Achtung!

Dies ist eine Internet-Sonderausgabe des Aufsatzes
„Linguistic documentation and the encoding of textual materials“
von Jost Gippert (2005).
Sie sollte nicht zitiert werden. Zitate sind der Originalausgabe in
Essentials of Language Documentation,
edited by Jost Gippert, Nikolaus P. Himmelmann, Ulrike Mosel,
Berlin / New York: de Gruyter 2006 (Trends in Linguistics, 178), 337-361
(chapter 14)
zu entnehmen.

Attention!

This is a special internet edition of the article
“Linguistic documentation and the encoding of textual materials”
by Jost Gippert (2005).
It should not be quoted as such. For quotations, please refer to the original edition in
Essentials of Language Documentation,
edited by Jost Gippert, Nikolaus P. Himmelmann, Ulrike Mosel,
Berlin / New York: de Gruyter 2006 (Trends in Linguistics, 178), 337-361
(chapter 14).

Linguistic documentation and the encoding of textual materials

Jost Gippert

Introduction

In the documentation of languages, the notation of textual materials in written form has always played a significant role, even after the development of audiovisual means of storage. The digital age has brought about but a minor change in this respect in that we can now expect our written data to be usable by many people and for many centuries without necessarily being printed and distributed as books. To reach this aim, a few preliminaries must be kept in mind, however, which will be addressed in this chapter.

Writing down textual materials in digital form is different from using a pencil and a sheet of paper as it presupposes the adaptation of clearly defined **codes** in a twofold sense: the encoding of characters, i.e., of the letters in the words to be written down, and the encoding of the elements of textual structure, i.e., of headlines, examples, vocabulary lists, etc. Both kinds of encoding are crucial for the exchange of data with other people: A future user who has no information on what encoding schemes you may have applied will probably have great difficulties in trying to re-decode (and read) what you wrote – in the worst case, your data will be totally irretrievable. In the following pages, I shall briefly explain why this is to be expected and what can be done to avoid it. We will start with the encoding of the smallest units of text, i.e. characters, and proceed to larger elements such as words, phrases, and syntagms. Other types of encoding that may be at issue here (esp. file encoding) will be addressed en passant.

1. The Encoding of Characters: From 7-Bit to 32-Bit

1.1. Mainframe computers: The ASCII age

In all modern digital equipment, the encoding of characters is based on a given set of correspondances of characters with numerical values, every

character being represented by one unique value. To encode the two times 26 letters (lower and upper case) of the Latin alphabet plus the digits from 0 to 9, the punctuation marks, parentheses and the like, a set of less than 100 unique values is necessary, and this is why the „stone age" mainframe computers of the 1960s to 1970s were based on a so-called 7-bit encoding: With 7 bits, 2^7= 128 characters can be encoded uniquely. The most popular standard developed on this basis is the so-called ASCII standard ("American Standard Code for Information Interchange"), cf. Table 1.

Table 1. Standardized 7-bit encoding (ASCII)

	0										1									
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
000																				
020														!	"	#	$	%	&	'
040	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;
060	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
080	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~													
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
	0										1									

It is clear that on the basis of this encoding scheme, English texts could easily be digitized, but German, French, or Spanish texts could not, let alone Greek, Russian, or Chinese texts in their original scripts. This does not mean, however, that it was impossible then to process texts in "exotic" languages. What was necessary was the invention of encoding schemes that used more than one digital unit to represent certain characters. Cp. Table 2 which shows the 7-bit adaptation of a Sanskrit text, the Rigveda, which was produced in the 1970s on a mainframe computer, with the "traditional" transcription added for comparison. It is clear that this encoding had at least two disadvantages: it was hardly possible to visualize the text as it should be on a computer screen, which resulted in lots of inputting errors, and the encoding was not transparent (or "self-explaining") in the sense that the individual items (letters, diacritics, accent marks) could have been easily

determined by somebody who was not involved in the encoding process themselves. It is true that this encoding met the condition of being consistent in that a given sequence of codes always represented the same character, and this is why these texts can be used and analyzed even today. Nevertheless, it was too clumsy to be maintainable for a longer period.

Table 2. Non-standard 7-bit encoding (Rigveda 7,1)

```
R700123011  AGNI!M+ NA!RO DI:!D)ITIB)IR ARA!N\YOR HA!STACYUTI:
            JANAYANTA PRAS=ASTA
R700123012  !M / DU:RED9!S=AM+ G9HA!PATIM AT)ARYU!M
R700123021  TA!M AGNI!M A!STE VA!SAVO NY 9&N\VAN SUPRATICA!KS\AM
            A!VASE KU!TAS= CI
R700123022  T / DAKS\A:!YYO YO! DA!MA A:!SA NI!TYAH-
R700123031  PRE!DD)O AGNE DI:DIHI PURO! NO! 'JASRAYA: SU:RMYA:&
            YAVIS\T\)A / TVA:!
R700123032  M+ S=A!S=VANTA U!PA YANTI VA:!JA:H-
```

1	*agníṃ náro dī́dhitibhir aráṇyor hástacyutī janayanta praśastám / dūredṛ́śaṃ gṛhápatim atharyúm*
2	*tám agním áste vásavo ny ṛ̀ṇvan supraticákṣam ávase kútaś cit / dakṣā́yyo yó dáma ā́sa nítyaḥ*
3	*préddho agne dīdihi puró nó 'jasrayā́ sūrmyā̀ yaviṣṭha / tvā́ṃ śáśvanta úpa yanti vā́jāḥ*

1.2. PCs, Macs, DOS, and MS Windows: 8-bit based standards and non-standards

With the extension of the ASCII encoding basis to 8 bits, this problem was at least partially overcome. On an 8-bit (= 1-byte) basis, $2^8 = 256$ characters can be encoded uniquely, and since the early 1980's, many 8-bit encoding schemes were developed and applied, adding „special" characters such as those representing the German "umlaut vowels" *ä, ö, ü,* the accented vowels *é, à, ô* etc. of French, or the Spanish palatal nasal, *ñ*, to the inventory. Unfortunately, this was not done in an equal, "standardized" way right from the beginning; instead, several leading computer companies developed their own individual schemes, which resulted in serious problems whenever data were to be exchanged between systems. Compare Tables 3-5 which show the encoding systems used in IBM / DOS computers, Mac computers, and MS Windows – only the latter one is more or less identical with the 8-bit

Table 3. Non-standard 8-bit encoding ("DOS/IBM", "Extended ASCII", "Code-page 437")

	0										1								
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
000		☺	☻	♥	♦	♣	♠	·	◘	○	◙	♂	♀	♪	♫	☼	►	◄	↕	‼
020	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼		!	"	#	$	%	&	'
040	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;
060	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
080	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	⌂	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï
140	î	ì	Ä	Å	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	Pts	ƒ
160	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»	░	▒	▓	|
180	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐	└	┴	┬	├	─	┼	╞	╟
200	╚	╔	╩	╦	╠	═	╬	╧	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█
220	▄	▌	▐	▀	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	∅	∈	∩
240	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■					
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
						0										1				

Table 4. Non-standard 8-bit encoding (MAC OS)

	0										1								
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
000																				
020														!	„	#	$	%	&	´
040	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;
060	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
080	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~		Ä	Å	Ç	É	Ñ	Ö	Ü	á	à	â	ä	ã
140	å	ç	é	è	ê	ë	í	ì	î	ï	ñ	ó	ò	ô	ö	õ	ú	ù	û	ü
160	†	°	¢	£	§	·	¶	ß	®	©	™	´	¨	≠	Æ	Ø	∞	±	≤	≥
180	¥	µ	∂	Σ	Π	π	∫	ª	º	Ω	æ	ø	¿	¡	¬	√	ƒ	≈	Δ	«
200	»	…		À	Ã	Õ	Œ	œ	–	—	"	"	'	'	÷	◊	ÿ	Ÿ	⁄	¤
220	‹	›	fi	fl	‡	•	‚	„	‰	Â	Ê	Á	Ë	È	Í	Î	Ï	Ì	Ó	Ô
240		Ò	Ú	Û	Ù	ı	ˆ	˜	¯	˘	˙	˚	¸	˝	˛	ˇ				
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
						0										1				

standard used in web environments up til now, the ANSI standard ("American National Standards Institute") also known as ISO standard no. 8859-1 (the special MS-Windows characters are displayed on a grey background within Table 5).

Table 5. Standardized 8-bit encoding (ANSI, ISO-8859-1, MS-Windows, Codepage 1252)

	0										1								
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
000																				
020														!	„	#	$	%	&	´
040	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;
060	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
080	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~				,	ƒ	„	…	†	‡	ˆ	‰	Š	‹
140	Œ					‘	’	“	”	○	—	–	˜	™	š	›	œ			Ÿ
160		¡	¢	£	☼	¥	|	§	¨	©	ª	«	¬	-	®	¯	°	±	²	²
180	’	µ	¶	-	¸	¹	º	»	¼	½	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç
200	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û
220	Ü	Ý	Þ	ß	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
240	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ				
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
					0										1					

Still, these encoding systems were not sufficient for the immediate encoding of other scripts such as Greek, Cyrillic, or Chinese. This is why from the middle of the 1980s on, so-called "code pages" were developed for 8-bit based computers, in which, just as in the examples shown above, the "upper" area exceeding the basic ASCII plain (values above 128) was used to encode various other character sets. Some of these code pages have been standardized within the ISO standard 8859; cf., e.g., Table 6 contrasting the Cyrillic code page ISO 8859-5 with the ANSI standard, ISO 8859-1.

Table 6 a/b. Standardized 8-bit mapping: ISO-8859-1 vs. ISO-8859-5

a.

32	!	„	#	$	%	&	'	(	)	*	+	,	-	.	/	47
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	63
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	79
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	95
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	111
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		127

160	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	®	¯		175
176	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	191
192	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	207
208	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	223
224	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	239
240	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	255

b.

32	!	„	#	$	%	&	'	(	)	*	+	,	-	.	/	47
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	63
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	79
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	95
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	111
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		127

160	Ё	Ђ	Ѓ	Є	S	І	Ї	J	Љ	Њ	Ћ	Ќ	·	Ў	Џ		175
176	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	191
192	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	207
208	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	223
224	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	239
240	№	ё	ђ	ѓ	є	s	і	ї	j	љ	њ	ћ	ќ	§	ў	џ	255

Apart from these "official" extensions, an unknown amount of local or even personal 8-bit encoding systems were developed since the early 1980s to meet the needs of languages and linguists. As a matter of fact, whenever somebody developed and applied a certain font the encoding of which did not match one of the standardized code pages, a new encoding system was created from scratch. Applying the method of "font mapping", we could thus meet, e.g., the requirements of Ancient ("Polytonic") Greek to be noted in original characters as well as Iranian languages to be rendered in a scholarly Latin transcription (cp. Tables 7-8).

The problem about all this is that whenever "font mapping" is applied, the basic requirement of documentation, viz. the persistence and recoverability of data, cannot be guaranteed as there is no unique one-to-one-relation between the character to be encoded and a given digitized value. If, e.g., we applied the Greek 8-bit font illustrated in Table 8, the value of 231 would represent a Greek lower case letter *pi* (π); the same value would stand for a Cyrillic *cha* (ч), however, if we used a font matching the standard codepage ISO 8859-5, and it would represent a Latin *c* with cedilla (ç) if we used the plain ANSI standard. This means that whenever an 8-bit encoding is applied in the encoding of textual materials, additional information must be stored as to what code page or font encoding is valid for a given character – this information, however, is not encodable as such in a

Table 7. Non-standard 8-bit encoding: Ancient ("polytonic") Greek

	0										1									
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
000		᾿	῎	῍	῾	῞	῝	.		˚										
020		§			῏	῟	´	`	´ͅ	Ϝ	ϟ	ϡ		!	"	ᾔ	ῄ	ῂ	ῇ	'
040	(	)	*	†	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;
060	ή	ἧ	ὴ	?	ς	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
080	P	Q	R	S	T	U	V	W	X	Y	Z	[	ᾐ	]	ἢ	·	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
120	x	y	z	ἡ	|	ἠ	ῆ	ἅ	ἄ	ü	έ	ἂ	ä	ὰ	ᾶ	ῖ	ἒ	ἔ	ὲ	ϊ
140	ἲ	ὶ	Ä	ἴ	ὅ	ὕ	ἕ	ὂ	ö	ὸ	ὒ	ὺ	ϋ	Ö	Ü	ἃ	ἓ	ἳ	ὃ	ὓ
160	ά	ί	ό	ύ	ῴ	ῲ	ῷ	ᾠ	ᾡ	ᾥ	ᾧ	ᾦ	ἵ	ὕ	ἇ	ᾑ	ᾕ	Γ	Δ	ᾗ
180	ᾔ	ᾗ	Θ	ώ	ὼ	Λ	ῶ	ὠ	Ξ	ὡ	Π	ὥ	Σ	ὢ	ὧ	Φ	ὤ	Ψ	Ω	ᾷ
200	ἶ	ὖ	ᾴ	ᾄ	ἥ	ἣ	ἦ	ἀ	ἐ	ἰ	ὀ	ὐ	ᾴ	ᾄ	ἆ	α	ὥ	γ	δ	ε
220	ζ	η	ϑ	ι	ϰ	β	λ	μ	ν	ξ	ὦ	π	ϱ	σ	τ	υ	φ	χ	ψ	ω
240	ϱ	ἷ	ὗ	ᾆ	ᾇ	ῃ	ῳ	ᾳ	ἁ	ἑ	ἱ	ὁ	ὑ	ῦ	ὄ					
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
	0										1									

Table 8. Non-standard 8-bit encoding: Latin font with diacritics

	0										1									
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
000		.	˙	¯	˘	ˇ	´	`	¨	^	˝	.	˙	ˇ	˘	ˇ	´	`	¨	‟
020	˝	§	⁀	˛	¸	Ł	Þ	ħ	u	˚	ʻ	ʽ		!	"	#	†	°	+	'
040	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;
060	<	=	>	?	√	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
080	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	¯	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	≈	ž̊	ü	é	â	ä	à	å	ç	ê	ë	è	ï
140	î	ì	Ä	ø	ė	æ	œ	ô	ö	ò	û	ù	ẏ	Ö	Ü	ã	ẽ	ĩ	õ	ũ
160	á	í	ó	ú	ñ	ŋ	ā	ē	ī	ō	ū	ā́	ǰ	ī́	ł	ū́	ā̀	ĕ	ī̀	ı
180	ū̀	ą̇	ā̆	x́	x^{u}	ž	ŋ^{u}	ṛ̀	ĭ	r̄	ŭ	ą	ę	į	ǫ	ų	i̯	u̯	ə	ə̄
200	ə̯	ą̄	ą́	ę̄	ę́	ḗ	ḗ	į̄	į́	ų̄	ų́	ū̃	ỹ	ý	β	ƀ	č	ḍ	đ	δ
220	ǵ	ġ	ɡ̨	γ	ḫ	ß	ḥ	ƕ	ḱ	ḷ	ḷ́	ḹ	l̃	ṃ	m̃	m̊	m̨	ṅ	ń	ŋ́
240	ṇ	ṛ	ṛ́	ṝ	ṝ́	r̃	ś	ṣ	š	š̊	ṣ̌	ṯ	ṭ	ϑ	þ					
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
	0										1									

standardized way, and it gets lost all too easy when data are transferred across systems. One example may suffice to illustrate this effect which would be hazardous for a long-term storage of textual materials.

1.3. Conversion and the loss of data: An example

Table 9a shows the first ten lines of a Svan folk song, digitized in the early 1980s in a DOS environment with a special font covering the requirements of the Latin transcription of South Caucasian languages. Encoded as a plain text, with no additional information whatsoever on the font, i.e., the encoding used, the text would have appeared as displayed in Table 9b on the DOS "system" screen, and the recovery of what symbol stands for what character would have been a hard task indeed. Imagine a linguist working in 200 years time who would not have any other information on the language in question (which may well have died out by then – Svan is among the languages dealt with in our DoBeS project "Endangered Caucasian Languages in Georgia"),[1] he or she would have no chance to restore the "values" of the crucial "characters" and thus to reestablish the text itself.

Table 9 a/b. Font mapping in 8-bit encoding: Svan sample text

a.	b.
1 *voǯ ġal sabirelo Nuarsala!*	1 vo■ ╟al sabirelo Nuarsala!
2 *Mušvraši ṭubas esġəri,*	2 MuΩvraΩi ∩ubas es╠╩ri,
3 *sgobin laǯxvidax Čolšare,*	3 sgobin la■xvidax ⌐olΩare,
4 *min ǯixaldax si moḳtare,*	4 min ■ixaldax si mo╓tare,
5 *esran irix min amxvare.*	5 esran irix min amxvare.
6 *ka laǯšədax ečxän-amxän,*	6 ka la■Ω╩dax e—xän-amxän,
7 *meqrär šəq̇asuġv eǯlaǯix,*	7 meqrär Ω╩ɪasu╟v e■la■ix,
8 *ču laǯṭəxix Mušvra ṭubas.*	8 —u la■∩╩xix MuΩvra ∩ubas.
9 *Davberxo lekva esq̇adäs,*	9 Davberxo lekva esɪadäs,
10 *Davbrar q̇ōrars xocq̇analix:*	10 Davbrar ɪпrars xocɪanalix:
11 *ləmšare sgoǯix mušgvriša.*	11 l╩mΩare sgo■ix muΩgvriΩa.

1.4. Unicode: Towards a world-wide standard

What, then, is the way out of this problem? The answer is clear: To be able to uniquely encode all characters that have been used in writing down human languages (including both "national" scripts and alphabets and linguistic "metascripts" such as the International Phonetic Alphabet), the basis of encoding must be extended far beyond the 1-byte (8-bit) standard. This is exactly what has been undertaken since the early 1990s when the so-called "Unicode" standard was created: Based on 16 bits (or 2 bytes), this standard comprises 2^{16} = 65536 basic "code points" used for the "unique" encoding of characters. Considering that for the Chinese script alone, far more than 65,000 different characters have been used throughout history, it is clear that even this standard is not yet sufficient to cover all characters used by mankind at all times. A further extension is envisaged, however, in the 32-bit standard ISO 10646 which provides a total of (2^{32} =) 4,294,967,296 code points; as a matter of fact, the Unicode standard is but one subset of this "infinite" inventory, just as the ANSI standard (ISO 8859-1) is a subset of Unicode and the ASCII standard, a subset of ANSI (cp. Figure 1).

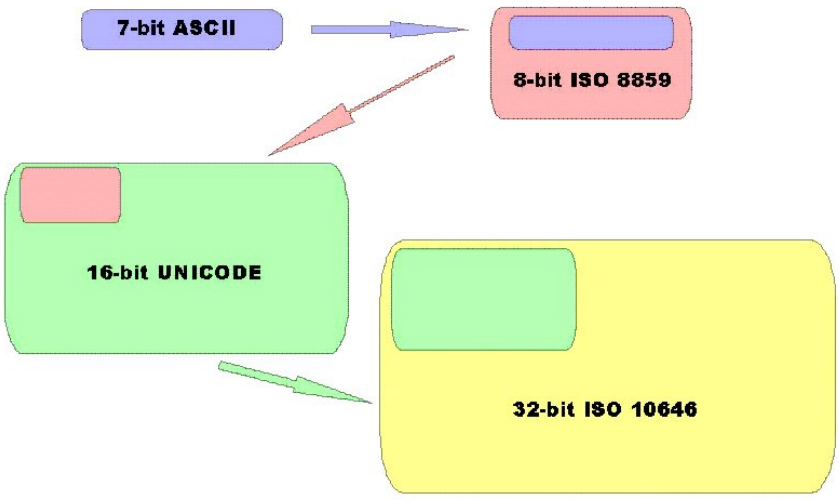

Figure 1. From 8-bit to 32-bit encoding

Along with the expansion of the World Wide Web, Unicode encoding has become more and more prominent since the late 1990s, and it is the encoding basis of more and more up-to-date operating systems and word processors. There can be no doubt that this is a huge advantage for the purposes of linguistic documentation. Cf., e.g., Tables 10a and b which show a few of the "blocks" of Unicode characters: the distinction of a Cyrillic *cha* (ч) and a Latin *c* with cedilla (ç) is now guaranteed by their different code points (hexadecimal number 0447 = decimal 1095 vs. hexadecimal 00E7 = decimal 231), and various Latin-based characters used in transcription systems can now as well be encoded as characters of the Greek, Georgian, or Chinese scripts.

Table 10 a/b. 16-bit encoding: Unicode blocks Latin and Cyrillic

a.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000																
001																
002	?	!	„	#	$	%	&	'	(	)	*	+	,	-	.	/
003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
008																
009																
00A	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	®	¯		
00B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
00C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
00D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
00E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
00F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

b.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
040	Ѐ	Ё	Ђ	Ѓ	Є	S	I	Ї	J	Љ	Њ	Ћ	Ќ	Ѝ	Ў	Џ
041	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
042	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
043	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
044	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
045	ѐ	ё	ђ	ѓ	є	s	i	ї	j	љ	њ	ћ	ќ	ѝ	ў	џ
046	Ѡ	ѡ	Ѣ	ѣ	Ю	ю	А	а	Ꙗ	ꙗ	Ж	ж	Ꙟ	ꙟ	Ӂ	ӂ
047	Ѱ	ѱ	Ѳ	ѳ	V	v	Ѵ	ѵ	Ѷ	ѷ	Ꙋ	ꙋ	Ѹ	ѹ	Ѻ	ѻ
048	Ҁ	ҁ	҂					◌҃	◌҄	◌҅	Й	й	Ь	ь	Ҏ	ҏ
049	Ґ	ґ	Ғ	ғ	Ҕ	ҕ	Ж	ж	Ҙ	ҙ	К	к	Қ	қ	Ҝ	ҝ
04A	Ҟ	ҟ	Ҡ	ҡ	Ң	ң	Ҥ	ҥ	Ҧ	ҧ	Ҩ	ҩ	Ҫ	ҫ	Ҭ	ҭ	Ү	ү
04B	Ұ	ұ	Х	х	Ҳ	ҳ	Ц	ц	Ҷ	ҷ	Ч	ч	Ҹ	ҹ	Һ	һ	Ҽ	ҽ	Ҿ	ҿ
04C	І	Ӂ	ӂ	Ӄ	ӄ	Л	л	Ӆ	ӆ	Ӈ	ӈ	Ӊ	ӊ	Ӌ	ӌ	Ӎ	ӎ			
04D	Ӑ	ӑ	Ӓ	ӓ	Æ	æ	Ӗ	ӗ	Ə	ə	Ә	ә	Ӛ	ӛ	Ӝ	ӝ	Ӟ	ӟ	Ӡ	ӡ
04E	Ӣ	ӣ	Ӥ	ӥ	Ӧ	ӧ	Ө	ө	Ӫ	ӫ	Ӭ	ӭ	Ӯ	ӯ						
04F	Ӱ	ӱ	Ӳ	ӳ	Ӵ	ӵ			Ӹ	ӹ						
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

In passing it may be noted that Unicode was not the first attempt to prevent the chaos of 8-bit font mapping by 16-bit encoding. As early as 1988, the word processor WordPerfect 5.0 was introduced which comprised a set of 1632 uniquely encodable characters, among them Greek, Cyrillic, and

Japanese (*hiragana* and *katakana*) sets, plus a block of 255 "user definable" entities. In this way, WP 5 encoded texts may meet the requirements of unique character encoding even today, and it should be possible to keep the information they contain intact when transferring these texts into the Unicode standard. Unfortunately, the WP encoding system was not widely used and the opportunities it offered were mostly ignored; thus we cannot expect the automatic conversion routines for WP 5 texts offered by, e.g., MS Word 2000 to correctly interpret and re-encode any one of the non-standard characters that may be contained in them. Cf. Table 11, which illustrates what happens when the Svan folk song we have dealt with above (cf. Table 9) is consistently encoded in WP 5 and then automatically converted into a MS-Word text. It is especially the replacement of "unidentified" characters by an undifferentiated underline score (_) which makes the conversion result unusable and irrepairable. The same holds true for the automatic conversion provided by later versions of WordPerfect itself (e.g., WP 9); here, we find a replacement of, e.g., ə by B, which is at least confusing. This all means that a correct conversion of WordPerfect 5 encoded texts (or, at least, of the characters contained in them) into Unicode encoding is possible, but it still requires special programming.

Table 11 a/b. Automatic text „conversion": Svan example

	a.	b.
1	*vo_ Ÿal sabirelo Nuarsala!*	*voÅ ġal sabirelo Nuarsala!*
2	*Mušvraši _ubas esŸ_ri,*	*Mušvraši Mubas esġBri,*
3	*sgobin la_xvidax _olšare,*	*sgobin laÅxvidax volšare,*
4	*min _ixaldax si mo_tare,*	*min Åixaldax si moåtare,*
5	*esran irix min amxvare.*	*esran irix min amxvare.*
6	*ka la_š_dax e…xän-amxän,*	*ka laÅšBdax ečxän-amxän,*
7	*meqrär š__asuŸv e_la_ix,*	*meqrär šBïasuġv eÅlaÅix,*
8	*…u la___xix Mušvra _ubas.*	*ču laÅMBxix Mušvra Mubas.*
9	*Davberxo lekva es_adäs,*	*Davberxo lekva esïadäs,*
10	*Davbrar _Çrars xoc_analix:*	*Davbrar ïōrars xocïanalix:*
11	*l_mšare sgo_ix mušgvriša.*	*lBmšare sgoÅix mušgvriša.*

1.5. Coexisting standards: The worst case scenario

The question now is, are we really on the safe side after Unicode has become the world-wide basis of character encoding? To be honest, there are still quite a lot of puzzling problems to be solved, not only with respect to the conversion of older material. The major problem lies in the fact that for the time being, digital word processing is characterized by the actual coexistence of 16-bit and 8-bit encoding systems. Just as the 8-bit ANSI standard was integrated into the 16-bit Unicode standard as one of its "blocks", all Unicode-based word processors such as MS Word 2000 have been designed to be ready to handle 8-bit encoded texts alongside 16-bit encoded ones. In the same way, Unicode-based operating systems such as MS Windows 2000 have been designed to be able to incorporate 8-bit encoded fonts side by side with 16-bit encoded ones. A few examples may suffice to show what confusion this may bring about.

Table 12a displays the fragment of a Georgian verb list which was typed in MS Word 6, using a plain 8-bit based Georgian font mapped onto the 8-bit ANSI encoding scheme. When I received this text file from a colleague in Georgia via e-mail two years ago, I tried to open it in MS Word 2002 (XP Office). The result was funny, to say the least: What appeared on the screen was a text in Japanese *katakana* script instead (cp. Table 12b). When I opened the text in Open Office 1 instead, another result appeared: The Georgian characters were now replaced by Latin characters with diacritics (cf. Table 12c), which was a foreseeable result bearing in mind that the original encoding was 8-bit based. After applying the correct Georgian font to this text within Open Office, the intended look (as in Table 12a) reappeared, and the text could even be re-mapped onto a transcriptional font which used the same 8-bit code points (cf. Table 12d). Trying to apply the Georgian font to the "Japanese" looking output of MS Word 2002 changed nothing, however; the *katakana* characters remained *katakana* characters (as displayed in Table 12b).

Table 12 a-d. Automatic text "conversion": Georgian example (wordlist)

a. Original text (MS Word 6)

0020010M	გააღვიღება (გააღვიღებ-ისა)	0020020M	გააზნაურება (გააზნაურებ-ისა)
0020030M	გაბმა (გაბმ-ისა)	0020040N	გაგა-ჲ (გაგ-ისა)
0020050M	გაგება (გაგებ-ისა)	0020060P	გაგებულ-ი (გაგებულ-ისა)
0020070M	გაგზავნა (გაგზავნ-ისა)	0020080N	გაგზავნა-ჲ (გაგზავნ-ისა)

b. Same text after cross-version transfer (MS Word 6 > MS Word 2002)

0020010M	ツタタテナノヒトチタ (ツタタテナノヒトチ - ノモタ)	0020020M	ツタタニへタヨメトチタ (ツタタニへタヨメトチ - ノモタ)
0020030M	ツタチフタ (ツタチフ - ノモタ)	0020040N	ツタツタ - ホ (ツタツ - ノモタ)
0020050M	ツタツトチタ (ツタツトチ - ノモタ)	0020060P	ツタツトチヨヒ - ノ (ツタツトチヨヒ - ノモタ)
0020070M	ツタツニタナへタ (ツタツニタナへ - ノモタ)	0020080N	ツタツニタナへタ - ホ (ツタツニタナへ - ノモタ)

c. Same text after cross-program transfer (MS Word 6 > Open Office 1)

0020010M	ÂÀÀÃÅÉËÄÁÀ (ÂÀÀÃÅÉËÄÁ-ÉÓÀ)	0020020M	ÂÀÀÆÍÀÖÒÄÁÀ (ÂÀÀÆÍÀÖÒÄÁ-ÉÓÀ)
0020030M	ÂÀÁÌÀ (ÂÀÁÌ-ÉÓÀ)	0020040N	ÂÀÂÀ-Î (ÂÀÂ-ÉÓÀ)
0020050M	ÂÀÂÄÁÀ (ÂÀÂÄÁ-ÉÓÀ)	0020060P	ÂÀÂÄÁÖË-É (ÂÀÂÄÁÖË-ÉÓÀ)
0020070M	ÂÀÂÆÀÅÍÀ (ÂÀÂÆÀÅÍ-ÉÓÀ)	0020080N	ÂÀÂÆÀÅÍÀ-Î (ÂÀÂÆÀÅÍ-ÉÓÀ)

d. Same with different font-assignment (within Open Office 1)

0020010M	gaadvileba (gaadvileb-isa)	0020020M	gaaznaureba (gaaznaureb-isa)
0020030M	gabma (gabm-isa)	0020040N	gaga-j (gag-isa)
0020050M	gageba (gageb-isa)	0020060P	gagebul-i (gagebul-isa)
0020070M	gagzavna (gagzavn-isa)	0020080N	gagzavna-j (gagzavn-isa)

How can this odd behaviour of MS Word be explained? Obviously, the program executes a five-step strategy when it encounters texts encoded by other (older) versions:

— it first checks whether the document is Unicode-encoded;
— if not, it checks whether the character distribution might meet the "typical" distribution of one of the known codepages;
— if yes, it assumes that codepage to be represented;
— it converts the 8-bit characters of the assumed codepage into the equivalent characters of Unicode;
— it stores the Unicode characters in memory.

Reapplying the original 8-bit fonts can then be no remedy if they do not meet the Unicode encoding assumed, as in the given case where a "Japanese" codepage was assumed to be present.

1.6. Persisting non-standards: The Private Use Area

One other problem that may be crucial even in Unicode times is the persistence of at least one area that is designed for font mapping. This is the so-called "Private Use Area" (PUA) which comprises 6144 non-predefined characters in the blocks E000-EFFF and F000-F7FF. Quite like the "user definable area" of WordPerfect 5, it can be assigned *ad libitum* by companies, user groups, or individuals, with the result that additional information is necessary to distinguish the characters "encoded" in it. Table 13 shows what can happen when a wrong font is applied to visualize PUA encoded characters; in the worst case, the intended information will again be lost.

Table 13 a/b. 16-bit font mapping: The "Private Use Area"

a.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E80	.	!	(	)	«	»	[illegible]	[illegible]	[illegible]	[illegible]	.	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
E81	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ء	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
E82	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
E83	ث	ث	ټ	ٽ	ڂ	ڂ	ڂ	ڂ	ځ	ځ	ځ	ځ	ډ	ډ	ٻ	ٻ
E84	ړ	ړ	ڙ	ڙ	ڕ	ڕ	ڒ	ڒ	ڔ	ڔ	ږ	ږ	ښ	ښ	ښ	ښ
E85	ڠ	ڠ	ڠ	ڠ	ك	ك	ک	ک	گ	گ	گ	گ	ګ	ګ	ګ	ګ
E86	ڳ	ڳ	ڳ	ڳ	ڵ	ڵ	ڵ	ڵ	ﻻ	ﻻ	ڼ	ڼ	ڼ	ڼ	وَ	وَ
E87	ۋ	ۋ	ێ	ێ	ێ	ێ	ې	ې	ڀ	ڀ	ٺ	ٺ	ٿ	ٿ	ی	ں
E88	لأ	لأ	لأ	لأ	ڽ	ڽ			ڄ							
E89																
E8A																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

b.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E80	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
E81	·	·	·	·	·	⺁	𠂇	𠂉	𠃌	𠃊	㑳	㑇	⺈	⺋	龴	㖞
E82	㘚	㘎	⺌	⺗	㥮	㤘	龵	㧏	㧟	㩳	㧐	龶	龷	㭎	㱮	㳠
E83	⺧	龹	龸	⺪	䁖	䅟	⺮	䌷	⺳	⺶	⺷	龻	䎱	䎬	⺻	䏝
E84	䓖	䙡	䙌	龼	䜣	䜩	䝼	䞍	⻊	䥇	䥺	䥽	䦂	䦃	䦅	䦆
E85	䦟	䦛	䦷	䦶	龺	龾	䲣	䲟	䲠	䲡	䱷	䲢	䴓	䴔	䴕	䴖
E86	䴗	䴘	䴙	䶮	䜌	·	·	·	·	·	·	·	·	·	·	·
E87	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
E88	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
E89	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
E8A	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

1.7. Suggestions and recommendations

As far as character encoding is concerned, all this leads to a few general recommendations that may be helpful with respect to both data exchange and long-term archiving of textual materials:

— Wherever possible, be sure to use 16-bit encoding, not 8-bit encoding;
— if using 16-bit encoding, avoid addressing the Private Use Area.
— If 8-bit encoding is required, try not to mix up several fonts with a different encoding in one and the same document;
— always keep track of what font-and-encoding you are using;
— always inform the receivers about all this and provide the fonts (if legally possible).

Archivers should be even more rigid:

— They should convert all 8-bit documents into 16-bit Unicode documents and
— they should not use the Private Use Area for the encoding of characters.

But how to produce 16-bit encoded texts? As we have seen, the most common word processors of today are designed to handle both 8-bit and 16-bit encodings. Using MS Word 2002 under MS Windows XP and typing with a "national" keyboard as provided by the operating system, you can be quite sure that what you type will be stored in 16-bit encoding. If, however, you want to add some characters from, e.g., an IPA font, by using the symbol insertion menu, you should check whether the Unicode value given for the character in question matches the respective code point of Unicode or not – if not, the font you intend to use is most probably 8-bit encoded. As a matter of fact, MS Word 2002 does allow for mixtures of 8-bit and 16-bit encodings within a given text document – which may turn out to be the worst case as far as data exchange and storage is concerned. Problems may also occur when you use special keyboard drivers supplied by third parties such as Tavultesoft Keyman: These may have been designed for 8-bit encoding alone, giving you no chance to enter 16-bit encoded text with them. If you intend to design your own keyboard driver with Keyman or with the MS Keyboard Layout Creator, be sure to use Unicode encoding as its basis. Note, by the way, that the SIL Shoebox program was exclusively 8-bit based; it interacted well with Keyman drivers, but also only on an 8-bit basis. The newly developed Toolbox now is Unicode-based and should work well with 16-bit based Keyman layouts.

2. The encoding of text elements: Surface appearance vs. content markup

2.1. Text structure visualized

Let us now turn to the second topic of this chapter, viz. the encoding of the structural elements of texts. To clarify what this means, it is helpful to look again at the Svan text we have dealt with above (cf. Table 9). Even without any knowledge of the language, we will immediately have the impression that this text consists of verses. This is clearly indicated by two signals we are used to in reading poetical texts, viz. the relative shortness of lines, and the numbers (from 1 to 11) given to each line. There are many further elements of textual structure involved, however. First, we will easily guess that the text consists of five sentences, partially extending across verses and partially consisting of subordinate clauses: This is indicated by the punctuation marks used. Then, we will be able to state that the text consists of 38 words, in their turn indicated by either empty spaces or punctuation marks adjoining their first and last characters.

2.1.1. The basic elements

This may all sound trivial, but as a matter of fact, it can be crucial indeed for the documentation of textual materials to consider and markup their internal elements when preparing them for future usage, and this should be done as consistently as the encoding of the characters appearing in words. So what elements are we talking about? Among the basic elements of every kind of text, we have already mentioned words (consisting of characters when written down), phrases, clauses, sentences; on a higher level, we will meet sections, paragraphs, chapters, text parts and the like. For many of these elements, we intuitively adapt signals we have been used to since we were at school, such as spaces indicating word boundaries, full stops indicating sentence breaks, or “hard” line breaks indicating the end of a section or paragraph. For a consistent encoding of a digital text, this may not be sufficient, though. Another example may suffice to illustrate why.

2.1.2. An illustrative example

In Table 14, we see a specimen from an 18th century grammatical treatise in Georgian, digitized using MS Word 6. Without even a faint knowledge of the Georgian script, a reader may guess that the first line of the text is a heading, given that it obviously consists of but one word, is centered on the line and seems to be represented in a bold face font. As to the other lines of texts, the reader will as easily suspect that this is an interplay of questions and answers, the former being clearly indicated by question marks. One more suggestion might impose itself: as the first word of every question and answer is separated by a colon and marked by an extra spacing of characters, and as these words are repeated throughout questions and answers, they might indicate the names of people speaking (as in a theater play). All these assumptions are correct: we do have an interplay of questions and answers, uttered by two different persons here (one Ioane, one Nikolaoz), and the first line is the heading (it simply means "On grammar"). The reason why it was so easy to find all this out is that here again, marking methods were applied that we are used to in reading – centering of lines, usage of boldface, spacing of characters, etc. For computational purposes, however, these markings, which we may call **surface-oriented**, are arbitrary and insufficient in a twofold sense.

Table 14. Georgian text specimen

ღრამმატიკისათჳს

ი ო ა ნ ე მ: ოთხნი იგი გვარნი მოძღვრებითნი, რომელნიცა შეუდგებიან, დაემდევრებიან ღრამმატიკასა.

ნ ი კ ო ლ ა ო ზ მ ა ნ: რაჲ არს სახელები მათი?

ი ო ა ნ ე მ: განსაზღვრება, განწვალება, აღმოჩენა და აღლევა.

ნ ი კ ო ლ ა ო ზ მ ა ნ: კვალად რაჲ საჳმარ არს ცნობად?

2.1.3. Program features vs. standards

First, the centering of lines may be a common feature of all existing word processors today, but it is by no means standardized: The encoding of this feature simply depends on the program structure. To illustrate what this

means, Table 15 shows a part of the internal code of the given MS Word document. Here we detect the word contained in the heading (Georgian ღრამმატიკისათვის "On grammar", stored in 8-bit form) at the end of what appears to be a sixth line, followed by a "clear text" form of the questions and answers. There is no indication adjoining the "head" word that it must be centered or boldfaced, and none that it represents a heading. All this must be inferred, by the interpreting program, from the unreadable code preceding it (or from a similar looking block of coding elements added at the end of each MS Word document). Imagine somebody were to decode this document in 200 years time, without having any access to the internal program code structure of MS-Word 6 – he or she would certainly not be able to extract anything from it but the "plain text", and all the additional information contained in the centering of lines and boldfacing of words would be lost (as a matter of fact, many of us have witnessed this effect when trying to open MS Word documents of the 1980s in later versions). The same would be true for the "spaced" characters indicating the speakers in the text: the spacing is here, too, covered by a program-internal function and would be lost together with the knowledge of the code. It would not be a good idea, by the way, to avoid this latter effect by inserting the character spaces manually instead of using the word processor function for it: if, as we have seen above, we use spaces to distinguish words from each other, the spaced name I O A N E would automatically appear as five words (consisting of but one character each) to any computational analysis, and it could not be found when searching for "IOANE".

Table 15. Program-specific encoding of Georgian text specimen

```
ÐÏà¡±ÿáÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿAÿþÿCÿÿÿÿÿÿÿÿÿÿÿÿÿAÿÿÿÿÿ´AÿÿÿÿÿØ
ÿÿÿÿÿÿÿ BÿÿÿÿÿÀÿÿÿÿÿÿFÿÿÿMicrosoft Word 6.0 Documentÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿMSWordDocÿÿÿÿWord.Document.6ÿô9²qÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿ  Aÿÿÿÿÿ´AÿÿÿÿÿØAÿÿÿ  Aÿÿÿÿÿ´AÿÿÿÿÿØAÿC:\WORKDIR\TEMPLATE\NORMAL.DOTÿÿ
ÿÿÿüAÿÿÿÿÿ BÿÿÿÿÿDBÿÿ ÿÿÿhBÿÿÿÿÿŒBÿÿÿÿÿÿÿÿÿÙRAMMAºI±ISATWS
IOANEM: OTXNI IGI GVARNI MO¾ÙVREBITNI, ROMELNICA ¸EUdGEBIAN,
dAEMdEVREBIAN ÙRAMMAºI±ASA.
NI±OLAOzMAN: RAJ ARS SAXELEBI MATI?
IOANEM: GANSAzÙVREBA, GANÂVALEBA, AÙMOÆENA dA AÙLEVA.
NI±OLAOzMAN: ±VALAd RAJ SAQMAR ARS CNOBAd?
IOANEM: ESE, VITARMEd çOELI RAM ARSOBITI M±ITX–ÉLI SAzÙVARS E¾IEBS.
NI±OLAOzMAN: MERMEd RAJ SAQMAR ARS CNOBAd?
```

2.1.4. What you see is NOT what you get

What, then, can be done to avoid a loss of the information concerning the structuring of texts and their elements? First, we should get rid of an ideal in text processing which has become very widespread these days, viz. "WYSIWYG": "What you see is what you get". It may be true that the text you type in on your computer today will look quite the same on the screen and in a printout, but all this is restricted to a very ephemeral use: the next generation of users of your text may have no access to the sophisticated codings of your word processor and will thus "get" anything else but what you "saw". Second, we should give up the idea that the use of mere printing devices (such as boldfacing, spacing of characters, and the like) might be enough to indicate the function of text elements. Instead, we should adapt ourselves to what may be called "content markup" whenever our texts are meant to be stored for documentation purposes.

2.2. A half-way solution: HTML

In recent years, the marking up of text elements has indeed become more and more widespread especially by the expansion of the World Wide Web and the prescription to use a certain unified text encoding structure, the so-called Hypertext Markup Language (HTML), for documents to be provided in it. Tables 16 a and b show the Georgian text specimen converted into a plain HTML text (as source code and visualized with a standard web browser); here, you will easily find the markup devices corresponding to the centering and boldfacing of the heading, viz. the markers ... and ... . What you will miss is the special markup of the speakers' names; this cannot be present as the spacing of characters is not markable as such in HTML. But even if it were (actually, so-called "cascading style sheets", CSS, can be used for this purpose), it would be no good idea to use this kind of markup alone – future users might hardly grasp the idea what it stands for as the spacing of characters has no standardized meaning. In the same way, it remains unclear what the centering and the bolding of the first line is to indicate – that this is a heading remains a mere guess. As a matter of fact, the markup provided by HTML contains but very few "content" elements. One is the group of markers from <H1> to <H6> which should be used to denote several levels of headings. In our case, it would be much better to mark our heading with one of these elements (re-

placing ... by <h1 align=center> ... </h1>) – the outer appearance would then be secondary and adaptable to future uses.

Table 16a. Plain HTML encoding of Georgian text specimen

```
<HTML>
   <HEAD>
      <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
      charset=iso-8859-1">
      <TITLE>Grammatika</TITLE>
      <META NAME="KeyWords" CONTENT„Georgian Grammar">
   </HEAD>
   <BODY>
      <DIV>
         <P ALIGN="CENTER"><B>ÙRAMMA°I±ISATWS</B></P>
      </DIV>
      <DIV>
         <P><SPAN>IOANEM: </SPAN><SPAN>OTXNI IGI GVARNI
         MO¾ÙVREBITNI, ROMELNICA ¸EUdGEBIAN, dAEMdEVREBIAN
         ÙRAMMA°I±ASA.</SPAN></P>
         <P><SPAN>NI±OLAOzMAN: </SPAN><SPAN>RAJ ARS SAXELEBI
         MATI?</SPAN></P>
         <P><SPAN>IOANEM: </SPAN><SPAN>GANSAzÙVREBA,
         GANÂVALEBA, AÙMOÆENA dA AÙLEVA.</SPAN></P>
         <P><SPAN>NI±OLAOzMAN: </SPAN>±VALAd RAJ SAQMAR ARS
         CNOBAd?</SPAN></P>
         ...
      </DIV>
   </BODY>
</HTML>
```

Table 16b. Browser output of Georgian HTML text specimen

ღრამმატიკისათჳს

იოანემ: ოთხნი იგი გვარნი მოძღვრებითნი, რომელნიცა შეუდგებიან, დაემდევრებიან ღრამმატიკასა.
ნიკოლაოზმან: რაჲ არს სახელები მათი?
იოანემ: განსაზღვრება, განჭვალება, აღმოჩენა და აღლევა.
ნიკოლაოზმან: კვალად რაჲ საჴმარ არს ცნობად?
…

2.3. Real content markup: XML

The more information of this type is to be encoded, the less will HTML markup suffice. For a consistent markup of the contents of a text, you will have to go one step further and adapt the eXtensible Markup Language, XML (a derivate of the Standard Generalized Markup Language, SGML). This alone will allow you to provide for future users all the knowledge you might have on the text materials you are working on. In an XML markup, you will easily be able to declare not only the heading of the text as its heading but also the speakers as speakers, their utterances as questions and answers relating to each other, and any other text element that might be useful to define. Table 17 shows the Georgian grammar example provided with a minimal XML markup; you will easily note the difference as against the HTML markup which consists in the meaningfulness of the tags.

```
<?xml version=„1.0“ encoding=„utf-8“?>
  <part>
    <pnum>1</pnum>
    <chapter>
      <cnum>1</cnum>
      <heading>ღრამმატიკისათვს</heading>
      ...
      <utterance>
        <unum>1</unum>
        <utype>question</utype>
        <speaker>ნიკოლაოზმან</speaker>
        <sentence>
          <snum>1</snum>
          <item>
            <inum>1</inum>
            <itype>word</itype>რაჲ</item>
          <item>
            <inum>2</inum>
            <itype>word</itype>არს</item>
          <item>
            <inum>3</inum>
            <itype>word</itype>სახელები</item>
          <item>
            <inum>4</inum>
            <itype>word</itype>მათი</item>
          <item>
            <inum>5</inum>
            <itype>question mark</itype>?</item>
        </sentence>
      </utterance>
```

```
        <utterance>
          <unum>2</unum>
          <utype>answer</utype>
          <speaker>იოანემ</speaker>
          <sentence>
            <snum>1</snum>
            <item>
              <inum>1</inum>
              <itype>word</itype>განსაზღვრება</item>
            <item>
              <inum>2</inum>
              <itype>comma</itype>,</item>
            <item>
              <inum>3</inum>
              <itype>word</itype>განწვალება</item>
            <item>
              <inum>4</inum>
              <itype>comma</itype>,</item>
            <item>
              <inum>5</inum>
              <itype>word</itype>აღმოჩენა</item>
            <item>
              <inum>6</inum>
              <itype>word</itype>და</item>
            <item>
              <inum>7</inum>აღვლევა</item>
            <item>
              <inum>8</inum>
              <itype>full stop</itype>.</item>
            <item>
          </sentence>
        </utterance>
        ...
      </chapter>
    </part>
  </text>
```

2.4. XML in language documentation: Going beyond plain text encoding

Of course, all kinds of analyses of linguistic units such as words and phrases can also be included in an XML markup, and this is the real advantage it has for the documentation of languages. You can be sure that future users will hardly be interested in sharing the surface beauty of a text document; what they will be interested in is as much information about the language as you can provide. For many years, linguists have used the Shoebox program for the purpose of noting down and annotating texts they collected during

their fieldwork, and for many of us the facilities offered by this program, especially the half-automatical process of interlinearization, is indispensable; cp. Figure 2 which exhibits a sample sentence in the Tsova-Tush or Batsbi language of the Caucasus[2]. The basic idea of interlinearization as provided by Shoebox consists in the vertical arrangement of interdependent annotation layers (tiers); these can include, as in the present example, different transcriptions and transliterations (here: Georgian script, Latin script, IPA), morphological analyses, the reference to lemmatic forms, translations of the lemmatic forms, etc. The Shoebox format is not sufficient in the sense of a thorough markup, though, as it has two disadvantages: the encoding used is still 8-bit based so that the correct display depends on the interpretative functions of the program; cf. Table 18 which shows the same Shoebox text when opened in a normal text editor. While the latter disadvantage has recently been overcome by the introduction of the Toolbox program, the Unicode-compatible successor of Shoebox 5.0, the second disadvantage remains: the interdependencies of the vertically aligned elements is not marked as such in a Shoebox / Toolbox text but depends on the interpretation of spaces between words. This is where XML markup would help: Only after the conversion of the Shoebox file into a Unicode based XML schema as the one displayed in Figure 3 can we be confident that all the information stored in the document will be accessible to later users for a long time.

\ref	0485								
\per	AS								
\trs	ჩუხუჲ	შუი^{n}	ნანიგორე^{n}	ჩაქ	დეც	ხილა^{n},	მე	ვაშბა^{n}	დაჰდიცდოლ^{o}.
\tl1	čuxuy	šui^{n}	nanigore^{n}	čaq	deç	xil'a^{n}	me	vašba^{n}	daḫdicdol^{o}
\ph	t͡ʃuxuj	ʃui^{n}	nanigore^{n}	t͡ʃaq	deɬs'	xiɬa^{n}	me	vaʃba^{n}	daɦdit͡sdol^{w}
\ts	ჩუხუჲ	შუი^{ნ}	ნანიგორე^{ნ}	ჩაქ	დეც	ხილა^{ნ}	მე	ვაშბა^{ნ}	დაჰდიცდოლ^{ო}
\ts1	čuxuy	šui^{n}	nanigore^{n}	čaq	deç	xil'a^{n}	me	vašba^{n}	daḫdicdol^{o}
\m	čux-uy	šui^{n}	nan-i-gore^{n}	čaq	d-eç	xil'a^{n}	me	vašba^{n}	daḫ-dic-d-ol-^{o}
\m1	čux-uy	šui^{n}	nan-i-gore^{n}	čaq	d-eç	xil'a^{n}	me	vašba^{n}	daḫ-dic-d-ol-^{o}
\lm	ჩუეხ	შუი^{ნ}	ნან	ჩაკ	დეწ^{ან}	ხილა^{ნ}	მე	ვაშბა^{ნ}	დაჰდიცოდა^{ნ}
\lm1	čuyx	šui^{n}	nan	čaq	deça^{n}	xil'a^{n}	me	vašba^{n}	daḫdicoda^{n}
\g	ბატკანი	თავისი	დედა	შორს	საშინრო_ა	ყოფნაყოლაქონა	რომ	ერთმანეთი	დავიწყება
\g1	batkani	tavisi	deda	šors	sačiro_a	qopna,qola,kona	rom	ertmaneti	davicqeba
\gl	lamb	own	mother	distant	to-be-necessary	to-be,to-have	that	each_other	to-forget
\p	**N.4Gr.**	**ReflPron.**	**N.2Gr.**	**Adv.**	**V.**	**V.**	**Conj.**	**Recipr.Pron.**	**V.**
\gr	Nom.Pl.	indecl.	Loc.Pl.+Postp.	indecl.	Pres.4Gr.	Inf.	indecl.	indecl.	Cond.4Cl.3Ps.
\fg	ბატკნები თავისი დედებისგან შორს უნდა იყვნენ, რათა ერთმანეთი დააივიწყდეთ.								
\fg1	batknebi tavisi dedebisgan šors unda iqvnen, rata ertmaneti daavicqdet.								
\fe	*The lambs must be apart from their mothers to forget them.*								
\c	33 09:29:50								
\dt	24/Apr/2005								

Figure 2. Shoebox text file with interlinearized annotations

Table 18. Same example as in Figure 2, viewed in normal text editor

\ref 0485
\per AS
\trs Äuxuy êui¤ nanigore¤ Äaq deÂ xiÚa¤, me vaêba¤ daÐdicdol†.
\tl1 Äuxuy êui¤ nanigore¤ Äaq deÂ xiÚa¤ me vaêba¤ daÐdicdol†
\ph tƒSuxuj Sui< nAnigore< tƒSAq detƒs' xiÂA< me vASbA< dAðditƒsdolW
\ts Äuxuy êui¤ nanigore¤ Äaq deÂ xiÚa¤ me vaêba¤ daÐdicdol†
\ts1 Äuxuy êui¤ nanigore¤ Äaq deÂ xiÚa¤ me vaêba¤ daÐdicdol†
\m Äux-uy êui¤ nan-i-gore¤ Äaq d-eÂ xiÚa¤ me vaêba¤ daÐ-dic-d-ol-†
\m1 Äux-uy êui¤ nan-i-gore¤ Äaq d-eÂ xiÚa¤ me vaêba¤ daÐ-dic-d-ol-†
\lm Äujx êui¤ nan Äaq deÂa¤ xiÚa¤ me vaêba¤ daÐdicoda¤
\lm1 Äujx êui¤ nan Äaq deÂa¤ xiÚa¤ me vaêba¤ daÐdicoda¤
\g baïÖani tavisi deda êors saÅiro_a çopna,çola,kona rom ertmaneti daviÂçeba
\g1 baïÖani tavisi deda êors saÅiro_a çopna,çola,kona rom ertmaneti daviÂçeba
\gl lamb own mother distant to-be-necessary to-be,to-have that each_other to-forget
\p N.4Gr. ReflPron. N.2Gr. Adv. V. V. Conj. Recipr.Pron. V.
\gr Nom.Pl. indecl. Loc.Pl.+Postp. indecl. Pres.4Gr. Inf. indecl. indecl. Cond.4Cl.3Ps.
\fg baïÖnebi tavisi dedebisgan êors unda içvnen, rata ertmaneti daaviÂçdet.
\fg1 baïÖnebi tavisi dedebisgan êors unda içvnen, rata ertmaneti daaviÂçdet.
\fe The lambs must be apart from their mothers to forget them.
\c 33 09:29:50

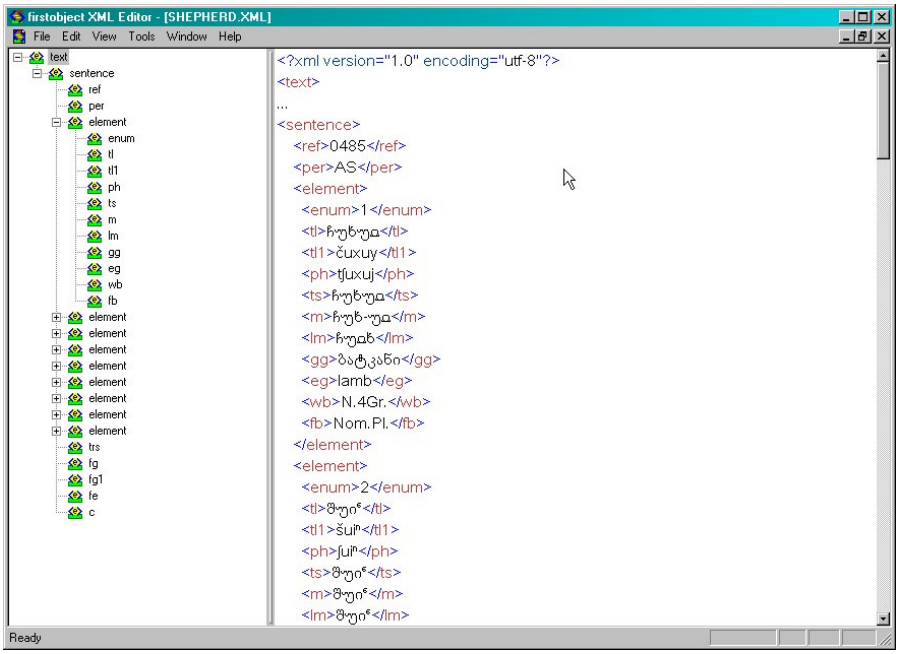

Figure 3. Same example as in Figure 2, converted into XML format

2.5. Outlook

It is true that the application of XML is not yet widely used by (fieldworking) linguists. It is also true, however, that it becomes more widespread every day, and lots of software programs that are dedicated to the production of consistent XML documents are now readily available (cf. the list attached below). No matter whether you intend to apply XML methods yourself in the near future or not, it may be worth while taking your time and visiting the website of the "Text Encoding Initiative" (TEI), just to learn more about what the structuring of textual elements means. Your linguistic work cannot but profit from this.

Notes

1. "ECLinG"; cf. the project homepage in http://titus.fkidg1.uni-frankfurt.de/ecling/ecling.htm.
2. The example is taken from the material recorded in the DoBeS "ECLinG" project; cf. n. 1.