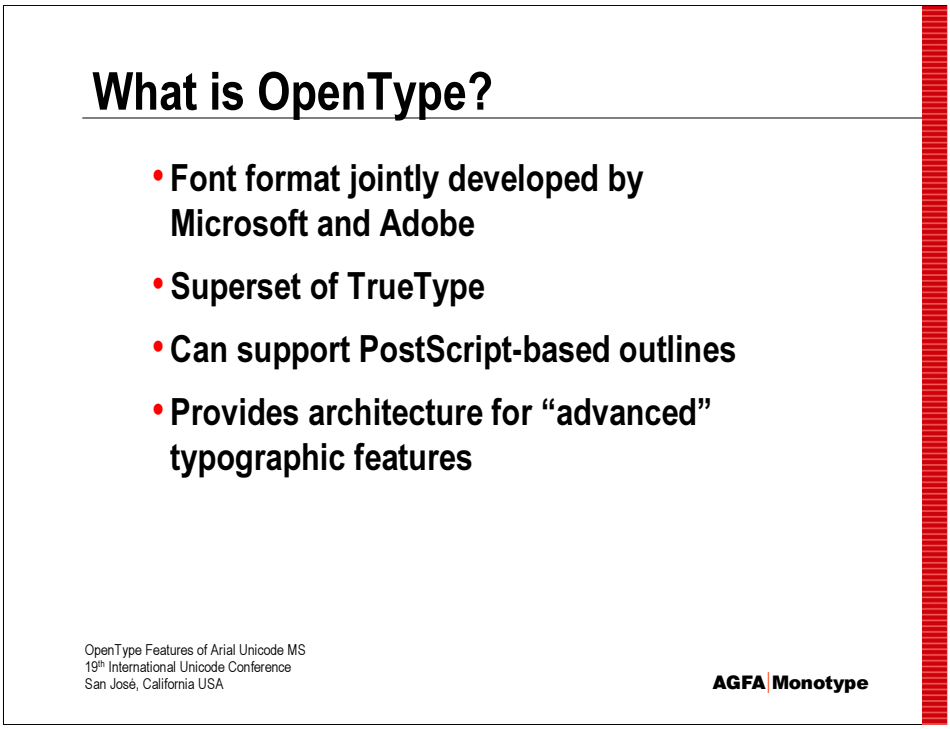
OpenType Features of Arial Unicode MS

Presented by

Joshua Hadley

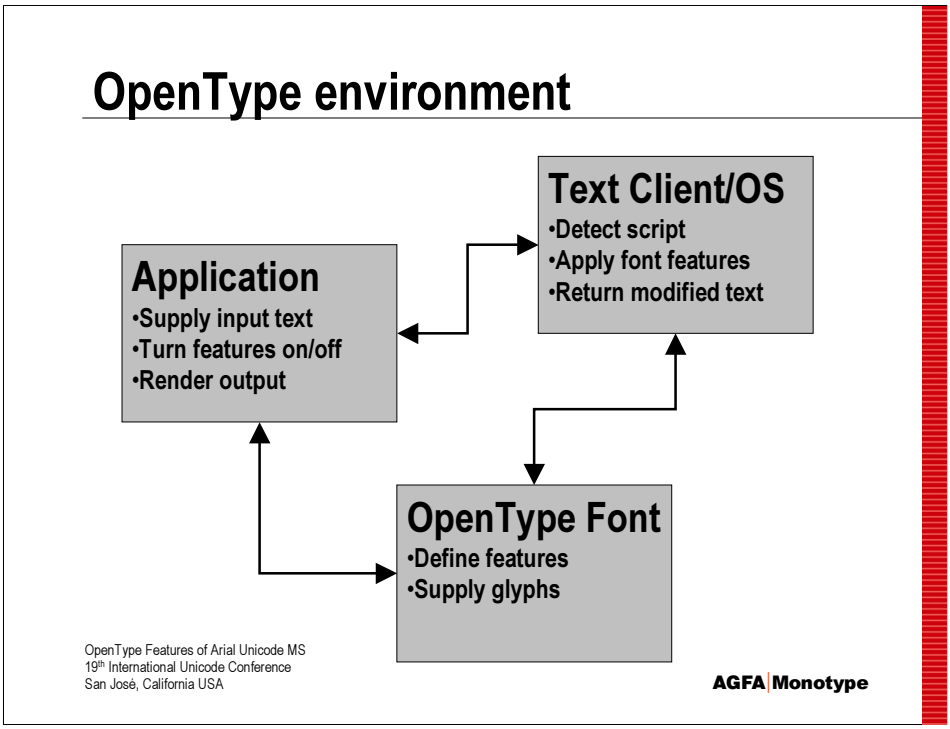
Agfa Monotype Corporation



OpenType is a font format jointly developed by Microsoft and Adobe. In 1998, the two companies joined forces to create a format that could simultaneously support the two most popular outline font formats: PostScript and TrueType. Microsoft's previous work on TrueType Open – mostly consisting of layout and "advanced typography" features – was also rolled into OpenType.

In fact, the OpenType format is actually an extension of the existing TrueType format. The file structure is identical, using a table-based architecture, and most existing TrueType fonts can technically qualify as OpenType merely by the addition of a Digital Signature ('DSIG') table.

PostScript fans are not quite as lucky: though a form of PostScript outlines are supported, it is not the familiar .PFB/Type 1 file format that many users have become accustomed to. However, Adobe has developed the Compact Font Format, or CFF, which is based on PostScript. Users and font developers should find that most Type1 fonts can easily be converted from Type 1 to CFF using Adobe-provided font tools.

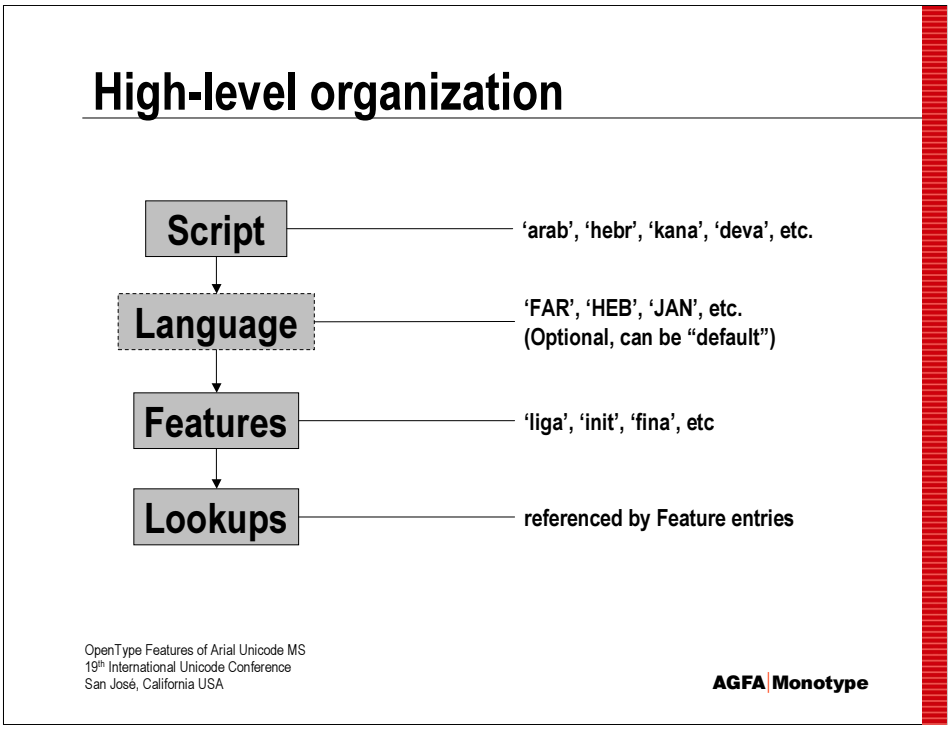


Under OpenType, there is a complex interaction of Application, Font, and Operating System. Each component of the OpenType environment needs to perform specific duties in order to make it all work.

The application supplies the input text (as entered by the user) to the Operating System/Text Client. The application can optionally query the font and override default features and activate or deactivate a font's OpenType features as needed. Once the processed text is returned from the Text Client, the Application can display it.

The Text Client – usually a part of the Operating System, known under Windows as USP10.DLL or "UniScribe" – is responsible for analyzing input text from the Application. It also queries the font so it knows which features are available. Upon completing the analysis of input text and applying available font features, it can return modified "output" text to the Application.

The OpenType font is the most static of the bunch. While the Application and Text Client are busy swapping text, the font basically exists to be read. It can be thought of as a repository for glyphs and feature lookups. But of course, without it there would be no display of text at all!



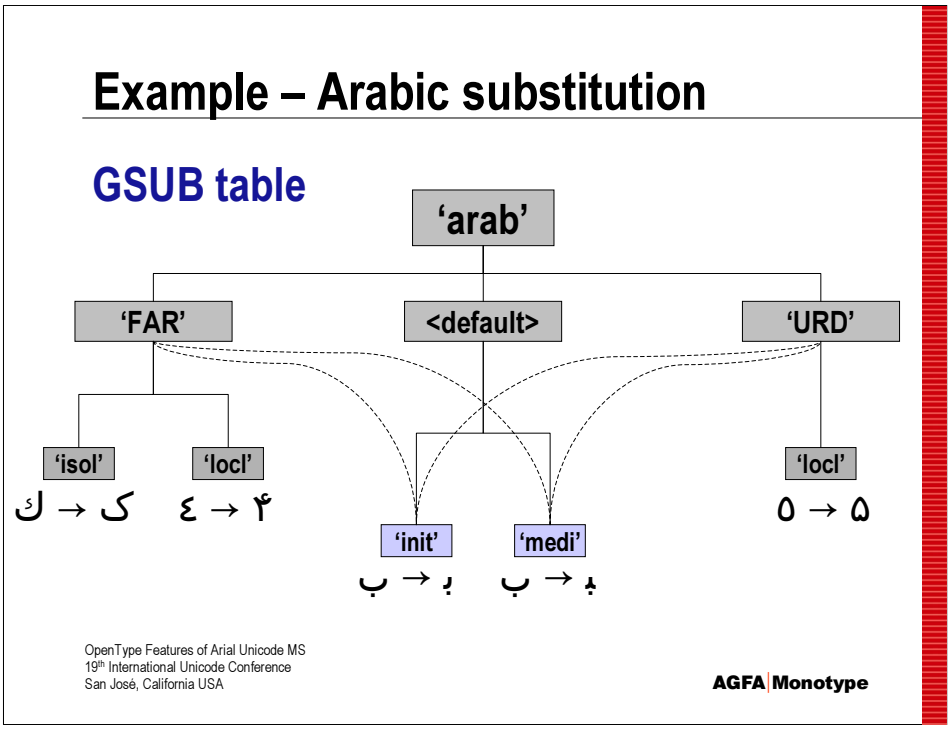
Within OpenType Layout tables ('GSUB' and 'GPOS'), there is a hierarchy for organizing the various features that can be applied.

At the top is the Script, or "system of writing". In OpenType, scripts are identified by four-letter codes such as 'arab' for Arabic, 'hebr' for Hebrew, 'kana' for Katakana, and so on.

Since many scripts are used to write multiple languages, the next level is Language. It is not required to sub-classify a feature with a language, but the option exists for cases where there are minor modifications to the writing system that vary with language. In OpenType, Languages are identified by three-letter codes such as 'FAR' for Farsi, 'HEB' for Hebrew, etc. [Note: currently, UniScribe does not support languages other than default].

At the next level are Features, which are identifiers of the "rules" or system for a particular script. Similar to Scripts, Features in OpenType are identified by four-letter codes. For example, in Arabic, the behavior of modifying a character's shape based on its position in a word is a "rule"; this is further broken down into Initial ('init'), Medial ('medi'), and Final ('fina') positional variants. There are many feature codes defined for many different general features of various languages.

Finally, there are Lookups. Lookups are the workhorses of OpenType features, and are basically lists describing how to transform input glyphs to output glyphs. There are several different types of lookups, including single/simple substitution (e.g. "for input glyph 1, substitute output glyph 4") as well as multiple, ligature, and class-based substitution. Lookups are simply indexed (e.g. Lookup 1, Lookup 2, etc.) rather than tagged, which allows them to be "feature independent". Thus it is possible to share Lookups among different features.



In this extremely simplified example, we have a 'GSUB' (Glyph Substitution) Table with Arabic ('arab') script. At the next level are the languages (Farsi and Urdu), as well as the default. Next come the feature tags ('init', 'medi', etc.), and finally, the lookups.

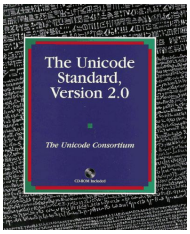
Notice that Farsi and Urdu are "sharing" the default language system's lookups for 'init' and 'medi', since these do not need to change from default.

NOTE: This example has been simplified for purposes of illustration. In a real implementation, the pointers from Farsi and Urdu to the default language features would point to the *lookups*, not to the feature tags as suggested by the graphic.

Arial Unicode MS in brief

Why Arial "Unicode"?

- **Contains glyph shapes to represent all 38,887 [non-PUA, non-Supplementary, non-Control] code points of The Unicode Standard, Version 2.1 – in a single binary TTF file.**

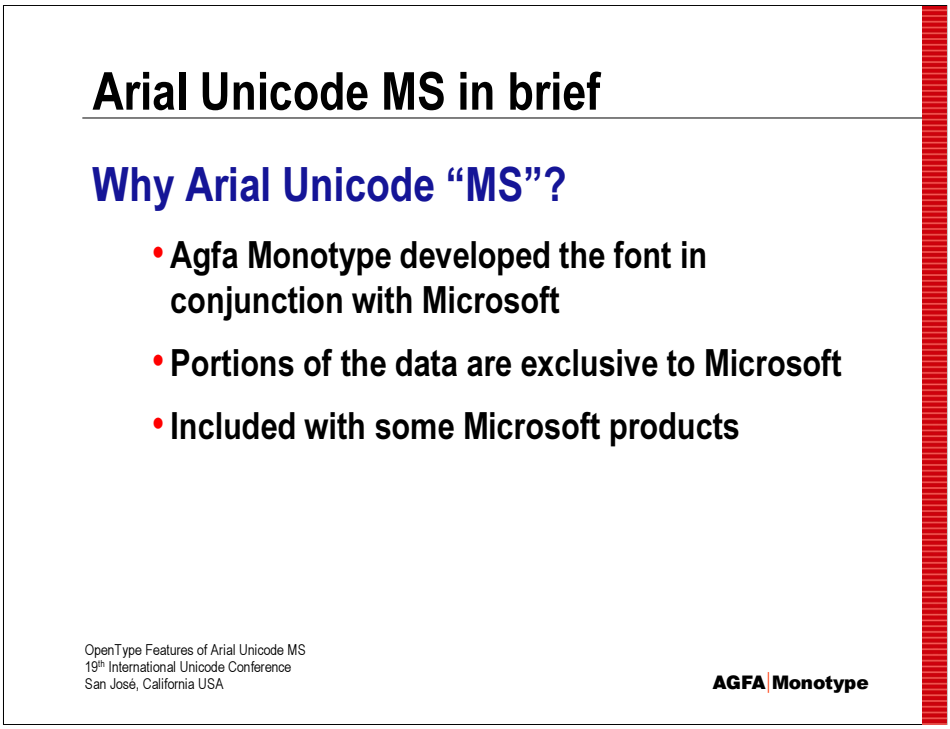


- **Uses Unicode encoding scheme**

OpenType Features of Arial Unicode MS
19th International Unicode Conference
San José, California USA

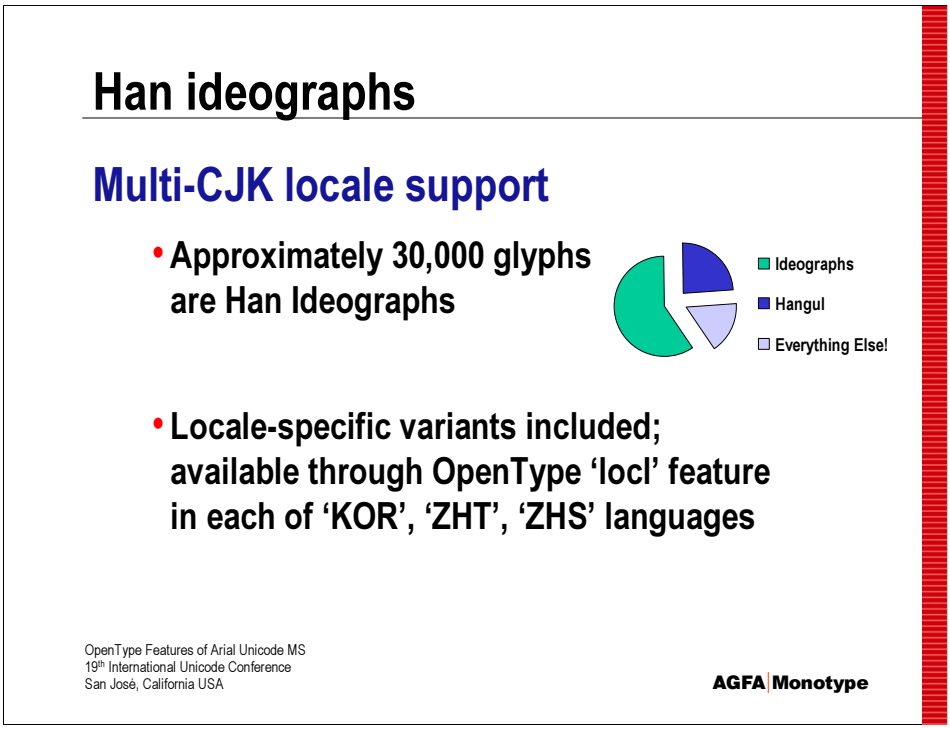
AGFA | Monotype

Arial Unicode MS contains over 50,000 glyphs – enough to represent each one of the 38,887+ code points of The Unicode Standard, Version 2.1 (and then some). The font includes a Unicode Character Mapping table and glyphs within the font take on properties as defined by The Unicode Standard.



The "MS" designator of Arial Unicode MS indicates that it is officially a Microsoft product. The font was a joint development effort between Agfa Monotype and Microsoft, with Microsoft supplying specifications and Agfa Monotype cobbling together the needed glyph data, TrueType instructions (hinting), as well as developing the OpenType layout information specific to the font's design & glyph repertoire.

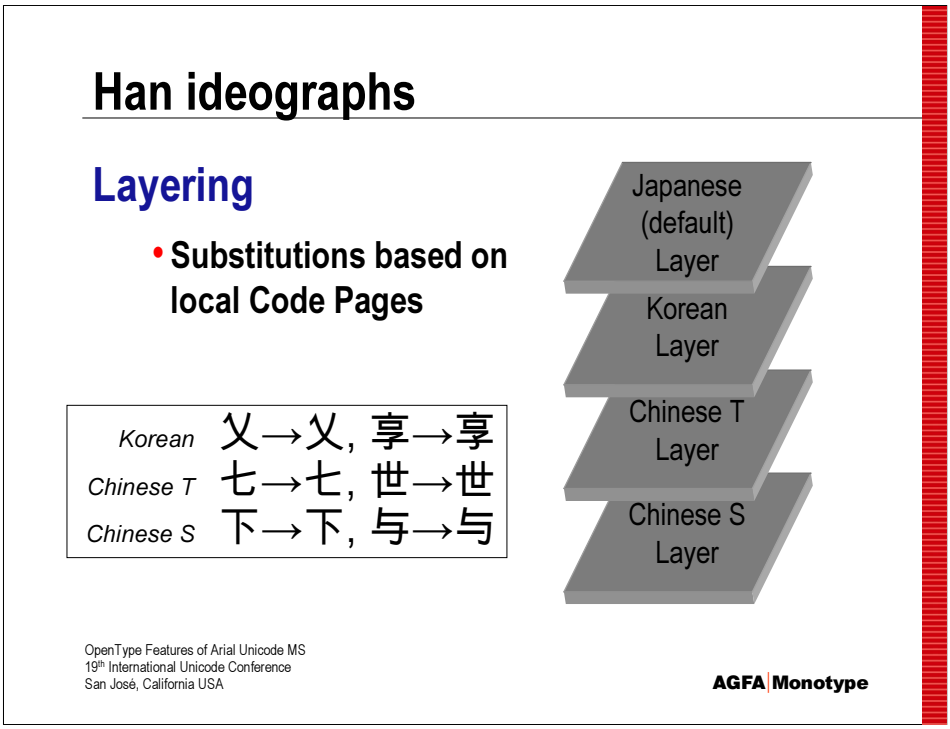
Some portions of the font data are licensed exclusively to Microsoft, meaning that the font as a whole is not available for general distribution. It is available only through Microsoft (currently provided with the Office 2000 installation, as well as an update via Microsoft's web site).



One of the more interesting of the OpenType features in Arial Unicode MS is the multi-CJK locale support. The Han Unification process undertaken by The Unicode Consortium did a great deal to simplify the encoding of the tens of thousands of CJK glyphs. However, Unification could not eliminate existing writing style preferences among the various CJK locales. Thus, it became necessary to provide up to four glyphs per Unicode code point, plus OpenType features to support substitution, in order to accommodate the various CJK locales in a single font file.

As a result, over 30,000 of the 50,000+ glyphs in the font are CJK ideographs. Of these, 20,902 are assigned Unicode code points to serve as "default" characters. The remaining locale-specific glyphs are unencoded, but accessible via the OpenType 'locl' feature.

The 'locl' feature appears under the Korean ('KOR'), Traditional Chinese ('ZHT') and Simplified Chinese ('ZHS') language tags under 'hani' script in the GSUB table.



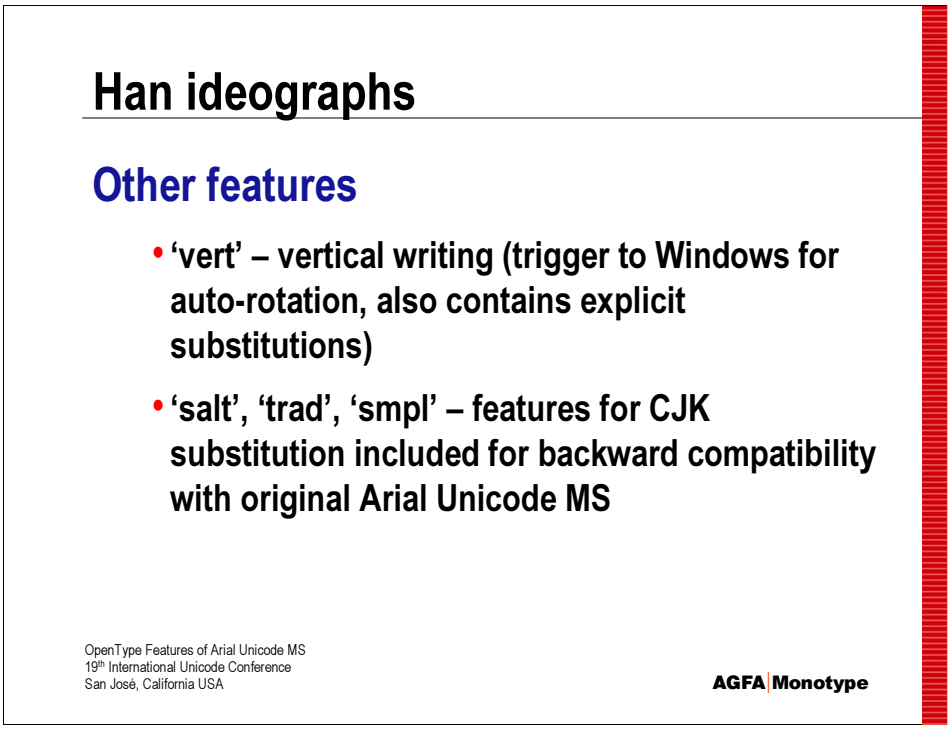
To minimize the glyph repertoire, it was decided to "layer" the Han ideographic portion of the glyph data, and encode characters based on a preferential order of, and appearance within, local code pages. The order of preference for encoding characters was: Japanese, Korean, Traditional Chinese, Simplified Chinese. Since not all local code pages include each of the 20,902 Han ideographs, and not all characters have writing style distinctions, things got a bit complicated!

The procedure for encoding/substituting was something like this:

- For a given code point, if it is in the Japanese code page, assign the Unicode code point.
- If there are other glyphs in other code pages conceptually corresponding to this code point but with different writing styles, create an OpenType entry for the locale pointing from this character to the other glyph(s).

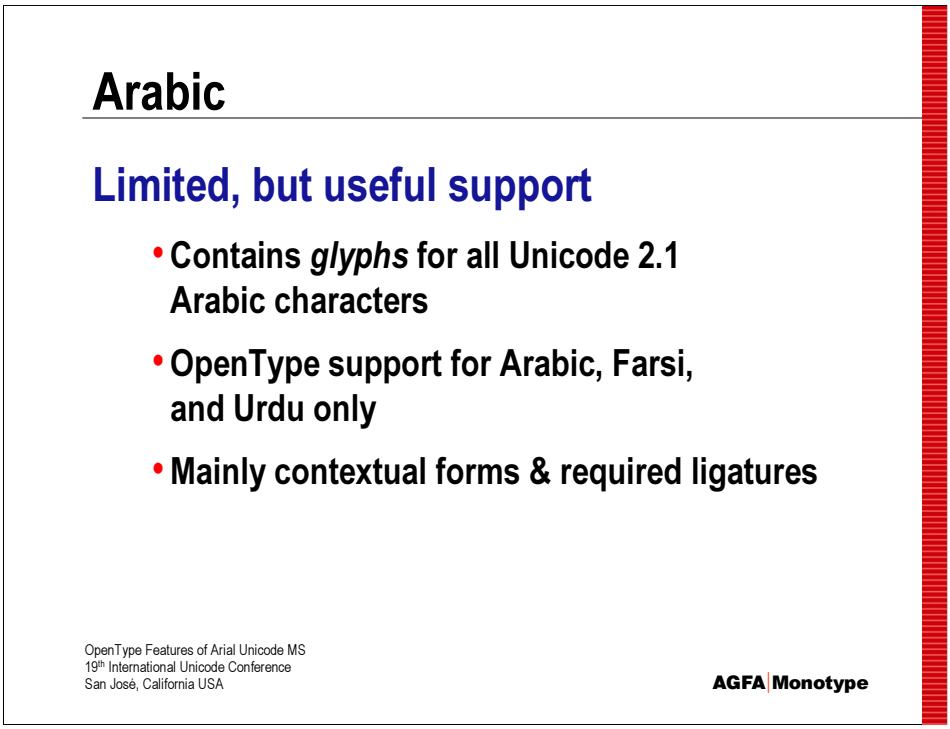
Once all code points in the Japanese code page were exhausted, the procedure moved through successive locale layers with a similar procedure.

As a result, all glyphs in the Japanese code page receive the explicit Unicode code point in the character mapping table, and a locale-specific substitution for Japanese is unnecessary – this is why the font seems to lack Japanese features. A given character from other code pages may or may not receive an explicit Unicode code point depending upon whether it appears in the Japanese (or other layer above) and whether it has a different writing style.



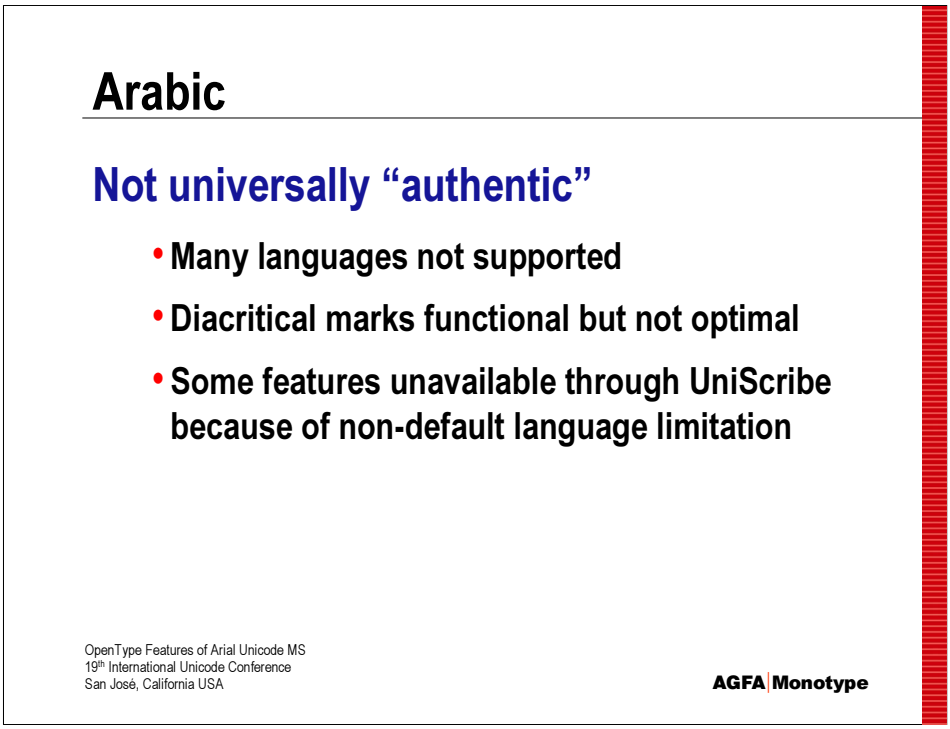
Arial Unicode MS also includes a 'vert' feature under 'hani' script. This feature contains lookups for about 50 glyphs, to provide correctly rotated and positioned glyphs for Vertical Writing mode. The 'vert' feature tag also serves as a trigger to Windows Far East operating systems to activate a special function that will automatically rotate ideographs and other related characters, without having explicit substitutions present in the lookup.

Finally, 'salt', 'trad', and 'smpl' features are included under 'hani' script, to provide compatibility with an earlier version of Arial Unicode MS. The features use the same lookups as the 'locl' features, only the tag is different. Use of these features is discouraged, and they may eventually be phased out – however, it is currently the only mechanism supported by UniScribe, because they appear under the default language system.



Though not originally specified as a publishing-quality Arabic font, Arial Unicode MS does contain a surprising amount of Arabic-related OpenType features. At minimum, there is a glyph to represent all of the Arabic characters defined within Unicode 2.1, so if nothing else, a default and understandable representation of each character is available.

OpenType support – for contextual variants, required ligatures, and diacritic positioning – is included for all characters used for Arabic, Farsi, and Urdu languages. For these languages, Arial Unicode MS is suitable for use much the same as other “standard” Arabic fonts.

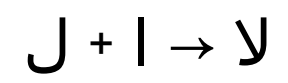


Because of time and space constraints, Arial Unicode MS does not include contextual variants and limitless ligature information for all characters within Arabic script. For example, “U+06B7 ARABIC LETTER LAM WITH THREE DOTS ABOVE” (Kurdish), has only a single variant available.

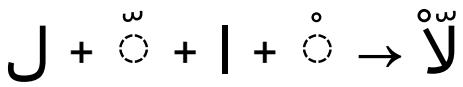
Support for proper placement of diacritical marks is available, but in some cases, not optimal. Collisions with some characters can occur, and overall positioning could be improved in some cases.

Finally, under Windows/UniScribe, some features associated with specific languages (Farsi and Urdu) are not available because UniScribe only supports the default language.

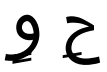
Arabic Examples



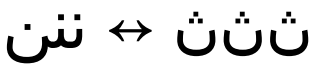
required ligature



ligature + diacritics



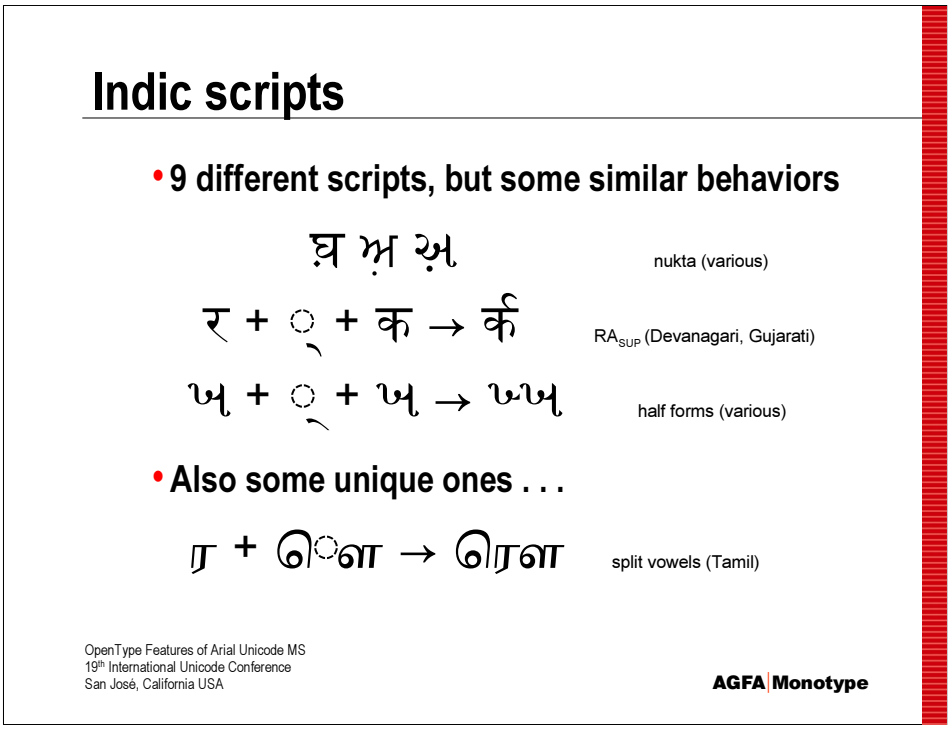
sub-optimal diacritic placement



missing language support

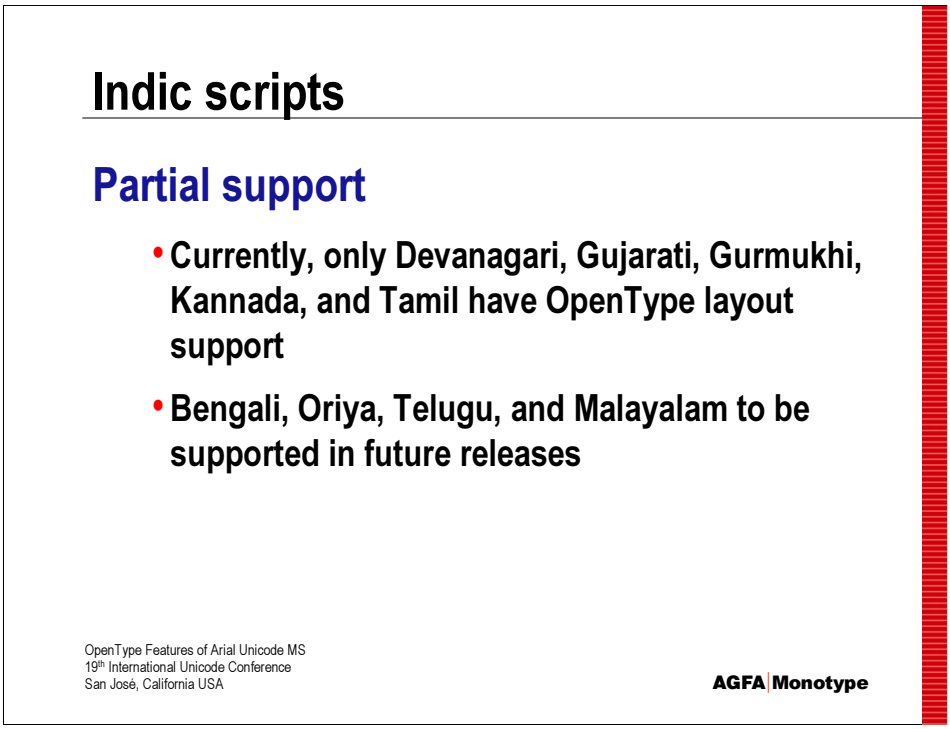
OpenType Features of Arial Unicode MS
19th International Unicode Conference
San José, California USA

AGFA | Monotype

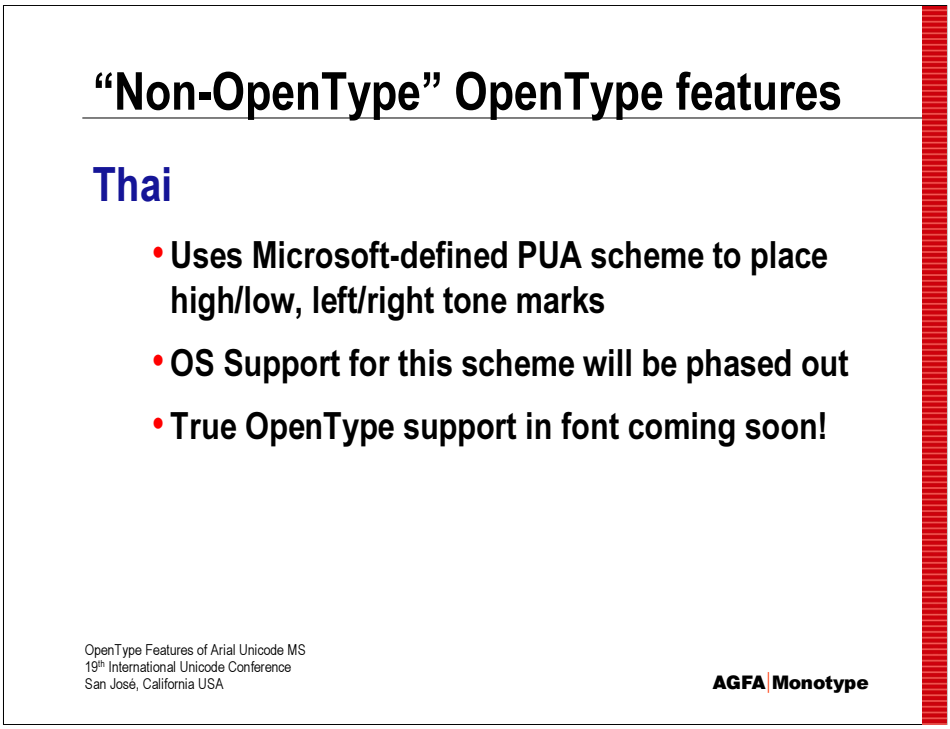


Some of the most complex OpenType features of Arial Unicode MS are within the Indic scripts. Fortunately, many of the concepts are similar among the nine different scripts. Most make use of a nukta consonant modifier, and many use half-consonant forms. Once a glyph repertoire for each script was decided, the common concepts were easier to "port" from one Indic script to another.

However, several scripts require unique features. In the example above, we see the Tamil AU vowel sign, a character which surrounds the associated consonant. Other Indic scripts have features unique unto themselves, requiring slightly different techniques for each one.



With the current release of Arial Unicode, only Devanagari, Gujarati, Gurmukhi, Kannada, and Tamil scripts have OpenType support. This will change in future releases and will be largely tied to support provided in UniScribe.



Current support of Thai in Arial Unicode MS relies on a scheme using the Private Use Area of Unicode (range U+F700 – F71c). Support consists of variant glyphs for high/low and left/right tone marks, and a system-level remapping of input codes from the Thai range (U+0e00 – 0e7f) to the PUA codes as needed, depending on context.

Microsoft has recently announced that support for this scheme is being phased out of Windows, so future versions of Arial Unicode are slated to have true OpenType support, most likely consisting of class-based substitutions and/or positioning features.

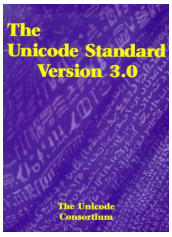
Future plans

Glyph support for Unicode 3.0+

- **Many scripts will be "chart support" only**
- **Cannot support all of Unicode 3.1 in a single font file because of OpenType format limitations**

OpenType Thai & Hebrew

- ขอขอบพระคุณ בְּרֵאשִׁית

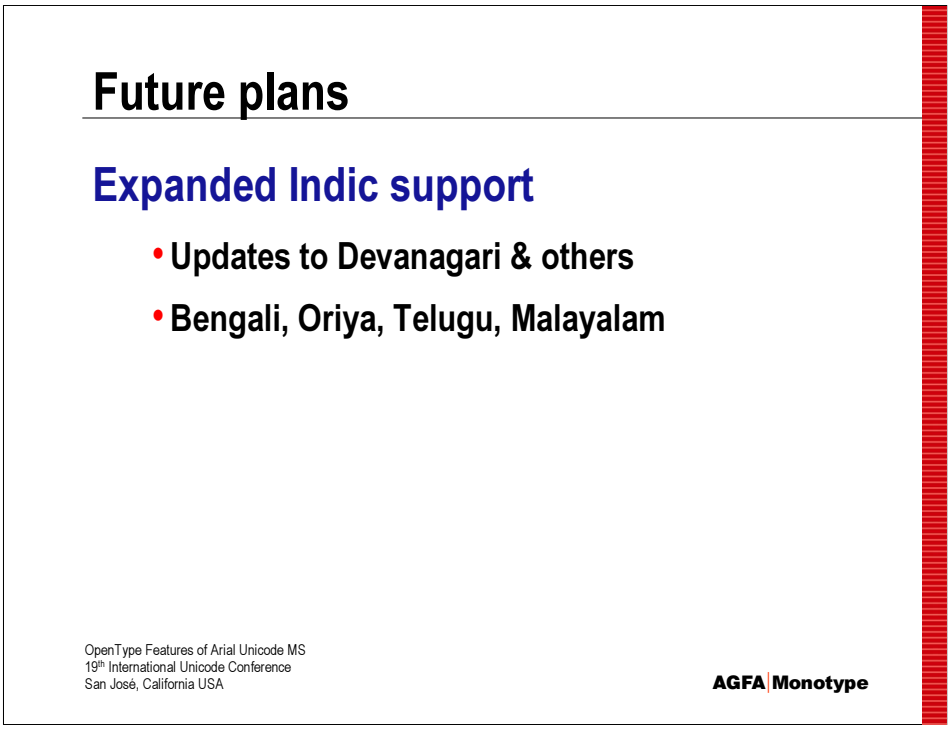


OpenType Features of Arial Unicode MS
19th International Unicode Conference
San José, California USA

AGFA | Monotype

With the current OpenType font format, it is possible to include up to 65,535 glyphs in a single font. Arial Unicode MS – supporting Unicode 2.1 - currently contains just over 50,000, leaving 15,000 or so available positions for "upgrades". Adding "chart-level" support for The Unicode Standard, Version 3.0 should be possible without any problems.

However, with the addition of the CJK Unified Ideographs Extension B, the total number of Unicode Code points exceeds 65,535. This means that even with a one-to-one character-to-glyph mapping scheme, it is not possible to provide support for "all of Unicode" in a single OpenType font. The most likely scenario is that some sort of higher-level font grouping scheme will be implemented, and the font will be divided into 2 or more files – most likely with the BMP characters in one file, and Supplementary Characters in an auxiliary file.



In addition to expanding existing features for Indic scripts, new support for Bengali, Oriya, Telugu, and Malayalam scripts is planned for future releases of Arial Unicode MS.

Demo, Questions & Answers, etc.