7-bit Meta-Transliterations for 8-bit Romanizations

Klaus Lagally[1]
Institut für Informatik, Universität Stuttgart
Breitwiesenstraße 20-22, 70565 Stuttgart, GERMANY
EMail: `mailto:lagally@informatik.uni-stuttgart.de`

Abstract: [7-bit encoding, transliteration]

We propose a general strategy for deriving 7-bit encodings for texts in languages which use an alphabetic non-Roman script, like Arabic, Persian, Sanskrit and many other Indic scripts, and for which there is some transliteration convention using Roman letters with additional diacritical marks. These schemes, which we will call "meta-transliterations", are based on using single ASCII letters for representing Roman letters, and digraphs consisting of a suitable punctuation character and an ASCII letter for representing letters with diacritics. A meta-transliteration is required to be uniquely reversible, human readable, and close to the intended transliteration. We present an example of a scheme that has been in use for several years to transliterate texts in Arabic, Persian, Urdu, Sindhi, and Biblical Hebrew.

1 Introduction

Texts in languages using an alphabetic non-Roman script, like Arabic, Persian, Sanskrit, and many other Indic scripts, are traditionally transliterated into Roman letters with additional diacritical marks whenever required, if the original script cannot conveniently be used; we call such a mapping a *Romanization.* Depending on the language in question and on the context, there are many different Romanization conventions in use, in spite of current standards activities. This diversity of notations not only hampers communication between scholars using different conventions, but also poses severe problems for automatic processing, since the common word processing programs do not cater well for input and display of an ample range of diacritics.

For automatic storage, processing and displaying, this set of Romanization characters is either represented, as e.g., in the USMARC encoding [USMARC90] used in the ALA-LC tables [Bar91] by the base characters and, if required, modifier characters denoting the diacritics; or it may be mapped into a set of 8-bit byte patterns by some one-to-one correspondence. Thus to every 8-bit byte pattern used, there corresponds uniquely a Roman letter, possibly with some diacritics attached, and also a character of the original script, if we started from a complete transliteration.

From a logical point of view, these three levels of notation are equivalent; for practical purposes they are not. Our source texts are probably denoted in the original script, linguistic research may well be done in the Romanization, and for automatic processing we need the byte representation.

There are two separate one-to-one transformations involved, one between the original script and the Romanization, and one between the Romanization and its byte representation. Fixing, and agreeing on, the mapping to a Romanization is a task for scholars working actively in the field concerned; one of the main concerns there beside uniqueness is readability, which is strongly influenced by current practice and conventions. Details shall not concern us here; let us just assume some one-to-one convention has been found and agreed upon[2]. The second one-to-one mapping, between the Romanization and the internal byte representation, need not be of concern to the user if, and only if, there is some suitable text processing software available which can handle the Romanization. If the stored data are used only locally, the internal representation does not matter; however, whenever we want to interchange encoded texts between several parties, possibly using different hardware platforms, and possibly connected via a communication net, it also has to be standardized in some way.

To our knowledge a cross-platform standard for the internal representation of diacritized Roman letters does not exist, and text processing software for handling Roman letters with diacritics is in short supply. We believe there are good reasons for this state of affairs:

- none of the available character standards, not even [ISO10646] or [UNICODE], let alone some member of the ISO8859 family, directly contains all the character combinations needed,
- the special characters needed cannot readily be generated by a standard computer keyboard, even using national extensions, so some auxiliary input convention is required,
- standard character oriented display hardware frequently cannot handle the extended character set.

The two latter problems could be circumvented with some effort, as they are when the original script is also supported (but in this case the transliteration is no longer needed anyway). The first problem is more serious: presently the only universally supported standard for data transmission across various computer platforms and operating systems is the 7-bit ASCII encoding [ISO646]. This is also true for information presentation on the World Wide Web [BL+94] since unfortunately, according to our experience, not all of the popular Web browsers correctly display all the characters in the upper half of the ISO8859-1 code table that is otherwise formally supported, and furthermore this table does not contain all the Romanization characters commonly used.

If we restrict the character set of our internal representation to the usable part of the 7-bit ASCII character set (128 characters minus the reserved control characters, minus the space character) we end up with a set of 94 symbols which is normally insufficient for the number of transliteration characters we need; thus a one-to-one mapping of characters is no longer possible. A 7-bit encoding has to assign longer code words to at least some of the Romanization symbols that we want to represent.

There already have been several efforts of defining 7-bit encodings for various Romanization conventions, and there is a multitude of possibilities and more or less arbitrary choices; we suggest a systematical way of designing such encodings for some given Romanization convention.

2 Meta-Transliterations

We propose the term "meta-transliteration" for a mapping from a given "Romanization alphabet" (a set containing Roman letters and associated diacritics, digits and punctuation symbols) into a set of "code words" (nonempty strings of ASCII characters), which fulfills the following requirements:

- it is uniquely reversible
- it is human readable (after little practicing)

Once a meta-transliteration is available, we no longer need special hardware and software to write and display transliterated text; a simple ASCII text editor is sufficient. Texts encoded in the meta-transliteration format can be transparently transmitted between computers even of different brands, and across the internet; and due to the reversibility the problem of displaying the Romanization, or even the original script, has to be solved at most once for every platform involved.

3 Conditions for reversibility

Encoding a Romanization into a meta-transliteration may be performed by a simple table lookup, yielding the code word for a given Romanization character. For a meta-transliteration to be uniquely reversible by sequential processing, it can be shown to be necessary and sufficient that the encoding used satisfies the *Fano condition: no code word may be a prefix of another code word.* Then for decoding a *code tree* may be used: a tree structure where each branch is labelled by some ASCII character, and where every terminal node corresponds to some Romanization character such that, if we follow a path from the root of the tree to some terminal node, the sequence of branch labels traversed corresponds to the code word in question. Obviously, encoding and decoding can be implemented very efficiently.

4 Readability considerations

Human readability certainly is a subjective concept; it is influenced by personal experience and preferences, and some of our suggestions may look arbitrary. However it turns out that we do not have too many choices, due to the Fano condition, if we insist on unique reversibility.

For the sake of readability it seems plausible to encode a Roman character by the same ASCII character if it carries no diacritic, and to require that, if it carries any diacritical marks, the corresponding ASCII character should be part of its code word. If there is at least one diacritic, the Fano condition implies that the base character must come at the end of the code word, and that the diacritics must be represented by one or more "meta-characters" preceding the base character. Furthermore, characters used as meta-characters may not be used as their own code words, but have to be represented differently, e.g., by adding a space.

For a representation to be uniquely reversible, we could use any non-letter characters as meta-characters; but the requirement of human readability severely limits our choices. In all cases it ought to be obvious to the reader whether a sequence of characters is a complete code word or not, such that he will not be misled, even temporarily, into assigning a meaning to an incomplete code word; and a complete code word should convey its meaning at a glance after a very short learning phase. Thus the meta-characters used should look similar to the diacritics they are intended to indicate.

If the diacritical marks needed were available as such within the ASCII set, or if there were sufficiently many special characters available that resemble them and are otherwise unused, we could use them for the needed meta-characters. However this is not the case, since most special symbols are needed for punctuation.

Fortunately there is a way to reuse some of the punctuation marks. We exploit the fact that the text to be encoded is not a completely arbitrary sequence of characters, but represents a sequence of words, and possibly punctuation, in some language, and thus carries an additional structure. Words are generally written as sequences of Romanization letters, possibly also containing digits (for indicating tones for far-eastern languages), hyphen, solidus and vertical bar for internal grouping. They are separated or terminated by punctuation symbols or white space. Punctuation symbols never occur at the beginning or within a word, and may thus be used as meta-characters, like the other special characters. Whenever a punctuation symbol itself must be represented, we assign a code word to it consisting of the symbol followed by a space.

Still the supply of useful meta-characters is severely limited, and we probably cannot find any assignment that is equally suitable for all conventions presently in use, so we may end up with several encoding variants for different languages.

5 Example: Arabic, Persian, Urdu, and Sindhi

The author's ArabTEX package[3] [Lag92a, Lag93a, Lag94, Lag96, Lag97a] uses a meta-transliteration for encoding texts in Arabic, Persian, Urdu, Sindhi [Lag97b], and Biblical Hebrew [Lag95b].

The encoding used has been modelled after the ZDMG transcription [DIN31635] which we had to extend slightly. The ZDMG system is not uniquely reversible, since sometimes different spellings are transcribed identically. In Arabic this happens with the female ending *-at*, the indefinite flexion endings *-un*, *-in*, *-an*, *alif maqṣura*, silent final *alif*, and a few rare special cases; in Persian there are several variants of denoting the Izafet. This ambiguity poses no problem for a scholar who can understand the meaning of the text, but it precludes the automatic generation of the Arabic or Persian writing from the input encoding. This, however, was our primary goal, so we had to enforce reversibility. In all cases of ambiguity mentioned above we use capital letters for indicating variants; this is possible since these are customarily not used within a word (for additional convenience we also allow using capitals for denoting long vowels, and a few alternate popular assignments.) This is sufficient to uniquely re-create the transcription. When generating the writing, in some cases we have to do some local contextual analysis; this is outside the scope of this report.

We thus obtain a completely reversible transliteration, from which both the transcription and also the original Arabic writing can be automatically generated without resorting to linguistic knowledge. By substituting different code trees we can equally well obtain the Romanization scheme used in the ALA-LC tables [Bar91], and also the ISO conventions [ISO/R233], which both differ only slightly from the ZDMG scheme. To our surprise it proved feasible to extend the notation used for Arabic and Persian in an upwards compatible way to also cover Urdu and Sindhi [Lag97b], producing both the ALA-LC transliteration and the writing in the extended Arabic script.

We use the following set of meta-characters:

period	for dot below or above:	`.h .d .t .s .z .g`	*ḥ ḍ ṭ ṣ ẓ ġ*
caret	for haček:	`^s ^g ^c ^z`	*š ǧ č ž*
understroke	for bar below or above:	`_d _t _s _h _a _i _u _e _o`	*ḏ ṯ s̱ ẖ ā ī ū ē ō*
colon	for diaeresis or two dots:	`:j :b :d :g :n`	*j̈ b̤ d̤ g̤ n̤*
tilde	for tilde above:	`~n`	*ñ*
comma	for acute:	`,c ,t ,d ,r ,n ,s`	*ć t́ d́ ŕ ń ś*

The latter assignment was necessary since the acute and grave accent themselves have been made into pseudo-letters for *hamza* and *`ayin*, like in the ZDMG convention. We thus also lose the single quoting symbols, and have to encode quotes by doubling them. We observe that in some cases the same meta-character can be reused for different diacritics, since not all possible combinations are required.

The following example for Arabic may indicate the degree of readability obtained:

7-bit input	transcription	Arabic script
`i^star_A ^gu.hA 'a^saraTa .hamIriN`	*ištarā ǧuḥā ʿašarata ḥamīrin*	اِشْتَرَى جُحَا عَشَرَةَ حَمِيرٍ

The Sindhi Romanization contains an inherent ambiguity in cases where it does not differentiate between an aspirated consonant and a sequence of two distinct Sindhi letters. If we want to produce the correct Sindhi writing, we have to disambiguate the two cases by inserting a hyphen into the 7-bit input wherever required. This is not a weakness of the meta-transliteration but of the given Romanization.

6 Biblical Hebrew

Within an extension of the ArabTEX system for processing Biblical Hebrew [Lag95b], we did not follow exactly one of the various existing Romanization conventions, but instead expanded the encoding already available by the missing symbols (primarily for denoting semi-vowels.) This decision was influenced by an existing application where Arabic and Hebrew were needed within the same document; and we feared that providing two very similar, but incompatible, meta-transliteration systems would lead to utter confusion. Still, even though the 7-bit encoding is the same, different Romanizations for Arabic and Hebrew may be chosen, since the code trees used can be easily switched.

In fact we only had to expand the use of the period as a meta-character to additionally indicate the semi-vowels, and as a writing hint for *dagesh lene* and *mappiq*:

period	for semi-vowels	`.a .e .i .o`	*ă ǽ ĕ ŏ*
period	for dagesh lene and mappiq:	`.b .g .d .k .p .t .h`	*b g d k p t h*

Using these extensions the following (well-known) example may be obtained:

```
.b.ir_e'^sIt .bArA' '.el_ohIm '_et ha^s^sAmayim w.i'_et hA'A|re.s:
```

bᵉreʾšît bārāʾ ʾᵆlōhîm ʾet haššāmayim wᵉʾet hāʾāræṣ:

בְּרֵאשִׁית בָּרָא אֱלֹהִים אֵת הַשָּׁמַיִם וְאֵת הָאָרֶץ׃

7 Conclusion

Our experience indicates that by judicially reusing punctuation characters and capital letters in places where they normally do not occur, 7-bit meta-transliterations can be constructed that are guaranteed to be uniquely reversible, and that are sufficiently close to some existing Romanization conventions to be fairly readable. If small deviations from the corresponding Romanization conventions can be tolerated, a single meta-transliteration scheme may serve for several languages, even though we doubt that the goal of an universal 7-bit encoding can be attained.

References

[Bar91] Randall K. Barry.
ALA-LC Romanization Tables. Transliteration Schemes for Non-Roman Scripts.
Library of Congress, Cataloging Distribution Service, Washington, 1991.

[BL+94] Tim Berners-Lee et al.
The World-Wide Web.
Communications of the ACM, 37(8):76–82, 1994.

[DIN31635] Deutsches Institut für Normung e.V.
Umschrift des Arabischen Alphabets.
DIN 31 635, 1982.

[ISO10646] International Organization for Standardization.
Universal Coded Character Set.
Technical report ISO DIS 10646 (draft international standard), ISO Geneva, 1992.

[ISO646] International Organization for Standardization.
Information processing – ISO 7-bit coded character set for information interchange.
ISO 646.

[ISO/R233] International Organization for Standardization.
International System for the Transliteration of Arabic Characters.
ISO/R 233 - 1961.

[Lag92a] Klaus Lagally.
ArabTEX, a System for Typesetting Arabic.
In *ICEMCO92, Proc. 3rd International Conference and Exhibition on Multi-lingual Computing (Arabic and Roman Script)*, pages 9.4.1–9.4.8, University of Durham, UK, December 10–12, 1992.

[Lag92b] Klaus Lagally.
ArabTEX, a System for Typesetting Arabic.
Report 1992/11, Universität Stuttgart, Fakultät Informatik, 1992.

[Lag93a] Klaus Lagally.
ArabTEX, a System for Typesetting Arabic. User Manual Version 3.00.
Report 1993/11, Universität Stuttgart, Fakultät Informatik, 1993.

[Lag93b] Klaus Lagally.
Some Problems in Arabizing LATEX.
Report 1993/15, Universität Stuttgart, Fakultät Informatik, 1993.

[Lag94] Klaus Lagally.
Some Problems in Arabizing LATEX.
In *ICEMCO94, Proc. 4th International Conference and Exhibition on Multi-lingual Computing (Arabic and Roman Script)*, pages 9.10.1–9.10.8, London, April 7–9, 1994.

[Lag95a] Klaus Lagally.
On the Use of Symbolic Markup in the Production of a Multi-Lingual Dictionary.
Report 1995/12, Universität Stuttgart, Fakultät Informatik, 1995.

[Lag95b] Klaus Lagally.
Processing Hebrew with ArabTEX.
`ftp://ftp.informatik.uni-stuttgart.de/pub/arabtex/hebrew.305`, 1995.
Short introduction.

[Lag96] Klaus Lagally.
On the Use of Symbolic Markup in the Production of a Multi-Lingual Dictionary.
In *ICEMCO96, Proc. 5th International Conference and Exhibition on Multi-lingual Computing (Arabic and Roman Script)*, pages 3.17.1–3.17.10, Cambridge, UK, April 11–13, 1996.

[Lag97a] Klaus Lagally.
ArabTEX Version 3 Overview.
`ftp://ftp.informatik.uni-stuttgart.de/pub/arabtex/arabtex.htm`, 1997.
Short introduction.

[Lag97b] Klaus Lagally.
Sindhi in ArabTEX.
`ftp://ftp.informatik.uni-stuttgart.de/pub/arabtex/sindhi.tex`, 1997.
Short introduction.

[UNICODE] The Unicode Consortium.
The Unicode Standard. Worldwide Character Encoding. Version 1.0, Volume 1.
Addison-Wesley, Reading, Mass., 1991.

[USMARC90] Library of Congress, Cataloging Distribution Service, Washington.
USMARC Specifications for Record Structure, Character Sets, Tapes, 1990.

---

[1] Klaus Lagally, born in Munich (GERMANY) in 1937. University studies in Mathematics and Physics, Ph.D. in Theoretical Physics 1967. Work on Operating Systems and Programming Languages. Professor of Computer Science 1976, Universität Stuttgart, Germany. Current research interests: Arabic text processing and typesetting, multi-lingual computing and communication.

[2] For Arabic this task, whose difficulty should not be under-estimated, has been tackled successfully as early as 1935 by the famous "Denkschrift" to the International Congress of Orientalists, which later led to [DIN31635] and [ISO/R233]. However, this is not a pure transliteration, but a hybrid between a transliteration and a transcription.

[3] ArabTEX is available via `ftp://ftp.informatik.uni-stuttgart.de/pub/arabtex`