To the BMP and beyond!

Eric Muller
Adobe Systems
Content

1. Why Unicode
2. Character model
3. Principles of the Abstract Character Set
4. The characters in 5.0
5. Development of the standard
6. Processing
7. Unicode and other standards
8. Resources
Part I

Why Unicode
ASCII

• 128 characters
  \[ A = 41 \]
• supports meaningful exchange of text data
• very limited: not even adequate for English:
  
  \[
  \begin{align*}
  &\text{Adobe®} \\
  &\text{he said “Hi!”} \\
  &\text{résumé†} \\
  &\text{cañon}
  \end{align*}
  \]
Many other standards

• national or regional standards
  ISO-Latin-1/9: targets Western Europe
  JIS: targets Japan
• platform standards
  Microsoft code pages
  Apple: MacRoman, etc.
  Adobe: PDFDocEncoding, etc.
• but none for many writing systems!
Unicode

- enables world-wide *interchange* of data
- contains all the *major living scripts*
- *simple enough* to be implemented everywhere
- supports *legacy data* and implementation
- allows a *single* implementation of a product
- supports *multilingual* users and organizations
- conforms to *international standards*
- can serve as *the foundation* for other standards
Part II

Character model
Four layers

- abstract character set
  smallest components of written language
- coded character set
  adds name and code point
- character encoding forms
  representation in computer
- character encoding schemes
  byte serialization
Abstract character set

- character: *the smallest component of written language that has semantic value*
- wide variation across scripts
  - alphabetic, syllabary, abugidas, abjad, logographic
- even within scripts, e.g. “ch”:
  - two components in English
  - one component in Spanish?
- abstract character:
  *a unit of information used for the organization, control, or representation of textual data.*
Coded character set

- give a name and a code point to each abstract character
- name: LATIN CAPITAL LETTER A
- code point: pure number, no computer connection
  legal values: U+0000 - U+10FFFF
  space for over a million characters
- characters specific to a script mostly grouped
17 Planes

- 17 planes of 64k code points each
- plane 0: Basic Multilingual Plane (BMP, 1.0) frequent characters
- plane 1: Supplementary Multilingual Plane (SMP, 3.1) infrequent, non-ideographic characters
- plane 2: Supplementary Ideographic Plane (SIP, 3.1) infrequent, ideographic characters
- plane 14: Supplementary Special-purpose Plane (SSP, 3.1)
- planes 15 and 16: Private use planes (2.0)
Private Use Area

- for your own characters; will never be assigned
- must agree on the meaning of those code points
- Unicode does not provide a mechanism to do so
- very delicate to use
  - avoid it if possible
- distribution:
  - U+E000 - U+F8FF: 6,400 in the BMP
  - U+F0000 - U+FFFFF: 64k in plane 15
  - U+100000 - U+10FFFF: 64k in plane 16
Surrogate code points and scalar values

• Unicode was originally defined as a “16 bit character set”
• in 1996 (Unicode 2.0), realized that this was not enough
• code points set aside: surrogates code points
  U+D800 - U+DBFF: 1,024 high surrogates
  U+DC00 - U+DFFF: 1,024 low surrogates
• remaining code points: scalar values
  U+0000 - U+D7FF
  U+E000 - U+10FFFF
• surrogates code points must never appear in data
Character encoding forms: UTFs

- the representation of *scalar values* in computers
- each scalar value represented by a sequence of *code units*
- three forms, defined by:
  - size of the underlying code unit (8, 16, 32 bits)
  - method to convert a scalar value to code units
- all three forms can represent all scalar values and only the scalar values
- no escapes, self-synchronizing
UTF-8

- 8 bit code units, 1 to 4 units

<table>
<thead>
<tr>
<th>USV</th>
<th>Unit 1</th>
<th>Unit 2</th>
<th>Unit 3</th>
<th>Unit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-007F</td>
<td>0xxxxxxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0080-07FF</td>
<td>110xxxxx</td>
<td>10xxxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0800-D7FF</td>
<td>1110xxxxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
<td></td>
</tr>
<tr>
<td>E000-FFFF</td>
<td>1110xxxxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
</tr>
<tr>
<td>10000-10FFFF</td>
<td>11110xxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
</tr>
</tbody>
</table>

- e.g. \(F03F_{16} = 1111\ 0000\ 0011\ 1111_{12}\)
  \[\rightarrow 11101111\ 10000000\ 10111111_{16} = \text{EF}\ 80\ \text{BF}_{16}\]

- use this table strictly
- coincides with ASCII
- mostly found in protocols and files
UTF-16

• 16 bit code units, 1 or 2 units

<table>
<thead>
<tr>
<th>USV</th>
<th>Unit 1</th>
<th>Unit 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-D7FF</td>
<td>xxxxxxxxxxxxxxxxxxxxxxxxxx</td>
<td></td>
</tr>
<tr>
<td>E000-FFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10000-10FFFF (10000 bias)</td>
<td>110110xxxxxxxxxxxx</td>
<td>110111xxxxxxxxxxxx</td>
</tr>
</tbody>
</table>

• takes advantage of gap in scalar values
  110110xxxxxxxxxxxx = D800 - DBFF
  110111xxxxxxxxxxxx = DC00 - DFFF

• because of frequency of BMP, efficient
• appropriate for applications
• UCS-2 is ISO term for UTF-16 restricted to BMP
UTF-32

- 32 bit units, 1 units

<table>
<thead>
<tr>
<th>USV</th>
<th>Unit 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-D7FF</td>
<td>000000000000xxxxxxxxxxxxxxxxxxxxxxx</td>
</tr>
<tr>
<td>E000-10FFFFF</td>
<td></td>
</tr>
</tbody>
</table>

- convenient, but expensive
- rarely used
- UCS-4 is the ISO term for UTF-32
Character encoding schemes

- mapping of code units to bytes
- UTF-8: obvious
- UTF-16LE
  - little endian
  - initial FF FE (if present) is a character
- UTF-16BE
  - big endian
  - initial FE FF (if present) is a character
- UTF-16
  - either endianness
  - may have a BOM: FF FE or FE FF, not part of text
  - if no BOM, then must be BE
- UTF-32: similarly, UTF-32LE, UTF-32BE and UTF-32
Character Latin A

- abstract character: the letter A of the Latin script
- coded character:
  - name: LATIN CAPITAL LETTER A
  - code point: U+0041
- encoding forms:
  - UTF-8: 41
  - UTF-16: 0041
  - UTF-32: 00000041
Character Hiragana MA

• abstract character:
  the letter ま of the Hiragana script
• coded character:
  name: HIRAGANA LETTER MA
  code point: U+307E
• encoding forms:
  UTF-8: E3 81 BE
  UTF-16: 307E
  UTF-32: 0000307E
Character Deseret AY

- abstract character: the letter /ay/ of the Deseret script
- coded character:
  - name: DESERET CAPITAL LETTER AY
  - code point: U+1040C
- encoding forms:
  - UTF-8: F0 90 90 8C
  - UTF-16: D801 DC0C
  - UTF-32: 0001040C
## Terminology

<table>
<thead>
<tr>
<th>Basic Type</th>
<th>Character status</th>
<th>Code point status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphic</td>
<td>Assigned to abstract character</td>
<td>Designated (assigned) code point</td>
</tr>
<tr>
<td>Format</td>
<td>Assigned to abstract character</td>
<td>Designated (assigned) code point</td>
</tr>
<tr>
<td>Control</td>
<td>Assigned to abstract character</td>
<td>Undesignated (unassigned) code point</td>
</tr>
<tr>
<td>PUA</td>
<td>Not assigned to abstract character</td>
<td>Undesignated (unassigned) code point</td>
</tr>
<tr>
<td>Surrogate</td>
<td>Not assigned to abstract character</td>
<td>Undesignated (unassigned) code point</td>
</tr>
<tr>
<td>Noncharacter</td>
<td>Not assigned to abstract character</td>
<td>Undesignated (unassigned) code point</td>
</tr>
<tr>
<td>Reserved</td>
<td>Not assigned to abstract character</td>
<td>Undesignated (unassigned) code point</td>
</tr>
</tbody>
</table>

code points = scalar values + surrogates
Part III

Principles of the Abstract Character Set
Principles

- characters, not glyphs
- plain text only
- unification, within each script, across languages
- well-defined semantics for characters
- dynamic composition of marked forms
- equivalence for precomposed forms
- characters are stored in logical order
- round-tripping with some other standards
Characters, not glyphs

- the character U+0041 can equally well be displayed as A, A, A, ...
- sometimes different glyphs are required: U+0647: ٠ ١ ٢ ٣
- going from characters to glyphs: shaping

पूर्ति
Plain text only

- *plain text must contain enough information to permit the text to be rendered legibly, and nothing more*
- e.g. small capitals are not encoded for English
- different requirements for English and IPA
  - U+0262 LATIN LETTER SMALL CAPITAL G
  - voiced uvular stop in IPA
  - not used in English
Unification, within each script, across languages

- no distinction between English \texttt{A} and French \texttt{A} 
  \texttt{U+0041 LATIN CAPITAL LETTER A}
- single, regardless of its usage
- no confusion between Latin \texttt{A}, Greek \texttt{A} and Cyrillic \texttt{A} 
  \texttt{U+0041 A LATIN CAPITAL LETTER A} 
  \texttt{U+0391 A GREEK CAPITAL LETTER ALPHA} 
  \texttt{U+0410 A CYRILLIC CAPITAL LETTER A}
- fairly specific rules for Han unification
  Chinese hanzi
  Japanese kanji
  Korean hanja
  Vietnamese Chữ hán
Well-defined semantics for characters

- the intended use of a character is unambiguous
- the behavior of a character is unambiguous
  - properties
  - algorithms
Dynamic composition

- marked forms are a productive mechanism in writing systems
  - accents in Latin
  - negation in Math
  - vowels in Hebrew and Arabic
  - nukta in Indic scripts
  - etc.
- built from components:
  - \( \acute{e} : \text{U+0065 e U+0301} \)
  - \( \checkmark : \text{U+2208 ∈ U+0338} \)
  - \( \checkmark : \text{U+0921 ॡ U+093C} \)
Dynamic composition (2)

• can have multiple marks
• from base character outwards

\[ \begin{align*}
  & u \ddot{o} \ddot{o} \rightarrow \ddot{u} \\
  & u \ddot{o} \ddot{u} \rightarrow \ddot{u} \\
  & a \cdot \breve{c} \rightarrow \breve{a} \\
  & a \breve{c} \cdot \rightarrow \breve{a}
\end{align*} \]
Equivalence for precomposed forms

- some precomposed characters are encoded
  \( \text{U+00E9 \( \acute{e} \)} \)
- canonical equivalence
  \( \text{U+00E9 \( \acute{e} \) } \equiv \text{U+0065 e U+0301 \( \grave{e} \)} \)
- similarly for Hangul syllables
  \( \text{U+D4DB \( \dot{p} \) } \equiv \text{U+1111 \( \dot{p} \) U+1171 \( \check{t} \) U+11B6 \( \dot{e} \)} \)
- A process shall not assume that the interpretation of two canonically equivalent character sequences are distinct
Characters are stored in logical order

- logical order ~ pronunciation order ~ typing order

<table>
<thead>
<tr>
<th>storage</th>
<th>display</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASDF</td>
<td>ASDF</td>
</tr>
<tr>
<td>גבגש</td>
<td>שלבב</td>
</tr>
</tbody>
</table>

- combining marks (when separate) follow their base character
Round-tripping with some other standards

• the price for acceptance
• often at odds with other principles
• extra characters:
  U+00E9 é, U+FB00 ff, U+F900 壹
• canonical decomposition
  U+00E9 é ≡ U+0065 e U+0301 ⃣
  U+F900 壹 ≡ U+8C48 壹
• compatibility decomposition
  U+FB00 ff ≈ U+0066 f U+0066 f
Part IV

The characters in 5.0
How many?

- 99,024 assigned graphic and format characters
  - 71,226 Han ideographs
  - 11,172 Hangul syllables
  - 16,486 alpha/symbols
  - 140 format characters
- 875,441 reserved, total
  should last a while!
# The scripts

<table>
<thead>
<tr>
<th>Script</th>
<th>IPA</th>
<th>Greek and Coptic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latin</td>
<td>IPA</td>
<td>Greek and Coptic</td>
</tr>
<tr>
<td>Cyrillic</td>
<td>Armenian</td>
<td>Hebrew</td>
</tr>
<tr>
<td>Arabic</td>
<td>Syriac</td>
<td>Thaana</td>
</tr>
<tr>
<td>Bengali</td>
<td>Gurmukhi</td>
<td>Gujarati</td>
</tr>
<tr>
<td>Oriya</td>
<td>Tamil</td>
<td>Telugu</td>
</tr>
<tr>
<td>Kannada</td>
<td>Malayalam</td>
<td>Sinhala</td>
</tr>
<tr>
<td>Thai</td>
<td>Lao</td>
<td>Tibetan</td>
</tr>
<tr>
<td>Myanmar</td>
<td>Georgian</td>
<td>Hangul</td>
</tr>
<tr>
<td>Ethiopian</td>
<td>Cherokee</td>
<td>Canadian</td>
</tr>
<tr>
<td></td>
<td>Runic</td>
<td>Aboriginal</td>
</tr>
<tr>
<td>Ogham</td>
<td>Runic</td>
<td>Tagalog</td>
</tr>
</tbody>
</table>
## The scripts (2)

<table>
<thead>
<tr>
<th>Hanunoo</th>
<th>Buhid</th>
<th>Tagbanwa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khmer</td>
<td>Mongolian</td>
<td>Hiragana</td>
</tr>
<tr>
<td>Katakana</td>
<td>Bopomofo</td>
<td>Kanbun</td>
</tr>
<tr>
<td>CJK Ideographs</td>
<td>Yi</td>
<td>Old Italic</td>
</tr>
<tr>
<td>Gothic</td>
<td>Deseret</td>
<td>Musical Symbols</td>
</tr>
<tr>
<td>Arrows</td>
<td>Math Operators</td>
<td>Math Symbols</td>
</tr>
<tr>
<td>Misc Technical</td>
<td>Control Pictures</td>
<td>OCR</td>
</tr>
<tr>
<td>Enclosed Alpha.</td>
<td>Box Drawing</td>
<td>Block Elements</td>
</tr>
<tr>
<td>Geometric Shapes</td>
<td>Misc Symbols</td>
<td>Dingbats</td>
</tr>
<tr>
<td>Braille Patterns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### The scripts (3)

<table>
<thead>
<tr>
<th>Script</th>
<th>Script</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limbu</td>
<td>Tai Le</td>
<td>UPA</td>
</tr>
<tr>
<td>Linear B</td>
<td>Aegean numbers</td>
<td>Ugaritic</td>
</tr>
<tr>
<td>Shavian</td>
<td>Osmanya</td>
<td>Cypriot syllabary</td>
</tr>
<tr>
<td>Hexagrams</td>
<td>Tetragrams</td>
<td>New Tai Lue</td>
</tr>
<tr>
<td>Buginese</td>
<td>Glagolitic</td>
<td>Coptic</td>
</tr>
<tr>
<td>Tifinagh</td>
<td>Syloti Nagri</td>
<td>Old Persian</td>
</tr>
<tr>
<td>Kharoshthi</td>
<td>Balinese</td>
<td>N’Ko</td>
</tr>
<tr>
<td>Phags-pa</td>
<td>Phoenician</td>
<td>Sumero-Akkadian Cuneiform</td>
</tr>
</tbody>
</table>
**Greek: U+0370–U+03FF**

The Greek script is used for writing the Greek language and (in an extended variant) the Coptic language. The Greek script had a strong influence in the development of the Latin and Cyrillic scripts.

The Greek script is written in linear sequence from left to right with the occasional use of nonspacing marks. Greek letters come in upper- and lowercase pairs.

**Standards.** The Unicode encoding of Greek is based on ISO 8859-7, which is equivalent to the Greek national standard ELOT 928. The Unicode Standard encodes Greek characters in the same relative positions as in ISO 8859-7. A number of variant and archaic characters are taken from the bibliographic standard ISO 5428.

**Polytonic Greek.** Polytonic Greek, used for ancient Greek (classical and Byzantine), may be encoded using either combining character sequences or precomposed base plus diacritic combinations. For the latter, see the following subsection, “Greek Extended: U+1F00–U+1FFF.”

**Nonspacing Marks.** Several nonspacing marks commonly used with the Greek script are found in the Combining Diacritical Marks range (see Table 7-1).

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Alternative Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+0300</td>
<td>COMBINING GRAVE ACCENT</td>
<td>varia</td>
</tr>
<tr>
<td>U+0301</td>
<td>COMBINING acute accent</td>
<td></td>
</tr>
</tbody>
</table>
Code charts

Greek and Coptic

<table>
<thead>
<tr>
<th></th>
<th>037</th>
<th>038</th>
<th>039</th>
<th>03A</th>
<th>03B</th>
<th>03C</th>
<th>03D</th>
<th>03E</th>
<th>03F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Adobe Systems - To the BMP and beyond!

July 20, 2006 - Slide 40
Based on ISO 8859-7

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0374</td>
<td>GREEK NUMERAL SIGN ≡ 0397 H.dexia keraia</td>
<td>0389</td>
<td>GREEK</td>
</tr>
<tr>
<td></td>
<td>= dexia keraia</td>
<td></td>
<td>≡ 0399 I.</td>
</tr>
<tr>
<td></td>
<td>• indicates numeric use of letters</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ 02CA modifier letter acute accent</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>≡ 02B9 modifier letter prime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0375</td>
<td>GREEK LOWER NUMERAL SIGN ≡ 039F O.aristeri keraia</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= aristeri keraia</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• indicates numeric use of letters</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ 02CF modifier letter low acute accent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0376</td>
<td>&lt;reserved&gt;</td>
<td>038D</td>
<td>&lt;reserved&gt;</td>
</tr>
<tr>
<td>0377</td>
<td>&lt;reserved&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0378</td>
<td>&lt;reserved&gt;</td>
<td>038E</td>
<td>GREEK TONOS</td>
</tr>
<tr>
<td>0379</td>
<td>&lt;reserved&gt;</td>
<td></td>
<td>≡ 03A5 Y.TONOS</td>
</tr>
<tr>
<td>037A</td>
<td>GREEK YPOGEGRAMMEN</td>
<td>038F</td>
<td>GREEK TONOS</td>
</tr>
<tr>
<td></td>
<td>≡ 03A9 Ω.TONOS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Adobe Systems - To the BMP and beyond! July 20, 2006 - Slide 41
Part V

Development of the standard
Unicode Inc.

- the Unicode standard grew from work at Xerox and Apple
- the Unicode Consortium was incorporated in 1991
- six levels of membership
- ~120 members: companies, governments, individuals and organizations; ~20 voting members
Technical committees

- UTC: defines The Unicode Standard and associated standards and technical reports
- CLDR-TC: manages the Common Locale Data Repository and associated standards and technical reports
Standards

- The Unicode Standard
- A Standard Compression Scheme for Unicode
- Unicode Collation Algorithm
- Unicode Regular Expression Guidelines
- Character Mapping Markup Language
- Local Data Markup Language (LDML)
- Ideographic Variation Database
CLDR and LDML

• CLDR: Common Locale Data Repository collects and organizes locale data for the world highly cooperative effort formatting (and parsing) of numbers, dates, times, currency values, ... display names for language, script, region, currency, time-zones, ... collation order (used in sorting, searching, and matching text) identifying usage of measurement systems, weekend conventions, currencies, ...

• LDML: Locale Data Markup Language the XML markup in which the CLDR is represented
The Unicode Standard

- three levels of versions:
  - major (4.0): a new book is published
  - minor (4.1): no new book, but new characters
  - dot (4.0.1): no new characters
- stability guarantees
  - to ensure that data is perennial
- standard comprises:
  - a book
  - annexes (will be part of the 5.0 book, separate before)
  - the UCD
  - a release description, for non-major releases
ISO/IEC 10646

• defined by JTC1/SC2/WG2
• aligned with Unicode, via cooperation
  same repertoire
  same character names
  same code points
• does not define properties or processing
• current: ISO/IEC 10646:2003 + Amendments 1 and 2 + 5 characters
  corresponds to Unicode 5.0
Part VI

Processing
Properties and algorithms

- each character has a number of properties in the Unicode Character Database (UCD)
- algorithms based on properties easy to upgrade to a new repertoire in the standard or in technical reports
The Bidirectional Algorithm

- text is stored in logical order: שבעה $10.
- bidi computes the display order: שבעה 10 $10
- characters have directionality:

<table>
<thead>
<tr>
<th>chars</th>
<th>directionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C</td>
<td>L - Left to Right (strong)</td>
</tr>
<tr>
<td>י, ג, ד, ה</td>
<td>R - Right To Left (strong)</td>
</tr>
<tr>
<td>1, 0</td>
<td>EN - European Number (weak)</td>
</tr>
<tr>
<td>$</td>
<td>ET - European Number Terminator (weak)</td>
</tr>
<tr>
<td>.</td>
<td>CS - Common Number Separator (weak)</td>
</tr>
</tbody>
</table>
The Bidirectional Algorithm (2)

- bidi resolves the directionality of weak characters
  stored: $10.
  resolved: $10
  displayed: שנבט 10.

- context matters; e.g. adding a 5
  stored: $10.5
  resolved: $10.5
  displayed: שנבט 5.

- format characters to handle special cases
  stored: <RLO>abc<PDF>def
  resolved: abc def
  displayed: def cba
The Bidirectional Algorithm (3)

- shape of character can depend on directionality
  U+0028 LEFT PARENTHESIS
  function is opening parenthesis
  displays as ( in ltr
  displays as ) in rtl
- captured in the mirrored and mirror glyph properties
  can be overridden by higher level protocols
Unicode Normalization Forms

- multiple representations of the “same” text
  é vs. e ô
- a normalization form selects one of those representations
  e.g. allows binary comparisons
- two basic forms
  NFC: prefers *composed* characters
  NFD: prefers *decomposed* characters
- guarantee of stability
  e.g. for databases
Canonical Decomposition Property

- canonical decomposition is a property
- maps one character to one or more characters
- includes:
  - combining sequences: \( \acute{\varepsilon} \equiv \varepsilon \, \hat{o} \)
  - Hangul syllables: \( \text{踽} \equiv \text{ㅜ} \, \text{ㅏ} \, \text{ㅣ} \, \text{教え} \)
  - singletons: \( \Omega \equiv \Omega \) (ohm sign and omega)
NFD

• apply repeatedly the canonical decompositions
• reorder combining marks by increasing combining class

\[
\begin{array}{c}
\text{decompose} \\
\begin{array}{c}
\text{reorder}
\end{array}
\end{array}
\]

© Adobe Systems - To the BMP and beyond! July 20, 2006 - Slide 56
NFC

- transform to NFD
- recompose combining sequences

\[
\begin{align*}
\text{decompose} & \quad \text{NFD} \\
\text{reorder} & \quad \text{combine} \\
\text{combine} & \quad \text{combine}
\end{align*}
\]
NFKD, NFKC

- also use compatibility mappings when decomposing
- compatibility mappings are a mixed bag
- therefore, NFKD and NFKC are difficult to use
Case mappings

• simple mappings:
  one to one
  context independent
  avoid: insufficient for e.g. ligatures, German

• complex mappings:
  one to many: \( \text{β} \rightarrow \text{SS} \), or \( \text{fi} \rightarrow \text{Fl} \)
  contextual: \( \Sigma \rightarrow \varsigma \) in final position, not \( \sigma \)
  local-sensitive: \( i \rightarrow \iota \) in Turkish, not \( I \)

• case folding for caseless matches
Unicode Collation Algorithm

- many different sorting orders
  - English: péché < pêche; French: pêche < péché
  - Spanish (trad): ch single letter, between c and d
  - German (trad): ö equivalent to oe
- dictionary, phonebook, etc.
- sorting algorithm that:
  - is efficient
  - accounts for canonical and compatibility equivalences
  - can be tailored to implement most orders
  - by default, provides a reasonable sorting
Part VII

Unicode and other standards
Transcoding

Unicode

chars

UTF-8
UTF-16
UTF-32

code enc.
Transcoding

Unicode

chars

UTF-8
UTF-16
UTF-32

BAR

enc.

foo
Transcoding

chars

UTF-8
UTF-16
UTF-32

enc.

foo

Unicode

BAR

© Adobe Systems - To the BMP and beyond! July 20, 2006 - Slide 65
Transcoding

Unicde

chars

UTF-8
UTF-16
UTF-32
foo
Transcoding

Unicode

chars

enc.

UTF-8
UTF-16
UTF-32
foo

ASCII
UTF-8
ASCII
JIS
UTF-16
JIS
ISCII
UTF-32
ISCII
...
Character Hiragana MA (revisited)

- abstract character: the letter ま of the Hiragana script
- coded character:
  name: HIRAGANA LETTER MA
  code point: U+307E
- encoding forms:
  UTF-8: E3 81 BE
  UTF-16: 307E
  UTF-32: 0000307E
  ASCII: n/a
  JIS 0208: 245E
  KSX 1001: 2A5E
XML

- XML does just that!
- all characters are Unicode characters
- any encoding form (including non-UTFs) is acceptable
Anatomy of an implementation

UTF-16

transcoding layer

UTF-8

JIS
Anatomy of an implementation (2)
JIS X 0208:1997 and JIS X 0213:2000

- standard:
  - Japan
  - two complementary standards
  - market will probably demand both

- characters:
  - JIX X 0208: 7k
  - JIS X 0213: 4k
  - ~300 in plane 2

- mappings:
  - mapping to Unicode 3.2 does not use PUA

- encoding:
  - 2 bytes
GB 18030-2000

- standard:
  People’s Republic of China
  mandatory for products sold there
- characters:
  28k
- mappings:
  mapping to Unicode 3.2, BMP only, uses PUA for 25 characters
  mapping to Unicode 4.1: no PUA
- encoding:
  1, 2, or 4 bytes
HKSCS

- standard:
  Hong Kong SAR
  supplements BIG5
  mandatory for selling to the government
- characters:
  4,818 characters
  1,651 in plane 2
- mappings:
  mapping to Unicode 3.0 uses PUA for 1,686 characters
  mapping to Unicode 3.1-4.0 uses PUA for 35 characters
  mapping to Unicode 4.1 does not use the PUA
- encoding:
  2 bytes
The bad news

- there is not always an “official” mapping
different vendors do different things
- PUA conflicts:
  HKSCS 9571 (U+2721B) ↔ U+E78D
  GB18030 A6D9 (,) ↔ U+E78D
- PUA differentiation:
  HKSCS 8BFA (U+20087) ↔ U+F572
  GB18030 FE51 (U+20087) ↔ U+E816
- no need for PUA with Unicode 4.1
Part VIII

Resources
Unicode Inc. Resources

- Unicode Consortium: [http://www.unicode.org](http://www.unicode.org)
  - UAXes
  - technical reports
  - UCD
  - unibook: application to explore the UCD
  - online Unihan database
- The Unicode Standard, Version 4.0
  - ISBN 0-321-18578-1
  - also at [www.unicode.org](http://www.unicode.org)
- Unicode Guide
  - 6 pages laminated quick study guide
  - ISBN 9781423201809
Internationalization and Unicode Conference

- Internationalization and Unicode Conference once or twice a year
  IUC 30, Washington DC, November 2006
  (http://www.unicode.org/conference)
Other Resources

- IBM’s site for Unicode: http://www.ibm.com/developer/unicode
- Mark Davis’ site: http://www.macchiato.com
- Michael Everson’s site: http://www.evertype.com