci ignorent manifestement ce qu'il est dans son essence, mais encore ils ne savent que dire de l'étendue de sa sagesse[2] ; or ce n'est pas l'essence qui dérive de la sagesse, mais la sagesse de l'essence. Quand donc les écrivains sacrés ne peuvent pas même délimiter celle-là avec exactitude, quelle est la folie de ceux qui croient pouvoir soumettre son essence elle-même à leurs propres raisonnements[3] ? Écoutons donc ce que dit l'écrivain sacré à ce sujet : « La connaissance que tu as de moi m'a été un objet d'admiration[a]. » Mais suivons plus loin son propos : « Je te bénirai, parce qu'on t'admire avec crainte[b]. » Que signifient ces mots : « avec crainte » ? Nombreuses sont les choses que nous nous contentons d'admirer[4], mais non pas avec crainte, par exemple la beauté des colonnes, ou des chefs-d'œuvres de la peinture, ou des corps dans leur

gnage d'hommes inspirés. Loin de prétendre connaître l'essence de Dieu, ils ont un mouvement de recul, de confusion, même lorsqu'il s'agit de parler des manifestations de la sagesse.

3. Jean oppose ici la saisie claire du mystère de Dieu, dont se prévalent les Anoméens, καταλαμβάνειν μετὰ ἀκριβείας, à la connaissance conjecturale, forcément imparfaite lorsqu'elle ne s'appuie que sur des raisonnements humains. En fait, Eunome affirme qu'il ne s'appuie pas uniquement sur « ses propres raisonnements », mais grâce à une théorie du langage qu'il tire de l'Écriture (*Gen.* 1, 3), il attribue à Dieu l'origine du nom qui le désigne exactement. Voir *Apologia*, chap. 7, *PG* 30, 841.

4. Ici encore, le choix des textes permet à Jean de faire progresser son argumentation. L'usage du verbe θαυμάζειν dans le premier texte suggère l'admiration dans plus. Mais dans le second, l'adjonction de φοβερῶς montre l'homme saisi d'une crainte révérentielle en présence de Dieu, devant l'océan infini de sa sagesse : τὸ ἄπειρον... πέλαγος τῆς τοῦ Θεοῦ σοφίας.

4. the verse tag set;
5. the drama tag set;
6. the spoken tag set, containing declarations of tags used for transcriptions of spoken texts;
7. the dictionaries tag set, useful for printed dictionaries;
8. the terminology tag set, useful for terminological data files;
9. the general tag set and the mixed tag set: both allow the combined use of several tag sets from the list above.

For example, here are some of the elements available in all TEI documents:

`<p>` the paragraph element;
`<foreign lang="xx">` identifies a word or phrase as belonging to some language "xx" other than that of the surrounding text;
`<emph>` for emphasis;
`<hi rend="something">` marks a word or phrase as graphically distinct from the surrounding text: the `rend` attribute describes the rendition or presentation of the element contents;
`<distinct>` identifies any word or phrase which is regarded as linguistically distinct, for example as archaic, technical, dialectal, non-preferred, etc., or as forming part of a sublanguage. Attributes include: `type` which specifies the sublanguage or register to which the word or phrase is being assigned, `time` which specifies how the phrase is distinct diachronically, `space` which specifies how the phrase is distinct diatopically and `social` which specifies how the phrase is distinct diastatically.

It is interesting to note that while a fixed markup system like HTML provides the user with only a few, predefined, types of text highlighting, and on the other hand, while general markup systems like XML leave the user absolute freedom, TEI chooses an intermediate solution: there is a single element for highlighting (`<hi>`), but the user can specify the reason of highlighting or the presentation of highlighted text, using the `rend` attribute.

Furthermore there is a distinction between *emphasis*, *highlighting* and *distinction*; and again distinction can be done on several levels (based on type, time, space, social characteristics).

Besides elements, TEI also defines entities. For example, here are some entities related to full stops:

`&stop.abbr;` entity used for an abbreviation dot;
`&stop.sent;` entity used for a sentence period;
`&stop.abse;` entity used for an abbreviation dot which is also an abbreviation period;
`&stop.dec;` a dot used as decimal separator;
`&comma.dec;` a comma used as decimal separator;
`&middot.dec;` a midline dot used as decimal separator;
`&stop.space;` a dot used as numeric space character;
`&comma.dec;` a comma used as numeric space character;

This list somehow shows the boundary between Unicode characters and XML (in this case, TEI) entities: in this case Unicode provides only characters with distinct glyphs: the first four entities have the same glyph[3] and hence correspond to the same Unicode character `U+002E FULL STOP`; entity `&middot.dec;` could be encoded by `U+00B7 MIDDLE DOT` but doing this we loose the specificity of `&middot.dec;` of being used only as a decimal separator—on the other hand, Unicode provides

[3]Although in English typography there is more space left after `&stop.sent;` than after `&stop.abbr;`.

character U+2027 HYPHENATION POINT which has the same glyph as &middot.dec; but is intended to represent possible hyphenations in dictionaries: TEI provides the element <hyph> whose rôle is to *contain a hyphenated form of a dictionary headword, or hyphenation information in some other form,* but has no special entity for displaying hyphenation points.

This example shows that often markup systems (like TEI) and encoding systems (Unicode) overlap, and a thorough study of both may be necessary before choosing between the methods they provide: in this case, entities or characters. Of course the safest approach is to always rely on the higher level method (structuring): by using customized entities one can prepare a document in a uniform and consistent manner; the expansion of such an ntity can then be one or more Unicode characters (or even, in the worst case, a series of processing instruction tags which will instruct software how to represent a given entity).

3.1 A real-life example

In the following we will describe the TEI structuring of the beginning of the critical edition displayed on fig. **??**, taken from [Chr] and typeset by Ω (see §5.4 for more details on how Ω typesets critical editions).

Quoting [wTdc, chap. 19], *scholarly editions of texts, especially texts of great antiquity or importance, often record some or all of the known variations among different witnesses to the text. Witnesses to a text may include authorial or other manuscripts, printed editions of the work, early translations, or quotations of a work in other texts. Information concerning variant readings of a text may be accumulated in highly structured form in a critical apparatus of variants.* The witnesses in our example are denoted by capital letters 'A,'-'X.' In the TEI file, they are described in a *witness list*, in the following way:

```
<witlist>
<witness sigil='A'>Atheniensis Bibl. nat. 211</witness>
<witness sigil='B'>Basileensis gr. 39 (B. II. 15)</witness>
<witness sigil='C'>Vaticanus gr. 560</witness>
<witness sigil='E'>Atheniensis Bibl. nat 265</witness>
<witness sigil='L'>Oxoniensis Bodl. Cromwell 20</witness>

<witness sigil='D'>Sinaïticus gr. 375</witness>
<witness sigil='G'>Laurentianus Conv. sopp. 198</witness>

<witness sigil='O'>Vaticanus gr. 577</witness>
<witness sigil='V'>Vaticanus gr. 1526</witness>
<witness sigil='X'>Genuensis Bibl. Franz. Miss. urb. gr. 11</witness>
</witlist>
```

Once these sigla defined, they can be used in the tags of the critical apparatus.

Here are the first two lines of text:

```
<p>Διὸ παραινῶ φεύγειν αὐτῶν τὴν μανίαν· μανίαν γὰρ ἔγωγέ φημι εἶναι ἐσχάτην τὸ φι-
λονεικεῖν εἰδέναι τί τὴν οὐσίαν ἐστὶν ὁ θεός...</p>
```

where the upper dot is encoded as U+0387 GREEK ANO TELEIA.

The first apparatus entry concerns the word μανίαν (actually the second occurence of this word on the first line of text). This word appears as μανίας in manuscripts E, L, D, G, O, V, X (which can be grouped as in the witness list: EL, DG, OVX). This information is written in the TEI document as follows:

```
<app from="1.1.1" to="1.1.2">
<lem>μανίαν</lem>
<rdg wit="EL DG OVX">μανίας</rgd>
</app>
```

where app stands for "critical apparatus entry," lem for "lemma," and rdg for "reading group." Attribute values 1.1.1 and 1.1.2 are references to tags (called "anchors") included in the XML source code of the main text, which now looks like

```
<p>Διὸ παραινῶ φεύγειν αὐτῶν τὴν μανίαν· <anchor id="1.1.1"/>μανίαν<anchor id="1.1.2"/>
γὰρ ἔγωγέ φημι εἶναι ἐσχάτην τὸ φιλονεικεῖν εἰδέναι τί τὴν οὐσίαν ἐστὶν ὁ Θεός...</p>
```

The anchor tags are "stand-alone" tag to avoid nesting problems.

Note that on fig. **??** the word μανίαν has number "2" as exponent. This comes from the fact that it is in fact the second occurence of this word on the same line. This property cannot be included in the structure of the document for the very simple reason that it depends entirely on the output: if the line was a few centimeters narrower the two words would be on different lines and the exponent would not be needed.

This kind of meta-information (multiple occurences, line numbers, etc.) can only be calculated at the very final stage of document processing; this is done by the processing software (here, Ω) and no extra information needs to be included in the XML document.

Let's take a look at the third entry of the critical apparatus:

```
<app from="1.2.1" to="1.2.2">
<lem wit="B">φημι</lem>
<rdg wit="E O">transp. post. ἐσχάτης</rgd>
<rdg wit="cett.">om.</rdg>
</app>
```

In this case the tag lem itself takes also the wit argument, since the word φημι as written in the text has been taken from manuscript B. For manuscripts E and O there is a different ordering of words: φημι comes after ἐσχάτης, and finally in all other manuscripts ("cett." meaning "codices ceteri") the word is omitted. Here we have the choice between giving the explicit list of other manuscripts, as in the list of witnesses above, or use this elegant expression, very frequent in this context (for a complete list of such expressions —at least in the frame of Belles Lettres editions— one can consult [Bel, p. 47]).

Using TEI one can build a so called "in-line" apparatus, in the sense that the app elements will appear inside the main text, like this:

```
<p>Διὸ παραινῶ φεύγειν αὐτῶν τὴν μανίαν· <anchor id="1.1.1"/>μανίαν<app from="1.1.1">
<lem>μανίαν</lem> <rdg wit="EL DG OVX">μανίας</rgd> </app> γὰρ ἔγωγέ φημι εἶναι
ἐσχάτην τὸ φιλονεικεῖν εἰδέναι τί τὴν οὐσίαν ἐστὶν ὁ Θεός...</p>
```

(in this case only the from attribute is necessary, since the app element is placed at the end of the lemma, in the base text).

It is also possible to build an external apparatus (in a different file), in which case one needs opening and closing anchors for each lemma in the base text.

On fig. **??**, underneath and to the right of the critical apparatus, one can find two blocks of (regular) footnotes, which "belong" to the translation of the text. The first block gives the references of Biblical quotations; the second block consists of the translator's comments. The notes of the first block

are numbered with lowercase letters, the ones of the second one, with Arabic numbers. To structure these notes, TEI provides an element called `note`. This element takes a great number of attributes; in our case the following would be sufficient:

```
<p>... je vous le montrerai à l'évidence par le témoignage des écrivains sacrés<note
type="regular">Le mot προφήτης a des nuances... et de Paul</note> : non seulement
ceux-ci ignorent manifestement... </p>
```

where we call the second block of notes "regular" ones (and the first one "biblical" ones).

Notes can also be placed in a separate file: in that case the text should contain anchors at the locations of footnote marks.

Note that in this example we have not explicitly tagged the word προφήτης as being Greek. This is the advantage of using Unicode: one can consider that whenever words are formed from characters in the Greek page of Unicode, then these words are in Greek and should be typeset accordingly (in particular, they should be hyphenated according to Greek hyphenation rules).

This works very well for the specific example (because we have a unique correspondence between the character set and the language) but would not work if, for example, in the same document we had also Coptic, because in Unicode, Greek and Coptic alphabets share the same character set. One should even distinguish between modern and ancient Greek, since these two are not hyphenated in the same way.

In the general case, we do recommend language tagging, by which the example above would be written as:

```
<p>... je vous le montrerai à l'évidence par le témoignage des écrivains sacrés<note
type="regular">Le mot <foreign lang="gr">προφήτης</foreign> a des nuances... et
de Paul</note> : non seulement ceux-ci ignorent manifestement... </p>
```

Finally, to close this section, here is how parallel texts are structured using TEI. There are two possible methods: either place "anchors" in both texts, referring to each other, or delimit "segments" of text, correponding to each other. Using the first approach, here is how the two texts look like:

```
<p><anchor id="gr1" corresp="fr1"/>Διὸ παραινῶ φεύγειν αὐτῶν τὴν μανίαν· μανίαν
γὰρ ἔγωγέ φημι εἶναι ἐσχάτην τὸ φιλονεικεῖν εἰδέναι τί τὴν οὐσίαν ἐστὶν ὁ Θεός. <an-
chor id="gr2" corresp="fr2"/>Καὶ ἵνα μάθῃς ὅτι μανίας ἐσχάτης τοῦτο, ἀπὸ τῶν προφη-
τῶν ὑμῖν τοῦτο ποιήσω φανερόν· <anchor id="gr3" corresp="fr3"/>οἱ γὰρ προφῆται οὐ
μόνον τί τὴν οὐσίαν ἐστὶν ἀγνοοῦντες φαίνονται, ἀλλὰ καὶ περὶ τῆς σοφίας αὐτοῦ πόση
τίς ἐστιν ἀποροῦσι· <anchor id="gr4" corresp="fr4"/>καίτοι γε οὐχ ἡ οὐσία ἀπὸ τῆς
σοφίας, ἀλλ᾽ ἡ σοφία ἐκ τῆς οὐσίας...</p>
```

and

```
<p><anchor id="fr1" corresp="gr1"/>Je vous exhorte donc à éviter leur folie, car
c'est le comble de la folie que s'acharner à connaître Dieu dans son essence. <an-
chor id="fr2" corresp="gr2"/>Et pour que vous compreniez que c'est bien en effet le
comble de la folie, je vous le montrerai à l'évidence par le témoignage des écri-
vains sacrés : <anchor id="fr3" corresp="gr3"/>non seulement ceux-ci ignorent ma-
nifestement ce qu'il est dans son essence, mais encore ils ne savent que dire de
l'étendue de sa sagesse ; <anchor id="fr4" corresp="gr4"/>or ce n'est pas l'essence
qui dérive de la sagesse, mais la sagesse de l'essence.
```

Of course the amount of such anchor tags depends entirely on the precision the author wishes to attain, when displaying and processing the parallel texts. Often it is sufficient to place them at major punctuation marks (periods, semicolons, etc.).

3.2 References for further reading on TEI

We recommend of course the TEI Guidelines, a monumental documentation (very well written and with many examples) which can be found on the Web ([wTdc]) or ordered in printed form from the TEI Consortium ([wTei]). The only disadvantage of these Guidelines are that they have written with SGML in mind (before XML was even invented).

For French language readers we also recommend the Cahier GUTenberg #24: *TEI : Text Encoding Initiative* [Gut$_2$] and [wGut] which contains also complete (translated) documentation of a lighter version of TEI, called "TEI Lite."

The main Web site of the TEI Consortium is [wTei].

For readers interested in structuring of text corpora, we also recommend the Web site of the CES project [wCes]: *CES specifies a minimal encoding level that corpora must achieve to be considered standardized in terms of descriptive representation (marking of structural and typographic information) as well as general architecture (so as to be maximally suited for use in a text database). It also provides encoding specifications for linguistic annotation, together with a data architecture for linguistic corpora.*

4 Unicode in a scholarly context

As an example of issues arising when using Unicode in a scholarly context, we will discuss some problems related to Greek and Arabic, in a scholarly context.

4.1 Greek

4.1.1 Should we use combining diacritics or precomposed characters?

Greek letters with diacritics (accents and breathings) can be encoded in several ways:

1. vowels with acute accent can be encoded in three different ways, for example to obtain an 'ά' one can use U+03AC GREEK SMALL LETTER ALPHA WITH TONOS or U+1F71 GREEK SMALL LETTER ALPHA WITH OXIA or U+03B1 GREEK SMALL LETTER ALPHA followed by U+0301 COMBINING ACUTE ACCENT;[4]
2. most of the vowels with diacritics can be encoded in two ways: precomposed (in the range 1f00-1ffc) or using combining diacritics;
3. some vowels and diacritics combinations can be obtained *only* by using combining diacritics, like for example 'ε̃', 'ο̃'.

There is even a case where the same letter with diacritics can be obtained in eight (!) different ways: iota with dieresis and acute accent 'ΐ' can be obtained

1. as a precomposed character U+0390 GREEK SMALL LETTER IOTA WITH DIALYTIKA AND TONOS;

[4] Historically the fact that there are two characters for the same glyph and same linguistic entity (03ac and 1f71) is due to the (false) assumption that the accent of monotonic Greek (*tónos*) and the acute accent of the polytonic system (*oxía*) are distinct.

2. as a precomposed character U+1FD3 GREEK SMALL LETTER IOTA WITH DIALYTIKA AND OXIA;
3. as U+03B9 GREEK SMALL LETTER IOTA followed by U+0344 COMBINING GREEK DIALYTIKA TONOS;
4. as U+03B9 GREEK SMALL LETTER IOTA followed by U+0344 COMBINING DIERESIS and then by U+0301 COMBINING ACUTE ACCENT;
5. as U+03CA GREEK SMALL LETTER IOTA WITH DIALYTIKA followed by U+0301 COMBINING ACUTE ACCENT;
6. as U+03AF GREEK SMALL LETTER IOTA WITH TONOS followed by U+0344 COMBINING DIERESIS;[5]
7. as U+1F77 GREEK SMALL LETTER IOTA WITH OXIA followed by U+0344 COMBINING DIERESIS
8. as U+03B9 GREEK SMALL LETTER IOTA followed by U+0301 COMBINING ACUTE ACCENT and then by U+0344 COMBINING DIERESIS;

This is of course an exceptional case, but still it shows that there is a problem in representing Greek letters with diacritics. Normalization can of course be done automatically by software, but (a) if this normalization is towards precomposed characters, then the software should be aware of the fact that some combinations do not exist as precomposed characters, (b) on the other hand, if normalization is towards decomposition, then additional rules should be specified for cases like ΐ or ΐ́ where both diacritics are in the same combining class and it is not clear which one is more "outwards" than the other.

The most serious problem arises when encoding capital Greek letters with diacritics. The problem is the following: from a typographical point of view, accents and breathings that belong to capital letters are placed *in front* of them (and not *upon* them). Unfortunately even historical encodings of Greek (like the "beta encoding" of the *Thesaurus Linguae Graecae*) use these diacritics in a prepositive way.

In the case of Unicode, it is not clear if these letters should be decomposed

1. in the same way as lowercase letters: 'Έ' should be encoded U+0395 GREEK CAPITAL LETTER EPSILON followed by U+0313 COMBINING COMMA ABOVE and U+0301 COMBINING ACUTE ACCENT;
2. as a blank space with diacritic(s) followed by the letter: U+0020 SPACE U+0313 COMBINING COMMA ABOVE U+0301 COMBINING ACUTE ACCENT U+0395 GREEK CAPITAL LETTER EPSILON;
3. using *spacing diacritics:* U+1FCE GREEK PSILI AND OXIA followed by U+0395 GREEK CAPITAL LETTER EPSILON.

These approaches correspond to different philosophies: the first one is logical and linguistically accurate: after all the fact that diacritics are written in front and not above the letter is purely a typographical convention (in earlier printings diacritics were indeed placed *above* or sometimes *after* capital letters); the second and third approaches are graphical ones [the second being minimalistic and the third using an obscure set of Unicode characters: Greek spacing diacritics[6]].

We recommend the first approach, not only because it is the most conformant to Greek grammar, but also because in some cases a distinction should be made between *diacritics belonging to (capital) letters but graphically placed in front of them* and *diacritics belonging to suppressed letters*; the latter

[5] Acute accent and dieresis belong to the same Unicode combining class 230, consequently the rule that applies is that *combining marks with the same combining class are generally positioned graphically outwards from the base character they modify* [Uni, §3.9]: but in this case it is not clear weither the acute accent or the dieresis are "outward" since graphically the acute accent is placed *between* the dots of dieresis.

[6] Obscure indeed because in the Unicode book it is said that they are used *in the representation of Polytonic Greek texts*, without any former comment. If this sentence should be interpreted as "they shall be used in front of capital letters," then what can possible be the rôle of U+1FEF GREEK VARIA? Every Greek capital letter is either a vowel (in which case it necessarily takes a breathing) or a consonant (in which case it either takes a breathing or no diacritic at all): it is not possible to have a stand-alone grave accent in front of a capital letter.

can be considered as "stand-alone" diacritics, and this could indeed be a nice application of some of the Unicode Greek spacing diacritics. Here is an example:

῾Ομως κ᾽ ἕνας σκύλος, νά ! ᾽βγαίνει ᾽πίσω ἀπ᾽ τὸ δρυμό,

This sentence starts with a capital letter with breathing and accent: U+039F GREEK CAPITAL LETTER OMICRON followed by U+0312 COMBINING REVERSED COMMA ABOVE and U+0301 COMBINING ACUTE ACCENT while in front of words βγαίνει and πίσω there is an special mark representing suppressed letters: as shown in [Yhg or wYhg, §1.3.1], this mark should be treated as a breathing and not as an apostrophe because in some cases it is indeed combined with an accent. In these cases we recommend the use of spacing diacritics: βγαίνει and πίσω should be preceded by U+1FBF GREEK PSILI.

It should be noted that in this sentence there are two more marks having exactly the same shape as the smooth breathing: the ones at the end of words κ᾽ and ἀπ᾽. These are indeed apostrophes and should be encoded as U+0027 APOSTROPHE; the fact that they have the same shape as U+1FBF GREEK PSILI, is a Greek typographical convention.

4.1.2 Levels of diacritization

Decomposing precomposed characters into regular characters and combining diacritics has another advantage: software can more easily treat strata of text representing different levels of diacritization. Indeed we can divide Greek diacritics into four categories:

1. *accents*: acute U+0301 COMBINING ACUTE ACCENT, grave U+0300 COMBINING GRAVE ACCENT, circumflex U+0342 COMBINING GREEK PERISPOMENI;
2. *breathings*: rough U+0314 COMBINING REVERSED COMMA ABOVE and smooth U+0313 COMBINING COMMA ABOVE;
3. *dieresis* U+0308 COMBINING DIAERESIS;
4. *syllable length*: macron U+0304 COMBINING MACRON, brevis U+0306 COMBINING BREVE.

Decomposing characters means obtaining regular characters followed by one, two or three combining diacritics, each one in a different category. Software could be instructed to consider or to ignore anyone of these categories: one could search, index, or compare words taking all categories under consideration (ὅρος boundary ≠ ὅρος mountain ≠ ὀρός serum), or ignoring breathings (ὅρος = ὅρος ≠ ὀρός), or ignoring accents (ὅρος ≠ ὅρος = ὀρός), or ignoring both accents and breathings (ὅρος = ὅρος = ὀρός), etc.

4.1.3 The mute iota

Unicode provides a combining diacritic U+0345 COMBINING GREEK YPOGEGRAMMENI, a spacing diacritic U+1FBE GREEK PROSGEGRAMMENI and a certain number of vowels with a subscript iota, like ᾳ᾽ U+1FB3 GREEK SMALL LETTER ALPHA WITH YPOGEGRAMMENI. In fact all of these are results of a single grammatical phenomenon: the *mute iota*. This is a iota which is generally not pronounced: τῷ Θεῷ is pronounced *tó theó* and not *tói theói*.

There are different ways of representing the same phenomenon: in Anglosaxon typography of Greek texts a mute iota is represented by a regular size iota; in Greek or French typography of Greek texts, the mute iota can be represented as a subscript, or something even more special: in a all-caps context it can be found in lowercase, or as a small cap, or in very small size, etc.

Encoding of a Greek text should be independent of typographical conventions ! Therefore we recommend the use of an entity &muteiota; which, according to the context, could be expanded as

a regular iota U+03B9 GREEK SMALL LETTER IOTA or as a combining diacritic U+0345 COMBINING GREEK YPOGEGRAMMENI, or something else, at the author's discretion.

To close this section on mute iota, we would like to remind the reader (see [Yhg or wYhg, §1.3.7]) that the "subscript iota" typographical construct has been used also for other purposes than the mute iota: in Osmanli Turksih it is combined with letter iota to represent a specific sound.

4.2 Arabic

In Unicode, Arabic letters and diacritics are clearly and unambiguously represented. Nevertheless, in a scholarly context, there may be alternative ways to encode Arabic texts.

The problem is that the way Arabic letters are encoded in Unicode does not always corresponds to Arabic grammar. We will examine a few cases of this phenomenon.

4.2.1 The letter hamza

Unicode provides five characters: U+0621 ARABIC LETTER HAMZA, U+0623 ARABIC LETTER ALEF WITH HAMZA ABOVE, U+0624 ARABIC LETTER WAW WITH HAMZA, U+0625 ARABIC LETTER ALEF WITH HAMZA BELOW and U+0626 ARABIC LETTER YEH WITH HAMZA ABOVE, which in fact are *simply variant graphical representations of the same letter hamza*.[7]

There are very strict rules determining which form this letter should take:[8]

1. at word begin:
   (a) if the short vowel following the hamza is kasra, then the form U+0625 ARABIC LETTER ALEF WITH HAMZA BELOW is used;
   (b) otherwise, the form U+0623 ARABIC LETTER ALEF WITH HAMZA ABOVE is used;
2. inside a word:
   (a) if the hamza is preceded or followed by a kasra then it takes the form U+0626 ARABIC LETTER YEH WITH HAMZA ABOVE;
   (b) if the hamza is preceded by a damma and followed by damma, fatḥa or sukun, then it takes the form U+0624 ARABIC LETTER WAW WITH HAMZA;
   (c) if the hamza is followed by a damma and preceded by damma, fatḥa or sukun, then it takes the form U+0624 ARABIC LETTER WAW WITH HAMZA;
   (d) if the hamza is preceded by a fatḥa and followed by fatḥa or sukun, then it takes the form U+0623 ARABIC LETTER ALEF WITH HAMZA ABOVE;
   (e) if the hamza is followed by a fatḥa and preceded by fatḥa or sukun, then it takes the form U+0623 ARABIC LETTER ALEF WITH HAMZA ABOVE;
   (f) if the hamza is preceded and followed by sukun, then it takes the form U+0621 ARABIC LETTER HAMZA;
3. at then end of a word:
   (a) if hamza is preceded by a fatḥa, then it takes the form U+0623 ARABIC LETTER ALEF WITH HAMZA ABOVE;

[7] Grammarians are divided weither hamza is, or is not, a letter of the Arabic alphabet. We will consider that the former is indeed the case.

[8] In these rules, by fatḥa, kasra, damma and sukun we mean respectively U+064E ARABIC FATHA, U+0650 ARABIC KASRA, U+064F ARABIC DAMMA and U+0652 ARABIC SUKUN.

(b) if hamza is preceded by a kasra, then it takes the form U+0626 ARABIC LETTER YEH WITH HAMZA ABOVE;

(c) if hamza is preceded by a damma, then it takes the form U+0624 ARABIC LETTER WAW WITH HAMZA;

(d) if hamza is preceded by a sukun, then it takes the form U+0621 ARABIC LETTER HAMZA.

Using these rules, software like Ω or ArabTEX ([wLag]) can unambiguously determine the graphical representation of letter hamza. This can be quite interesting in a scholarly context, where grammatical identity is more important than graphical representation. Under these conditions, we recommend the use of an entity &hamza; for letter hamza that can be graphically represented in different ways according to the context.

But to use such an entity (and hence greatly simplify Arabic keyboarding and processing) two conditions have to respected: (a) text has to be vocalized and (b) word prefixes must be marked as such.

Indeed, in Arabic, short vowels are not always written: there is a "least effort" principle that implies that short vowels are only written when necessary to avoid ambiguities, and often even that principle is not followed and vowels are simply omitted. Without the information on vowels, the correct form of letter hamza cannot be calculated by software.[9] On the other hand, one could very well imagine vowels included in the data, but not displayed: this would allow the calculation of hamza and still keep the non-vocalized output.

The second condition deals with prefixes: in Arabic, prepositions (like و "and," بـ "with," لـ "to, for" etc.) as well as the definite article are attached to the word. In the hamza rules we gaved above, "word begin" actually refers to the word *without* prefix. But software is not able to identify a prefix as such (without intensive morphological, syntactical and semantical analysis). Hence, to be able to calculate the hamza form correctly, one has to mark up word limits, so that prefixes can clearly be recognized. For example there is Unicode character U+200D ZERO WIDTH JOINER that could be used for this purpose [although this is not explicitly stated in the Unicode book [Uni]]. It is well understood that this additional character should not break the contextual analysis of the word.

This may seem extra work, but in fact can be very useful in the processing of Arabic text.

4.2.2 The definite article

The definite article in Arabic الـ has a special property: at the beginning of a sentence the letter alef is pronounced, while inside a sentence it is assimilated with the last letter of the previous word: بُ ابتكالا يرب كالا is pronounced *alkitaabulkabiir* and not *alkitaabu alkabiir*. To point out the assimilation of the alef, a special diacritic is used, called *wasla*. In Unicode, instead of using a combining diacritic, the wasla has been precombined with the underlying alef, as U+0671 ARABIC LETTER ALEF WASLA.

On the other hand, when the alef is indeed pronounced, one uses no diacritic (or sometimes a fatḥa short vowel). The same definite 'article + word' cluster will be written with or without a wasla depending on its position in the sentence.

Another problem arising because of the definite article is (as in the previous section) the fact that since it is attached to the word, software cannot easily distinguish the article from the word, for sorting, indexing, searching, and processing in general.

For these reasons, in a scholarly context, it could be useful to either insert a special character between the definite article and the word, as suggested in the previous section, or use an entity &al; to encode it.

[9] Unless of course this software has also a post-vocalisation module; this is not a simple task since it requires heavy morphological, syntactical and semantical analysis.

It should be noted that other words than the definite article also take a wasla: this phenomenon is called the "unstable hamza."

4.2.3 Nounation

The three combining diacritics U+064B ARABIC FATHATAN, U+064D ARABIC KASRATAN and U+064C ARABIC DAMMATAN are marks of a grammatical phenomenon called *nounation*. The fact is that while the last two are simply diacritics placed on the last letter of a word, fatḥatan produces also a letter U+0627 ARABIC LETTER ALEF (except when the word finishes with a U+0629 ARABIC TEH MARBUTA, a U+0627 ARABIC LETTER ALEF followed by U+0621 ARABIC LETTER HAMZA, or a U+0623 ARABIC LETTER ALEF WITH HAMZA), for example:

بَدَلُو + fatḥatan → بَدَلًا *but* قُرُبَ + fatḥatan → قُرُبًا.

The problem is that there is a typographical convention which consists in placing the fatḥatan *before* the mute alef (as in بَدَلًا) but this convention is by no means followed by all Arabic writers: very often one also sees the fatḥatan placed *upon* the alef (as in بدلا). Since Unicode encodes the graphical image and not the grammatical substance, these two would be encoded differently, which is absurd since the whole difference is only due to a typographical convention.

For this reason we recommend the use of an entity &fathatan; which would stand for both the combining diacritic U+064B ARABIC FATHATAN and the mute U+0627 ARABIC LETTER ALEF, whenever necessary.

4.2.4 Conclusion: a different way of encoding Arabic

By adopting the above suggestions as a whole, we obtain a new way of encoding the Arabic language.[10] In fact this goes even farther: if we decide to encode short and long vowels as grammatical objects, for example by using XML entities, then we would have the following representations of these entities in Unicode: &short_a; would be U+064E ARABIC FATHA, while &long_a; would be U+064E ARABIC FATHA followed by U+0627 ARABIC LETTER ALEF; &short_i;, would be U+0650 ARABIC KASRA while &long_i; would be U+0650 ARABIC KASRA followed by U+064A ARABIC LETTER YEH; and finally &short_u; would be U+064F ARABIC DAMMA while &long_u; would be U+064F ARABIC DAMMA followed by U+064A ARABIC LETTER WAW.

In this case, U+064A ARABIC LETTER YEH and U+064A ARABIC LETTER WAW represent long vowels, so they can never take short vowels themselves. A distinction has to be made between these and the semi-consonants represented by the same Unicode characters U+064A ARABIC LETTER YEH and U+064A ARABIC LETTER WAW: these can take short vowels and play different grammatical rôles than the long vowels. One could imagine entities &semi_consonant_yeh; and &semi_consonant_waw; for these semi-consonants.

This way of encoding Arabic is much closer to the grammar of the language and hence makes any kind of processing easier. The necessity of vocalization makes this method more appropriate to scholarly contexts and probably unsuitable for everyday unvocalized modern Arabic text. This encoding method has been firstly implemented in ArabTEX ([wLag]) and is used by ArabTEX users all around the world (Arabic being keyboarded in a 7-bit transcription). Recently it has been implemented also in the Ω system: in this case Arabic can be keyboarded in 7-bit transcription or in Unicode.

[10] Note the above is true only for the Arabic *language* and not for other languages using the Arabic *writing system*.

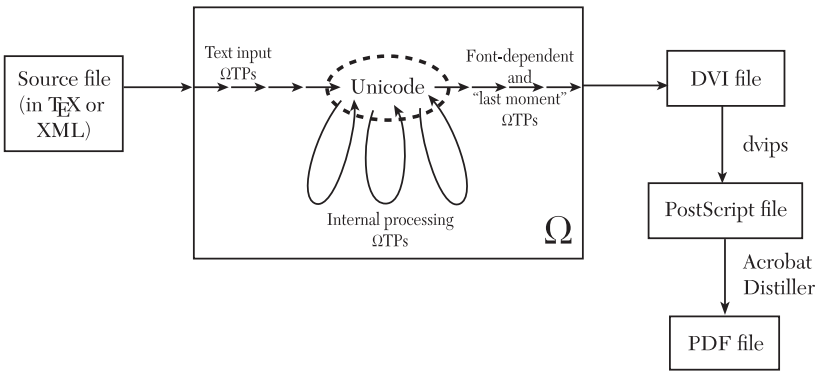

Figure 3: Schematic diagram of the Ω system.

5 Ω

The Ω system, developped by John Plaice and the author aims to produce typeset documents out of Unicode-encoded structured data (structured weither in XML or in the LaTeX markup system). Being an extension of TeX, Ω inherits its long list of typographical capabilities. The main innovation of Ω, with respect to TeX, is the notion of *Ω Translation Process* (ΩTP). These are *filters* which transform text as it is read by Ω. ΩTPs can be arbitrarily combined to form *filtering sequences*. These sequences can be activated or de-activated dynamically, according to processing instructions inserted in the text.

On fig. 3 the reader can see an overview of the Ω system. A first sequence of ΩTPs (§5.1) converts the text input into Unicode. The goal here is to attain the highest possible conformance to Unicode; if the input is already in Unicode then this step is skipped. Entities or processing instructions referring to characters not in Unicode are send to code points in the private area. Once in Unicode, a second sequence of ΩTPs (§5.2) applies all necessary linguistic transformations, while staying in Unicode, if necessary using the private zone. Finally, depending on the fonts used, a third sequence of ΩTPs (§5.3) sends the (eventually transformed) Unicode data to the font (or to adequate PostScript instructions).

Ω produces PostScript files which can be used for phototypesetting, or converted to PDF files. There is also the possibility of inserting special PostScript instructions in the output file, which will be used by Acrobat Distiller to insert hypertext links (and other meta-material provided by the PDF file format) into the PDF file: thru this method, Ω is able to produce online electronic documents with links, bookmarks, etc.

5.1 ΩTPs for text input

These ΩTPs convert text encoded in an arbitrary encoding into Unicode. At first sight, this step may seem useless since nowadays there are sufficiently powerful Unicode-compliant text editors which allow direct Unicode keyboarding of text. This is certainly true for ordinary text, but once again, in a scholarly context, these editors may still not be powerful enough. Let us suppose that you need to keyboard Arabic text, where some letters carry a special combining diacritic, for example a circumflex accent. This is perfectly valid in Unicode, and Ω is perfectly capable of typesetting such a construction, but text editors (even Unicode-compliant ones) will most probably break the contextual analysis of Arabic text. In such cases it is probably easier to work with a transcription, which will not cause any special problem to the text editor.

Also on some platforms (like the Macintosh) there are no Unicode-compliant text editors yet. In that case one must fight with encodings (as in the darks ages of 8-bit computing) and ΩTPs can be used to internally convert the text into Unicode, even if various parts of the text are written in different encodings: for example a file prepared on a word processor like Nisus Writer (on the Macintosh), using Latin, Arabic and Hebrew fonts, and saved as ordinary text, will contain binary data in the three encodings MacRoman, MacHebrew and MacArabic. If the user has properly tagged languages (using, for example, the `xml:lang` argument of XML elements) then the appropriate ΩTPs for each encoding will be automatically activated and de-activated.

Another very important use of this first sequence of ΩTPs is the conversion of *typewriter conventions* to abstract data. For example, in France there is a convention of leaving a blank space before double punctuation; this convention originates from the typewriter tradition and has its typographical counterpart: an interword space before a colon and a thin space before other double punctuation marks. This convention has no semantical value; it is even geographically restrained to metropolitan France and Belgium (neither in Canada, nor in Switzerland are these blank spaces used). For this reason it is more efficient, when structuring French text, to use XML entities for double punctuation. But this is not always the case (after all, text is more readable without entities); in a French context, Ω will detect blank spaces before double punctuation, swallow them, and replace punctuation marks by processing instructions (which can then be configured according to metropolitan French or Canadian standards, etc.).

A more sophisticated use of these ΩTPs is obtained when converting text written in "grammatically encoded Arabic," (see §4.2.4) into Unicode: four quite complicated ΩTPs had to be written, applying all the necessary rules of Arabic grammar (form of hamza, nounation, short vowels in front of long ones, wasla upon alif on the definite article, etc.).

Finally, these ΩTPs can be very useful for processing text in writing systems not yet provided by Unicode, like Egyptian hieroglyphics. The Centre for Computer-aided Egyptological Research (Utrecht University, [wCer]) distributes special WYSISYG editors (for Windows and Macintosh) which allow easy keyboarding of about 5,500 hieroglyphs (single, or combined in horizontal or vertical clusters, cartouches, etc.). These editors save the data in ASCII text files, where hieroglyphs are coded in a 7-bit transcription scheme. Ω can read these files, internally process the data in Unicode's private zone, and typeset the result using high quality fonts (also provided by the Center).

5.2 ΩTPs for internal processing

These correspond to various linguistic transformations, as already shown on fig. 1. They can be of different kinds:

5.2.1 Change of case

Between the world's writing systems there are seven (Latin, Greek, Coptic, Cyrillic, Glagolitic, Armenian, Georgian Khutsuri) whose letters have the "case" property (see [Uni, §4.2]). Unicode mentions three cases: *upper case*, *lower case* and *title case* (also called "all caps"). These are the three cases that may need different glyphs, and hence different Unicode characters. But when structuring document one may need other, more subtle, cases.

For example one could use two kinds of upper case: *contextual* and *mandatory*: in the sentence

Last week I saw John.

the 'L' of 'Last' is set in upper case because it is the first word of the sentence: this is contextual upper case; the word 'I' and the letter 'J' in 'John' are mandatory upper cases: they will be uppercase in all

contexts.

Another possible case is the *antithetic* one: in the German word 'StudentInnen' the 'I' is represented by an upper case glyphs because the surrounding letters are lowercase; the same word, in title case would be written 'STUDENTiNNEN': the letter 'I' must always be the opposite case as its surrounding.

Changing case is not always a simple match between Unicode characters: the Unicode book states the Turkish example 'i → İ' and 'ı → I'; there are more complicated examples in Greek: while normally 'ά' is sent to 'A' and 'ι' to 'I', the couple 'άι' will be sent to 'AΪ' (because in fact 'άι' denotes 'άϊ' and the dieresis is not written because the accent on the previous letter indicates that the diphthong is broken). Another example: in German, although 'ß' is normally capitalized as 'SS' in some cases it is capitalized as 'SZ' (for example 'Masse → MASSE' while 'Maße → MASZE.'

Ω has casing algorithms which depend on the writing system, the language and the context of the word or of the letter, as illustrated by the examples.

5.2.2 Basic micro-typography

Every language has its own typographical conventions, weither these are precisely described in (more-or-less) official manuals, or are part of the typographer's unwritten knowledge and craftmanship. Most of these rules are related to punctuation and spacing. Unicode, although being an (abstract) information exchange encoding, provides many characters which are of purely typographical nature and can quite efficiently represent at least the basic typographical conventions of most languages. For example the concept of opening and closing double quotes (which, structurally, could correspond to TEI tags <quote> and </quote>) will be represented by characters U+201C LEFT DOUBLE QUOTATION MARK and U+201D RIGHT DOUBLE QUOTATION MARK in English, or U+201E DOUBLE LOW-9 QUOTATION MARK and U+201C LEFT DOUBLE QUOTATION MARK in German, or U+00AB LEFT POINTING DOUBLE ANGLE QUOTATION MARK and U+00BB RIGHT POINTING DOUBLE ANGLE QUOTATION MARK in Canadian French. While in most languages double punctuation is attached to the preceding word, in French a thin space (U+2009 THIN SPACE) is placed between a word and either a semicolon, exclamation mark or question mark.

These transformations are characterized by two facts: (a) they are strictly Unicode-internal, and (b) they are font-independent. Ω uses ΩTPs to perform these transformations, depending on the currently active language.

5.2.3 Hyphenation

In a forthcoming version of Ω, hyphenation will be handled by one or more ΩTPs at the Unicode level. There are many contradicting facts which make hyphenation more complicated than any other linguistic transformation. First of all hyphenation is neither input-encoding- nor font-dependent, so logically it should be done at this level, where data is in pure Unicode. The transformation would consist in adding special characters (for example U+200B ZERO WIDTH SPACE or some other internal character) at places where hyphenation is allowed. But these inserted characters should no affect other transformations, weither these are linguistic or typographical: for example, contextual properties should remain the same, if not hyphenated, letters should retain their natural kerning, etc.

In fact hyphenation goes farther than that: it happens in several languages that words change when hyphenated. For example, in German, "backen" will be hyphenated "bak-ken," in Greek, "Μαΐου" will be hyphenated "Μα-ΐου" and in Armenian, "գառնուկ" will be hyphenated "գառն-ուկ." Ω has to perform these changes, without affecting the visual image of the word when it is not hyphenated.

Finally another important issue when hyphenating is the choice of the best hyphenation for a given word: for example in the German word 'Herausforderung' we can classify possible hyphenations as following: 'He^{-1}raus^{2}for^{1}de^{1}rung' where '2' means the ideal hyphenation (between components of the composite word), the '1's mean allowed but less preferable hyphenations, and '−1' stands for a hyphenation normally not allowed, but still exceptionally possible. Hyphenation algorithms should be subtle enough to ponderate their choices using such a scale.

5.2.4 Morphological analysis

For about five centuries the German language was printed in broken scripts: Fraktur, Schwabacher, etc. These have been abolished by the Nazis in 1943, and ever since no German authority ever came back to the Nazi decision. Today many broken script typefaces are available for the computer (see for example [wFra]). But in fact typesetting German in a broken script requires more than just changing the output font. Broken scripts have two variant of letter 's': the final/isolated one 'ſ' (which corresponds to the Latin 's' as used today) and the initial/medial one 'ſ' which is encoded by Unicode character U+017F LATIN SMALL LETTER LONG S. The problem is that the choice between the two 's' letters is purely linguistic: the word 'Eiſen' will be written with a long 's' because it is a simple word, while 'Eisberg' will be written with a short one because it comes from the composition of 'Eis' and 'Berg'. In some case components cannot be directly identifiable: 'Island' will be written with a short 's' although there is no word 'Is' (with the meaning of "ice," as in 'Iceland').

Another similar phenomenon (which this time is valid for German typeset both in broken or Roman script) is the breaking of ligatures (for example the 'ﬂ' ligature being written as two letters 'fl') between components of a composite word: compare 'Auflage' and 'Querflöte.' Once again the decision weither a ligature should be broken or not depends on morphology of the German language.

The author has developped a tool to perform these transformations (long/short 's' and ligatures), based on work by Oliver Lorenz who has developped DMM: a morphological analyzer of the German language, with a Perl API [wDmm]. For the moment this system has a success rate of about 95-97% and this is why the author uses it rather as a pre-processor than as an ΩTP (to be able to check and manually correct the result, before typesetting).

Well understood the most elegant solution would be to include this kind of information already in the XML file, using either Unicode characters U+017F LATIN SMALL LETTER LONG S and U+200C ZERO WIDTH NON-JOINER, or XML entities &longs; and &nolig; which could be expanded differently weither the text is to be displayed in broken or Roman script.

5.2.5 Contextual analysis

Writing systems like the Arabic, Syriac and Mongolian one, have contextual analysis mechanisms: letters change forms according to their context in the word, or according to semantic criteria. Unicode encodes the linguistic units and not their forms. In the Unicode book, the so-called *isolated* forms of letters are displayed, but each Unicode character in fact stands for all forms of the same letter. In some cases one sees the same glyph for different Unicode characters: this happens because the isolated forms are the same, but the contextual behaviour is different (in other words, some other, non-displayed, forms differ). This is the case, for example, for characters U+0649 ARABIC LETTER ALEF MAKSURA and U+06CC ARABIC LETTER FARSI YEH (the former has only two distinct forms while the latter, which is used in Persian, has four distinct forms).

Sometimes rules of contextual analysis are transgressed for semantical reasons. For example, in Arabic the letter U+0647 ARABIC LETTER HEH whose isolated form is ه, is sometimes written in ini-

tial form ﻫ as an abbreviation for ﻫ.ﺟ = A.H. (the Muslim year). To encode such exceptions, Unicode has two special characters: U+200C ZERO WIDTH NON-JOINER and U+200D ZERO WIDTH JOINER. Combining character U+0647 ARABIC LETTER HEH with the latter produces the initial form of the letter.

Contextual rules for these languages are not hard to implement: the only difficulty lies in the fact that combining characters, in any quantity, are not supposed to break the contexual behaviour of letters. Ω uses ΩTPs to calculate the forms of characters. This is not really a Unicode-internal transformation since Ω uses the private zone as a temporary recipient for Unicode characters with contextual information. Nevertheless we classify this kind of ΩTPs in this section because they are strictly script- and language-dependent: contextual analysis does not depend neither on the input encoding, nor on the font.

5.3 Font-dependent ΩTPs

Fonts can have different encodings: these ΩTPs will remap characters from Unicode to their positions in font encodings. The fact that ΩTPs can be dynamically activated and de-activated allows us to use fonts with different encodings in the same document. Often character representations have to be constructed by combining several glyphs (for example, in the case of accented letters). Sometimes there are extra ligatures in fonts; we call them *esthetic* ligatures, because they are not mandatory on a grammatical level, but rather typographical enhancements.

One encounters the most important case of such ligatures in a certain number of Arabic fonts: while there is only one mandatory Arabic ligature (لا) and one semi-mandatory (= God), such fonts can have thousands of additional esthetic ligatures. A few of these can be seen in the "Arabic Presentation Forms-A" Unicode zone; nevertheless we discourage the use of this zone because the set of such ligatures available in a font is very variable and because we believe that these characters do not comply to the Unicode design principles.

5.3.1 "Last moment" ΩTPs

The rôle of these ΩTPs is to introduce changes *after* all linguistic, typographical and font-dependent transformations have been done. For example, let us take the case of an Arabic grammar book where roots of words are set in black and the various prefixes, suffixes and infixes are set in some other colour (for example, blue) so that grammatical transformations are easier to understand. To change colour, one uses low-level PostScript code which must be inserted between letters *after* all other transformations (input and font remapping, contextual analysis, etc.) have been performed.

5.4 Using Ω as a batch program to typeset parallel texts and critical editions

Like TEX, Ω typesets by transforming the source file in a file containing the description of the printed page (weither in DVI, which is a file-format specific to TEX and Ω, or in PostScript or PDF). In an environment like Unix a script can launch Ω, extract information from the resultsing file and slightly modify the source file before launching Ω again. By this iterative process one is able to typeset document containing several flows of interrelated text, like the critical edition on fig. **??** where (a) the Greek text has to be kept parallel to the French translation, (b) the Greek text, as it advances, produces critical apparatus entries, (c) the French text, as it advances produces two bodies of footnotes, (d) Greek is on the left side, French on the right side, critical apparatus and footnotes fill the space underneath and on both sides and the last footnote can continue on the next page.

Using such a method (instead of a WYSIWYG system) has the advantage of offering total control upon the management of parallel text: in some cases, because of page constraints, it is more efficient to slightly override parallelism. The system measures the blank space between the main text and critical apparati/footnotes as well as the precision of parallelism (which of course depends on the number and accuracy of the anchor elements placed in both texts) and chooses the optimal solution through a rating system. It is even possible to instruct the system to convert some footnotes into endnotes (numbered differently) if this is going to improve considerably the result.

Whenever a critical apparatus entry is to be typeset, Ω (or rather this supra-Ω script) has to check if there is another occurence of the same text string as the lemma on the same line. For this, first of all, one has to define precisely what is meant by "same text string": indeed, depending on the publisher's specifications, strings with different capitalizations or different accentuations can be considered to be the same or not: in other words, if we consider the various transformations of text described above, the strings are the same modulo one or more of these transformations; also, if we consider accentuation to be one of the different strata of the document, then strings can be equal if one or more of these strata are ignored. The script has to check if two occurences of the same string occur on the same line: this can happen only after the second iteration of Ω, when the exact contents of each line (and words belonging to a pair of lines) are known; the check has to be re-iterated as well because any intermediate modification in the source code can change the whole paragraph.

A future version of Ω will perform these iterations internally; for the moment one needs a powerful machine to run Ω several times for each page of the critical edition.

5.5 References for further reading on XML

Ω is an extension of TEX. If the reader is not familiar with TEX, then we recommend the "TEX Bible," namely Donald Knuth's *The TEXbook* [Knu] as a start. There have been articles written on Ω, we would suggest the following: [Pla], [Yht] and [Yhn], all of them available also on the [wYhg] Web site.

Greek language readers may find some interest in reading [Yha], which presents Ω in the general framework of Greek typography.

6 Conclusion

Preparing scholarly documents using tools like XML, TEI, Ω —all of them involving Unicode— can be a complex process. In this paper we have attempted to give a general introduction to these tools and to mention a certain number of (at first sight mutually unrelated) issues arising in this process. Most of our examples were taken from Greek and Arabic: two languages with an unusually high degree of complexity and frequently used in scholarly documents. We hope that this paper will incite readers to use these tools and help them in efficiently producing high standard scholarly documents.

Bibliographical References

[Arm] Feydit, Frédéric *Manuel de langue arménienne*, Éditions Klincksieck, Paris, 1969.

[Bel] *Règles et recommandations pour les éditions critiques (Série grecque)*, Les belles lettres, Paris, 1972.

[Bra] Bradley, Neil *The XML Companion*, Addison-Wesley, Reading, 1998.

[Eck] Eckstein, Robert *XML Pocket Reference*, O'Reilly, Sebastopol, 1999.

[Gut] *Actes du colloque GUT'99, seconde partie : XML*, Cahiers GUTenberg 33–34, Paris, 1999.

[Gut$_2$] *Actes du colloque GUT'99, seconde partie : XML*, Cahiers GUTenberg 33–34, Paris, 1999.

[Knu] Knuth, Donald E. *The TEXbook*, Computer and Typesetting, vol. A, Addison-Wesley, Reading, 1984.

[Pla] Plaice, John and Yannis Haralambous *The Design and Use of a Multiple-Alphabet Font with Omega*, Proceedings of the Electronic Publishing Conference, Springer Lecture Notes in Computer Science 1375, 1998.

[Spe] Spencer, Paul *XML Design and Implementation*, Wrox Press Ltd., Birmingham, 1999.

[Uni] The Unicode Consortium, *The Unicode Standard, Version 2.0*, Addison-Wesley, Reading, 1996.

[Yha] Χαραλάμπους, Γιάννης *Ο σύγχρονος τυπογράφος*, ΗΥΦΕΝ, τόμος 2, Θεσσαλονίκη, 1999.

[Yhg] Haralambous, Yannis *From Unicode to Typography, a Case Study: Greek*, Proceedings of the 14th Unicode Conference, Boston, 1999.

[Yhn] Haralambous, Yannis, John Plaice and Johannes Braams *Never again active characters*, TUGboat 16 (4), 1995.

[Yht] Haralambous, Yannis and John Plaice *Typesetting with Omega, a Case Study: Arabic* Proceedings of the International Symposium on Multilingual Information Processing, Tsukuba (Japan), 1997.

References of Examples

[Ara] Aragon, *Le paysan de Paris*, folio Gallimard, Paris, 1972.

[Chr] Jean Chrysostome, *Sur l'incompréhensibilité de Dieu (Περὶ ἀκαταλήπτου)*, Sources chrétiennes, Le Cerf, Paris, 1970.

[Kor] *Ἀδαμαντίου Κοραῆ τὰ μετὰ θάνατον εὑρεθέντα συγγραμάτια, βουλῇ μὲν καὶ δαπάνῃ τῆς ἐν Μασσαλίᾳ κεντρικῆς ἐπιτροπῆς Κοραῆ ἐπιμέλειᾳ δὲ Ἀνδρέου Ζ. Μαμούκα συλλεγέντα τε καὶ ἐκδιδόμενα*, Τυπογραφεῖον Ἀδ. Περρῆ, ἐν Ἀθήναις, 1881 (reprinted by Μορφωτικὸν Ἵδρυμα Ἐθνικῆς Τραπέζης, Ἀθήνα, 1989).

References on the World Wide Web

[wAte] http://www.fluxus-virus.com

[wCer] http://www.ccer.ggl.ruu.nl/ccer/

[wCes] http://www.cs.vassar.edu/CES/

[wCmw] http://www.w3.org/TR/charmod

[wDmm] http://faui8l.informatik.uni-erlangen.de/IMMD8/Services/lt/datacollections/dmm-malagalexiconandmorphologicalrules.html

[wFra] http://www.fraktur.com

[wGut] http://www.gutenberg.eu.org/pub/GUTenberg/publications/

[wLag] http://www.informatik.uni-stuttgart.de/ifi/bs/lagally/arabrep/1.html

[wOas] http://www.oasis-open.org/cover

[wOme] http://www.ens.fr/omega

[wPiz] http://www.uic.edu/orgs/tei/pizza.html

[wSpy] http://www.icon-is.com

[wTei] http://www.tei-c.org

[wTdc] http://www.hcu.ox.ac.uk/TEI/Guidelines

[wUtr] http://www.unicode.org/unicode/reports/tr20

[wYhg] http://www.fluxus-virus.com/en/research.html

Colophon

This article has been typeset at the Atelier Fluxus Virus [wAte] using the Ω system (by John Plaice and the author) [wOme] on a Red Hat Linux system. The fonts used are Adobe New Caledonia for the Latin text, Bitstream Unicode Monospace for the source code, Monotype Naskh for the Arabic text and Monotype Greek 90, 91 and 92 for the Greek text.