

Phonetic Fonts and Phonetic Data Encoding

Peter Constable,
Non-Roman Script Initiative, SIL International
peter_constable@sil.org

Paper presented at the workshop on
[Web-Based Language Documentation and Description](#)
12-15 December 2000, Philadelphia, USA.

Abstract

For a common archive of linguistic data materials to succeed, it is essential to have interoperable solutions for working with phonetic text representations. This involves a common, interchangeable encoding of phonetic character data, and commonly available fonts and rendering technologies for displaying phonetic character data. This paper will discuss these issues, presenting the challenges to be solved, and investigating best common practices for character encoding of phonetic data. I will also discuss work in progress within SIL to provide font and rendering solutions, and will explore some possibilities to further extend the capabilities that our current work is providing.

1. INTRODUCTION

Information technologies for working with multilingual text are evolving, and we are currently at a key point of transition: in the past, software has supported 8-bit encodings of text using encoding standards designed for a limited number of major languages only. In addition, these systems often used rendering technologies that were not designed to support complex scripts. In contrast to this, we are entering an era of new technologies that use the Unicode™ Standard for character encoding, and that use “smart-font” rendering technologies to support a wide variety of complex scripts.

For linguists working with text containing phonetic transcription, character encoding and rendering needs were never met by any legacy standards. The lack of industry-standard solutions did not prevent linguists from cobbling together some way to get their work done, but the result was a variety of incompatible, custom solutions. The newer technologies do offer ways to create standards-based solutions that meet these needs, however. At this juncture, there is a need to design new solutions for linguists to work with phonetic transcription data utilizing the advantages of these new technologies in ways that conform to industry standards such as Unicode.

For the purposes of archived linguistic and language data, a documented character encoding for phonetic transcription is essential. It is also important to have documented means for rendering such text so that both those creating the content to be stored in the archive and those that consume the contents are able to work with the data. This document proposes best practices based on current technologies for encoding of archival phonetic transcription data. It also describes work in progress within SIL International to provide a font for phonetic transcription that represents a first step toward adopting new technologies, as well as plans that are being made to extend the capabilities of this font to make it more broadly useful for linguistic and language data.

2. THE PROBLEMS TO BE SOLVED

There are several problems with regard to phonetic transcription data that need to be solved. These fall into the following broad categories: character encoding issues, rendering issues, and other issues.

2.1 Encoding Issues

Existing phonetic fonts assume custom 8-bit encodings, with the result that there is no standard for interchange of phonetic transcription data: the meaning of data is only known when the data is accompanied by a particular font. For purposes of a linguistic archive and for general interchange of linguistic data, a common encoding standard is needed, as well as fonts that are designed to work with that standard. The Unicode Standard provides an ideal character encoding solution: it is an international standard that is receiving broad acceptance within the information technology industry. It is the default encoding assumed by important Internet and Web standards, such as XML. It also a universal character standard that aims to cover all writing systems in the world. If any phonetic symbols are in common usage among linguists but are not currently supported in Unicode, there are clear procedures to get such characters added to the standard. Thus, we need to adopt Unicode as the encoding for phonetic transcriptions, and we need phonetic fonts that conform to the Unicode standard.

Simply deciding to use Unicode does not resolve encoding issues, however. There is a problem of enumerating exactly what set of characters a phonetic font should support. There are two issues here. First, a font can certainly contain IPA characters that are currently supported in Unicode, but many linguists use other conventions for phonetic transcription, and require symbols not found in IPA. The problem here is to figure out exactly what phonetic symbols to include and how they are to be represented in Unicode.

Secondly, whether phonetic symbols are drawn from IPA or from other traditions, it is quite possible that some of the symbols that are required are not yet supported by Unicode. The problem here is how to address that gap. Proposals can be prepared to request that characters be added to Unicode. In the mean time, Unicode “private-use” code points can be used for encoding these symbols, but there is a problem of documenting and having parties agree on such assignments, which is a pre-requisite for data interchange.

There are also some areas in which Unicode offers alternative ways to represent characters, and it is not entirely clear which is the best choice. To give an example, many linguists use superscript digits to represent pitch levels. These could be encoded in Unicode in different ways:

- They can be encoded as directly superscripts using “compatibility” characters like “¹” (U+00B9, SUPERSCRIP ONE).

- They can be encoded as non-compatibility characters using the standard Arabic digits, e.g. “1” (U+0031, DIGIT ONE), and using markup or out-of-band information (e.g. formatting) to indicate that a pitch level is being represented rather than a cardinal numeric value.
- They can be encoded using pitch letters, such as “J” (U+02E9, MODIFIER EXTRA-LOW), with superscript digits being an alternate presentation of the encoded data (possibly handled using font feature mechanisms).

The purpose here is not to present this specific problem, and no further explanation of the issues regarding representation of tones is provided here. Rather, the purpose is to point to a general category of similar problems. There are other such encoding choices that would need to be made for encoding linguistic data, including phonetic transcriptions. These problems will require solutions if linguists are to have common practices that ensure the success of data interchange.

2.2 Rendering Issues

Existing fonts are designed to work with “dumb” rendering systems (systems that do not have any particular support for complex scripts and that assume a one-to-one correspondence between code points in the encoded data and glyphs in a font). These fonts, therefore, require direct encoding of presentation-form glyphs in order to deal with problems of contextual selection and positioning (e.g. selecting a dotless-i glyph when combining with a diacritic, and selecting alternate versions of over-striking glyphs for diacritics to match the widths and heights of base characters). Such encoding meets the needs of rendering, but create problems for most other kinds of processing (e.g. searching).

It is important to understand that Unicode encodes characters, and not glyphs. The direct encoding of presentation forms, as described above, is not supported by Unicode. Rather, Unicode assumes a character-glyph distinction, and assumes implementations that make use of newer “smart” rendering technologies in order to deal with issues related to complex script behaviours. Thus, we need phonetic fonts that are designed to work with “smart” rendering technologies.

There are alternative “smart” rendering technologies that might be used. The difficulties are that the existing alternatives are platform-specific, and not all are readily extensible. Here is a brief summary of the available technologies:

- Apple Computer’s *AAT* (*Apple Advanced Typography*, formerly *TrueType GX*) is an extension to the TrueType font format and is presently available only on the Mac OS. This technology is capable of handling a wide variety of script behaviours, and is readily extensible such that it can be used to handle support for writing systems for which support has not previously been implemented. Currently, creation of AAT fonts requires highly technical and specialized training.

Application support for complex-script rendering using AAT fonts is provided on Mac OS 8.6 and later through the *ATSUI* (*Apple Type Services for Unicode Imaging*) application programming interfaces. On OS X, the “Cocoa” interfaces (based on the NeXT operating system) also make use of AAT fonts. At present, few applications exist that make use of ATSUI or “Cocoa” interfaces, however, although Apple is working hard to make this easier for application developers.

- Microsoft and Adobe have collaborated on the development of the *OpenType* technology. OpenType is, in a sense, at once an extension of both the TrueType and Postscript Type 1 font formats.

OpenType is supported primarily on Microsoft Windows platforms. Adobe has implemented OpenType support for their own applications on the Mac OS (more on this below). Apple is also considering the possibility of providing support for OpenType at the system level in future versions of the Mac OS.

This font technology does *not* provide complete complex-script rendering support, however. It is dependent on client software having considerable knowledge of script behaviours. Writing individual applications each with hard-wired rendering behaviour for phonetic characters is possible, but very impractical and inefficient. It would also eliminate any possibility of end-user extensibility to support for rare or innovative phonetic characters, and it could easily result in different applications giving different results.

Both Adobe and Microsoft have recognized the problems for developers in adding knowledge of script behaviours directly into their applications, and each chose to develop general solutions for the benefit of their application developers. Adobe's *CoolType* engine is used by various Adobe products, and was developed to ensure consistent cross-platform capability for their products on the Mac and Windows platforms. CoolType is designed primarily to support advanced Roman-script typography in Adobe applications. It does not currently support general stacking of diacritics or most non-Roman scripts, and it is not available for use in non-Adobe applications. There have been no indications that this is likely to change.

Microsoft have solved this problem by developing a system component known as *Uniscribe*. This engine is being implemented at a system level such that at least basic support for non-Roman scripts becomes available to a wide variety of applications without requiring any effort on the part of application developers. (More advanced support does require developers to make use of special programming interfaces.) Uniscribe is only available on the Windows platforms, though, and system-wide support is provided only on Windows 2000.

In current versions, Uniscribe does not support general stacking of diacritics, and it does not provide any mechanisms for extensibility. We expect some support for stacking diacritics to appear before long, however, and it is possible that Microsoft may decide to provide some mechanisms for extensibility. We are hopeful regarding these possibilities, though have no certainty about what capabilities Microsoft software will provide until it is actually shipping.

- SIL has been developing the *Graphite* rendering technology. The primary aim is to provide complex-script rendering support on the Windows platforms that is readily extensible by implementers in field locations with a modest amount of training. Graphite uses an extension of the TrueType font format. It is similar in some important respects to Apple's AAT technology. In particular, it is like AAT, and different from OpenType, in that complete knowledge of script behaviour is encapsulated in the font; it is not necessary for application developers to hard-wire knowledge of these behaviours into their software.

Graphite is also like AAT in that it is readily extensible. What is significantly different, though, is how this is done. AAT requires one to design complex finite-state machines that go in the font. In contrast Graphite provides a high-level description language known as *GDL* (*Graphite Description Language*). GDL is a rule-based language that would be at least somewhat familiar to linguists familiar with the Generative tradition. A compiler is provided that takes a textual description of behaviours written in GDL and compiles this into a finite-state machine that gets embedded inside a TrueType font.

Currently, Graphite is only available on the Windows platforms, and only to applications that have been written to support certain programming interfaces. At present, applications that support Graphite are limited to a suite of linguistic applications currently being developed by SIL. There has been significant interest in Graphite from parties outside SIL, particularly in

relation to possibilities for open-source development and for porting to other platforms, such as Linux. At the time of writing, SIL has not yet made any specific plans to pursue such possibilities.

- As far as we are aware, there are not yet any non-proprietary solutions for complex-script rendering on Unix/Linux platforms. (Some work is in progress, but it is not yet clear how successful these efforts will be and how widely they will be adopted.)
- Sun Microsystems have worked on solutions for complex-script rendering within Java 2. This technology is like Uniscribe in the sense that support for a given script is hard coded and is not currently extensible. Also, current versions do not provide support for stacking diacritics as is needed for phonetic data.

Given this survey of the current state of “smart” font rendering technologies, we see that there is a question of what technologies to adopt. The problem is that there is not one technology that works on all platforms and that provides the kind of extensibility that linguists need in general.

For the particular purposes of web-based archives there are some particular concerns related to rendering. Users wishing to view phonetic transcription data in the archive must have browsers that support Unicode and a “smart” rendering technology, and must have access to phonetic fonts that conform to Unicode encoding and that utilize the rendering technology supported by their browser.

2.3 Other Issues

In addition to the major problem areas described above, there are other problems to be faced, described here.

Input methods will be needed to generate phonetic data from the keyboard. Many Windows users have relied on the Tavultesoft Keyboard Manager (“Keyman”) in the past, and a new version, currently in beta, will make it possible to create keyboard input methods that generate Unicode-encoded data. Keyman 5 is intended to run on Microsoft Windows 95/98/Me, Windows NT 4, and Windows 2000. (Note, however, that there are limitations to what Unicode characters can be entered from the keyboard into some applications, particularly on Windows 95/98/Me.) On other operating system platforms, tools are also needed, and I am not familiar with what may exist for creating keyboards that generate Unicode text.

Whether using Keyman or some other tool, there will be a need to create input methods for entering phonetic data.

As mentioned earlier, various custom, 8-bit encodings have been used in the past for phonetic data. As users begin to work with systems based on Unicode, there will be needs to transfer legacy-encoded data to and from Unicode. Thus, there will be a need for tools for defining mappings between legacy encodings (including custom encodings) and Unicode, and for using these mappings to perform data conversion. SIL International is currently working on tools for the Mac and Windows platforms to handle encoding conversion where custom encodings are involved. There may be other similar tools from other sources.

There is a less-immediate question of what typefaces are needed by linguists for phonetic data. This issue is independent of the technical problems that have been in focus here. As work is being done on creating new solutions to font-related problems, however, that does present an opportunity to consider the choices of typeface that are available. Consideration can also be given to improved font hinting is required in order to provide better legibility on low resolution devices.

3. SIL'S CURRENT WORK ON A UNICODE PHONETIC FONT

SIL is currently working on a new phonetic font that attempts to begin addressing some of the problems described above. It does not presume to solve all of these problems at once, and we assume that additional work on phonetic font solutions will be needed in the future.

The goal of this work is to produce the following items: a TrueType font, a Keyman keyboard, and an encoding conversion mapping table. Details follow:

The font is to have the following specifications:

- Typeface is SIL Doulos (a serif face), in one weight only (regular/"Roman").
- Assumed encoding of data is Unicode.
- Character inventory to be supported includes:
 - all characters included in MS Windows codepage 1252, and
 - all symbols that both (i) are in IPA 1996 (as specified in [1]) and (ii) that are also supported in the Unicode Standard, Version 3.0 [2] (i.e. the intersection of these two sets).
- Complex-rendering (contextual selection of alternate glyphs and diacritic positioning) will be implemented using Graphite technology.
- Diacritic positioning support will include support for arbitrary stacking of diacritics for combinations of up to four diacritics above and up to three below.
- Glyphs for contour tone letters will be included for all contour combinations that are supported in the SIL Encore Font System (all single-, two- and three-pitch contours, though some will be unified; for example a "321" contour will be rendered using the same glyph as a "31" contour). This does not, however, include "left-barred" versions used in Asia for some phonemic transcription. Pitch contours will be mapped as ligatures of single pitch-letter characters (U+02E5–U+02E9) in the rendering process.
- Presentation-form glyphs will not be directly encodable (except for those that are part of MS Windows codepage 1252).
- All characters and glyphs will be supported using a single TrueType font file.

The keyboard will have the following specifications:

- It will be produced using Keyman 5.
- It will be intended for use on MS Windows (various versions).
- It will be capable of generating Unicode character data, and will conform to the Unicode standard. (No specification is made here with regard to Unicode normal forms that it may or may not generate.)
- The layout and behaviour will be based on the keyboard produced by SIL for use with the SIL IPA93 fonts that was, in turn, based on the input behaviour of the SIL FindPhone application.

The encoding conversion mapping table will have the following specifications:

- It will be designed for use with the SIL TECKit encoding conversion tool.
- It will define a mapping between the encoding used in the SIL IPA93 fonts and Unicode.

All of the above pieces will undergo testing on US versions of Windows 98 and Windows 2000.

There may be additional deliverables (e.g. documentation) and functionality provided, but no guarantees are made beyond the specification presented here. Such additional features would only be things that present themselves as clearly useful candidates for addition during the course of

work, that do require any significant investment of additional resources, and that do not impede the completion of the features specified here.

3.1 Benefits of the Current Work Program

Completing the work program described will have several benefits. It will represent initial progress in addressing the problems that have been outlined, and will provide opportunity to evaluate real tools based on the new technologies and so come to a better understanding of the problems and of how to create solutions. It will give developers resources based on the new technology paradigms that they can use for creating and testing new applications. It will provide working tools that early adopters can use to begin creating and editing phonetic transcription data that is encoded using Unicode. It is also possible that the proposed mapping description for SIL IPA93 to/from Unicode will be fully adequate for supporting that legacy encoding and will never need to be revised or replaced.

3.2 Limitations of the Current Work Program

As mentioned above, this proposal does not attempt to solve all of the problems that were mentioned above. The following issues are left for future work:

- Specifying the complete inventory of characters to be support by an “ultimate” phonetic font. This may entail additional type design work to add glyphs for new characters to our existing designs. This may also entail the need to propose addition of new characters to the Unicode standard, and resolving “private-use” encodings for the interim.
- Choosing what “smart” rendering technologies to use on various OS platforms. This may also require resolving how to deal with the lack of availability of any such rendering technology on certain platforms.
- Resolving questions about how to encode certain character sequences for which Unicode provides more than one possible alternative.
- The current work does not address any questions regarding typefaces for use with phonetic data. As mentioned, only one weight and style in one serif face is being produced initially. We certainly anticipate that italic may be needed, and that sans serif or mono-width faces may also be needed. The most obvious alternatives would be for us to produce new fonts based on other faces in the Encore set: Sophia, Manuscript or Charis. It would be possible to consider novel face designs, however.

For users creating content, there is the additional need for tools for creating input methods on each target OS platform. We anticipate that Keyman 5 will be an adequate solution for Windows; solutions may be needed on the Mac OS or Unix/Linux. Of course, the matter of availability of browsers that support Unicode and “smart” rendering technologies on various platforms remains.

4. EXTENDING THE IMPLEMENTATION

Once this initial font is complete and tested, we anticipate making it freely available. For the purpose of common archives on linguistic data, this font will not be fully adequate: there will be needs to overcome some of the outstanding limitations. We in SIL are interested in extending the initial implementation, but we would want to be sure that we provide fonts that serve the broad interests of the linguistic community as best as possible, and we need to take into consideration the costs involved.

SIL has had some preliminary discussions with the TalkBank project regarding these issues. Based at University of Pennsylvania and at Carnegie-Mellon University, [TalkBank](http://www.talkbank.org) (www.talkbank.org) is a major effort funded by the National Science Foundation to develop infrastructure for research into communicative interaction. They have expressed interest in working with us to provide Unicode-conformant phonetic fonts as a resource for the linguistics community. In this partnership, SIL would function primarily as a vendor, creating the fonts, and TalkBank would function primarily as client, providing funding and interacting with the community at large to define requirements.

It will be necessary to work out a specification of requirements in each of three areas. The first is the choice of rendering technologies to be supported. In this regard, it is possible to create fonts that will work with either OpenType, AAT or Graphite (that is, the same fonts can work with any of the three rendering technologies). We have anticipated creating fonts that work with all three technologies.

The second area of requirements has to do with the exact selection of faces, weights and styles to be provided. This part of the specification will have the biggest impact on the costs involved, and so will need careful consideration.

The third aspect of the requirements will be to specify the complete inventory of characters to be supported. For the initial font, this was a very easy task since a well-defined inventory was used, the IPA 1996 specification. Extending this inventory will be the most difficult aspect of the specification since it is not clear in advance what should and should not be included. This part of the specification will have some affect on the costs, especially if new type design is required (i.e. if characters are needed that are not already supported by the Encore collection).

5. CONCLUSION

In conclusion, it is recommended as best practice for archived phonetic transcription data that the data be encoded using Unicode and viewed using Unicode-conformant fonts that utilize some “smart” rendering technology. We anticipate that such fonts will be provided to the linguistics community by SIL and TalkBank. There are certain outstanding issues that remain to be resolved, one of the most important being to identify an inventory of phonetic symbols to be supported by these fonts. There are also some questions regarding exactly how some phonetic transcriptions should be represented in Unicode, though this is a general issue that affects many writing systems that may be represented in language archives beside phonetic transcriptions.

There is a larger problem of needing client software that supports “smart” rendering technologies on various operating system platforms. Solving that problem is beyond the scope of this discussion, however.

6. REFERENCES

- [1] *The International Phonetic Association. 1999. Handbook of the International Phonetic Association. Cambridge: Cambridge University Press.*
- [2] *The Unicode Consortium. 2000. The Unicode standard, version 3.0. Reading, MA: Addison-Wesley.*