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:
  Adobe®
  he said “Hi!”
  résumé†
  cañon

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 fundation* 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

USV	Unit 1	Unit 2	Unit 3	Unit 4
0000-007F	*0*xxxxxxx			
0080-07FF	*110*xxxxx	*10*xxxxxx		
0800-D7FF	*1110*xxxx	*10*xxxxxx	*10*xxxxxx	
E000-FFFF				
10000-10FFFF	*11110*xxx	*10*xxxxxx	*10*xxxxxx	*10*xxxxxx

- e.g. $F03F_{16} = 1111\ 0000\ 0011\ 1111_2$

  → *1110*1111 *10*000000 *10*111111$_2$ = EF 80 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

USV	Unit 1	Unit 2
0000-D7FF	xxxxxxxxxxxxxxxx	
E000-FFFF		
10000-10FFFF (10000 bias)	*110110*xxxxxxxxxx	*110111*xxxxxxxxxx

- takes advantage of gap in scalar values
  - 110110xxxxxxxxxx = D800 - DBFF
  - 110111xxxxxxxxxx = 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

USV	Unit 1
0000-D7FF	*00000000000*xxxxxxxxxxxxxxxxxxxx
E000-10FFFF	

- 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  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>
<tr><th>Basic Type</th><th>Character status</th><th>Code point status</th></tr>
<tr><td>Graphic</td><td rowspan="4">Assigned to abstract character</td><td rowspan="5">Designated (assigned) code point</td></tr>
<tr><td>Format</td></tr>
<tr><td>Control</td></tr>
<tr><td>PUA</td></tr>
<tr><td>Surrogate</td><td rowspan="3">Not assigned to abstract character</td></tr>
<tr><td>Noncharacter</td></tr>
<tr><td>Reserved</td><td>Undesignated (unassigned) code point</td></tr>
</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: ه ه ه ه
- going from characters to glyphs: *shaping*

प / ◌ू / र / ◌् / त / ि

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 A and French A
  - U+0041 LATIN CAPITAL LETTER A
- single , regardless of its usage
- no confusion between Latin A, Greek A and Cyrillic A
  - U+0041 A LATIN CAPITAL LETTER A
  - U+0391 A GREEK CAPITAL LETTER ALPHA
  - 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

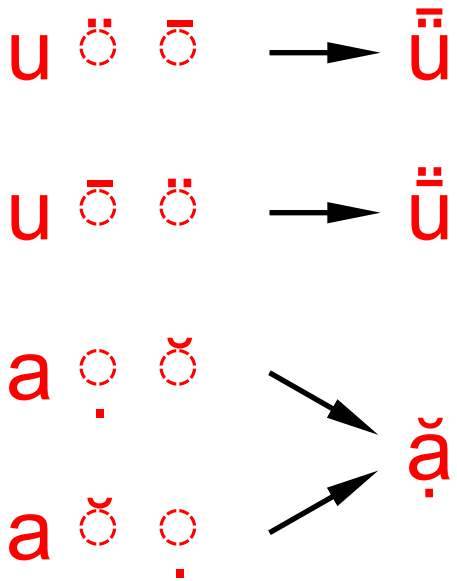
- the intended use of a character is unambiguous
- the behavior of a character is unambigous
  - properties
  - algorithms

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:
  - é : U+0065 e U+0301 ◌́
  - ∉ : U+2208 ∈ U+0338 ◌̸
  - ड़ : U+0921 ड U+093C ◌़

Dynamic composition (2)

- can have multiple marks
- from base character outwards



Equivalence for precomposed forms

- some precomposed characters are encoded
  U+00E9 é
- canonical equivalence
  U+00E9 é ≡ U+0065 e U+0301 ◌́
- similarly for Hangul syllables
  U+D4DB 뷓 ≡ U+1111 ᄑ U+1171 ᅱ U+11B6 ᆱ
- *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

storage	display
ASDF	ASDF
גבנש	שנבג

- 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 豈
- canonical decomposition
  U+00E9 é ≡ U+0065 e U+0301 ◌́
  U+F900 豈 ≡ U+8C48 豈
- 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

Latin	IPA	Greek and Coptic
Cyrillic	Armenian	Hebrew
Arabic	Syriac	Thaana
Bengali	Gurmukhi	Gujarati
Oriya	Tamil	Telugu
Kannada	Malayalam	Sinhala
Thai	Lao	Tibetan
Myanmar	Georgian	Hangul
Ethiopic	Cherokee	Canadian Aboriginal
Ogham	Runic	Tagalog

The scripts (2)

Hanunoo	Buhid	Tagbanwa
Khmer	Mongolian	Hiragana
Katakana	Bopomofo	Kanbun
CJK Ideographs	Yi	Old Italic
Gothic	Deseret	Musical Symbols
Arrows	Math Operators	Math Symbols
Misc Technical	Control Pictures	OCR
Enclosed Alpha.	Box Drawing	Block Elements
Geometric Shapes	Misc Symbols	Dingbats
Braille Patterns		

The scripts (3)

Limbu	Tai Le	UPA
Linear B	Aegean numbers	Ugaritic
Shavian	Osmanya	Cypriot syllabary
Hexagrams	Tetragrams	New Tai Lue
Buginese	Glagolitic	Coptic
Tifinagh	Syloti Nagri	Old Persian
Kharoshthi	Balinese	N'Ko
Phags-pa	Phoenician	Sumero-Akkadian Cuneiform

Block descriptions

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.

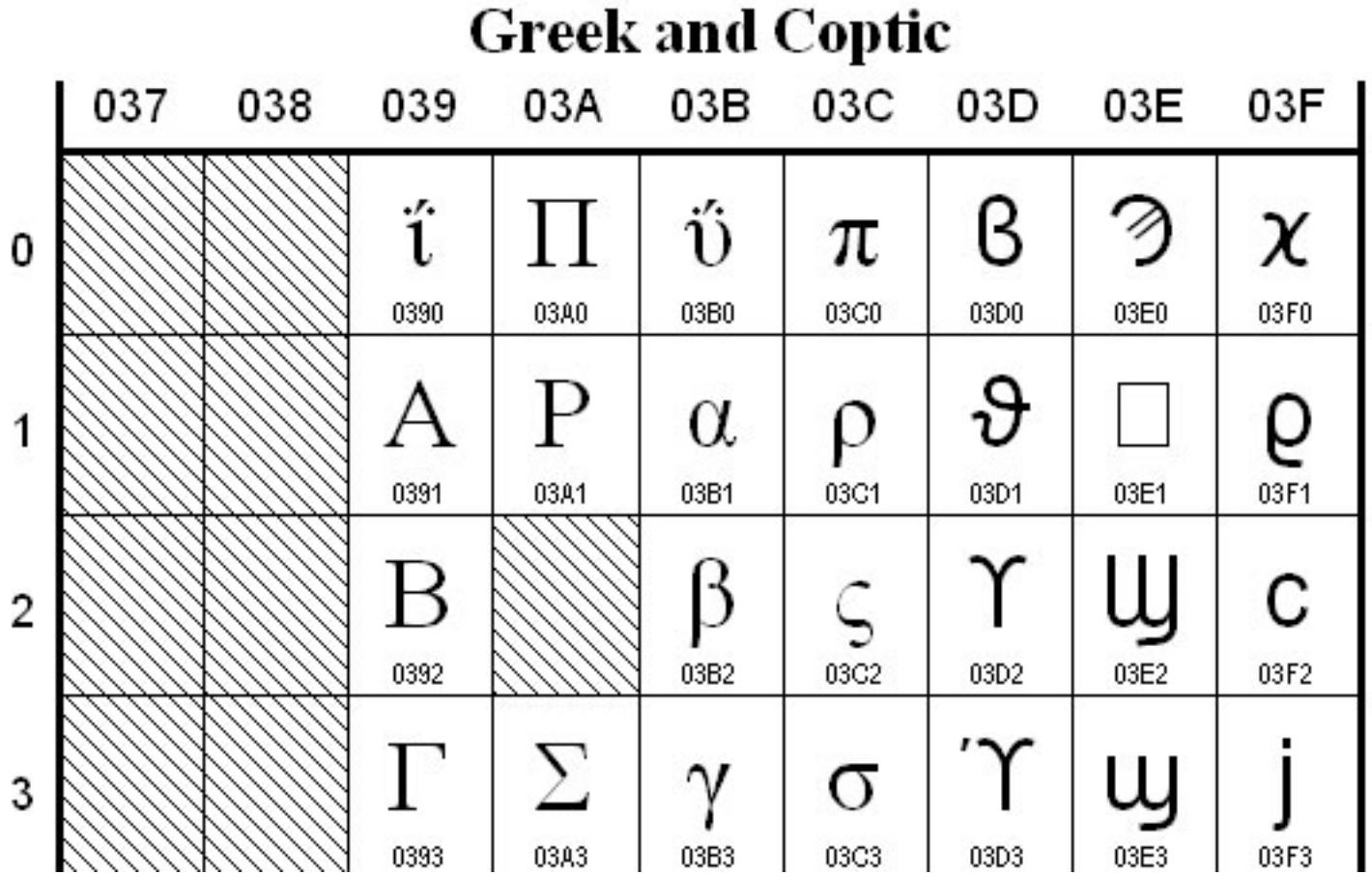
Polytonic Greek. Polytonic Greek, used for ancient Greek (classical and Byzantine), may be encoded using either combining character sequences or precomposed base plus diacritic combinations. For the latter, see the following subsection, "Greek Extended: U+1F00–U+1FFF."

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

Table 7-1. Nonspacing Marks Used with Greek

Code	Name	Alternative Names
U+0300	COMBINING GRAVE ACCENT	*varia*
U+0301	COMBINING ACUTE ACCENT	*tonos, oxia*

Code charts


Greek and Coptic
037
038
039
03A
03B
03C
03D
03E
03F
0
1
2
3
0390
03A0
03B0
03C0
03D0
03E0
03F0
0391
03A1
03B1
03C1
03D1
03E1
03F1
0392
03B2
03C2
03D2
03E2
03F2
0393
03A3
03B3
03C3
03D3
03E3
03F3

Code charts (2)

Based on ISO 8859-7

0374 ʹ GREEK NUMERAL SIGN
= dexia keraia
• indicates numeric use of letters
→ 02CA ˊ modifier letter acute accent
≡ 02B9 ʹ modifier letter prime

0375 ͵ GREEK LOWER NUMERAL SIGN
= aristeri keraia
• indicates numeric use of letters
→ 02CF ˏ modifier letter low acute accent

0376 <reserved>

0377 <reserved>

0378 <reserved>

0379 <reserved>

037A GREEK YPOGEGRAMMENI

0389 Ή GREEK
≡ 0397 H

038A Ί GREEK
≡ 0399 I

038B <reserve

038C Ό GREEK
TONOS
≡ 039F O

038D <reserve

038E Ύ GREEK
TONOS
≡ 03A5 Y

038F Ώ GREEK
TONOS
≡ 03A9 Ω

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: גבנש $10.
- bidi computes the display order: . $10 שנבג
- characters have directionality:

chars	directionality
A, B, C	L - Left to Right (strong)
ש, נ, ב, ג	R - Right To Left (strong)
1, 0	EN - European Number (weak)
$	ET - European Number Terminator (weak)
.	CS - Common Number Separator (weak)

The Bidirectional Algorithm (2)

- bidi resolves the directionality of weak characters

stored	גבנש $10.
resolved	גבנש $10 .
displayed	. $10 שנבג

- context matters; e.g. adding a 5

stored	גבנש $10.5
resolved	גבנש $10.5
displayed	$10.5 שנבג

- 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 overriden by higher level protocols

Unicode Normalization Forms

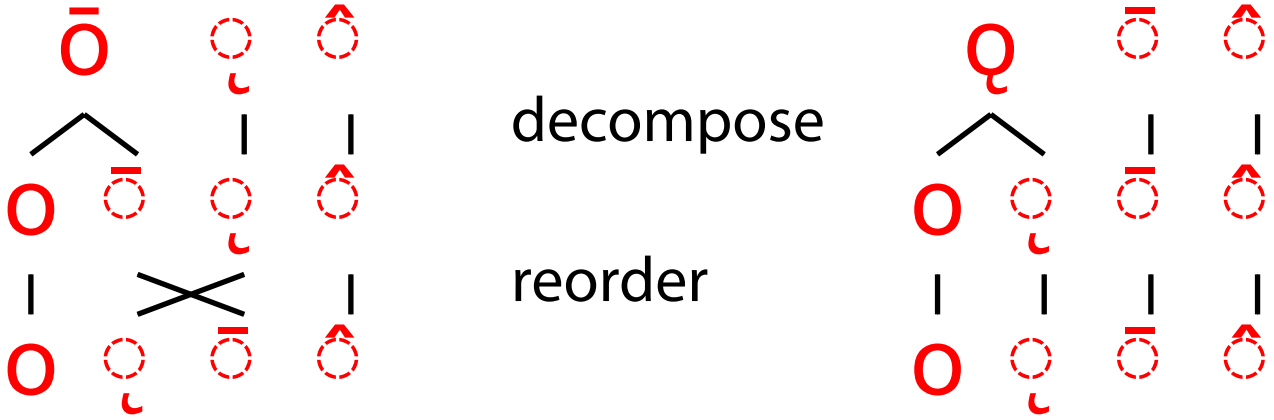
- multiple representations of the “same” text
  é vs. e ◌́
- a normalization form selects one of those representations
  e.g. allows binary comparisons
- two basic forms
  NFC: prefers *composed* characters
  NFD: prefers *decomposed* characters
- guarantee of stability
  e.g. for databases

Canonical Decomposition Property

- canonical decomposition is a property
- maps one character to one or more characters
- includes:
  - combining sequences: é ≡ e ◌́
  - Hangul syllables: 풿 ≡ ᄑ ᅮ ᆱ
  - singletons: Ω ≡ Ω (ohm sign and omega)

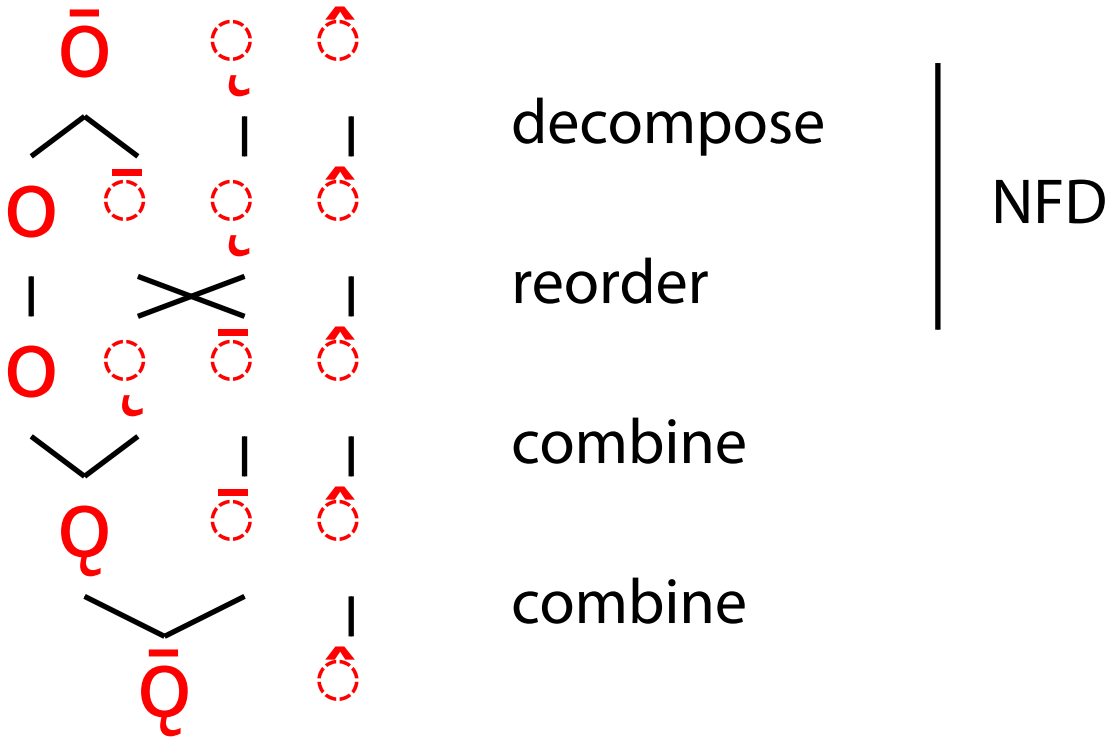
NFD

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



NFC

- transform to NFD
- recompose combining sequences



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: ß → SS, or fi → FI
  - contextual: Σ → ς in final position, not σ
  - local-sensitive: i → İ in Turkish, not I
- case folding for caseless matches

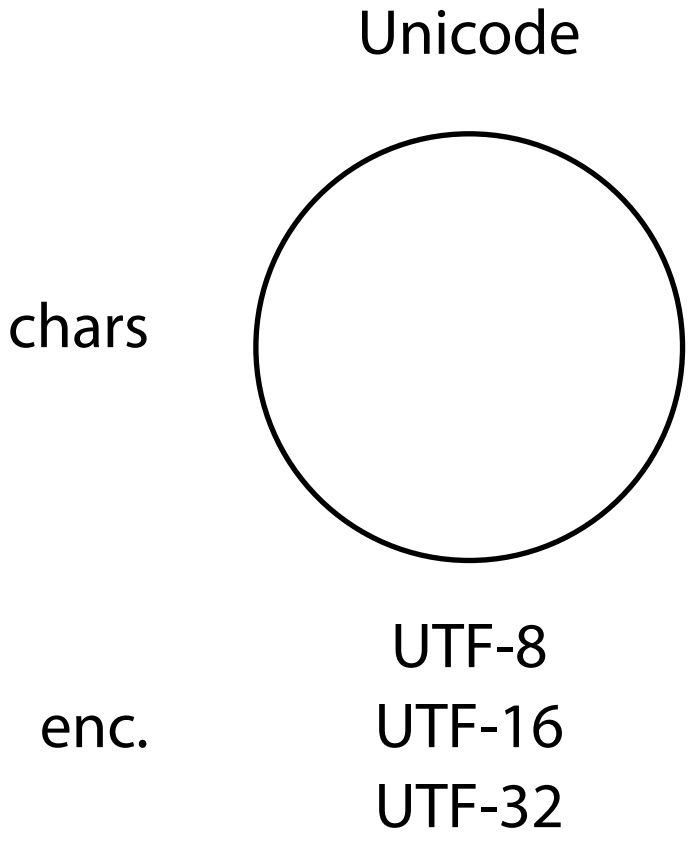
Unicode Collation Algorithm

- 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

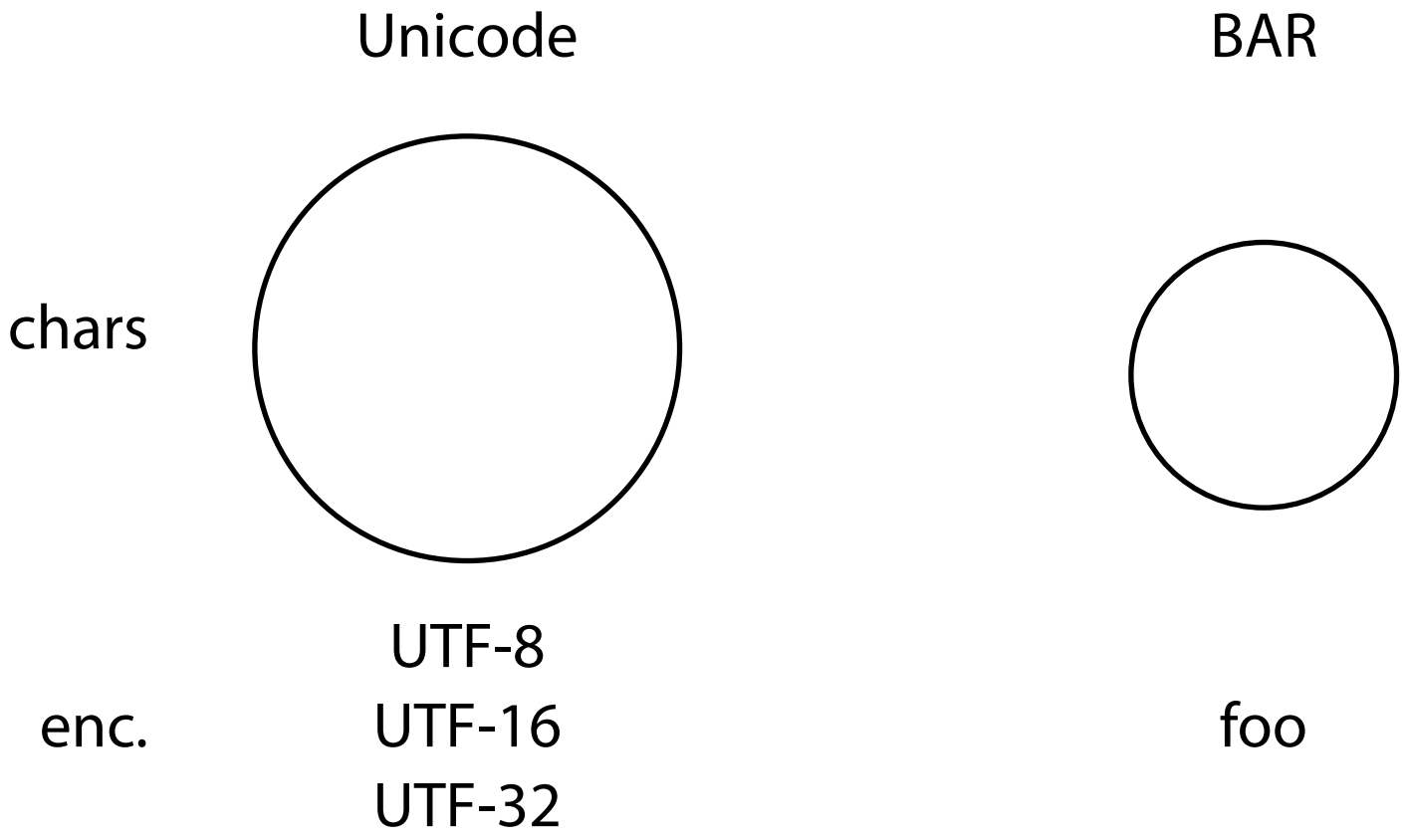
Part VII

Unicode and other standards

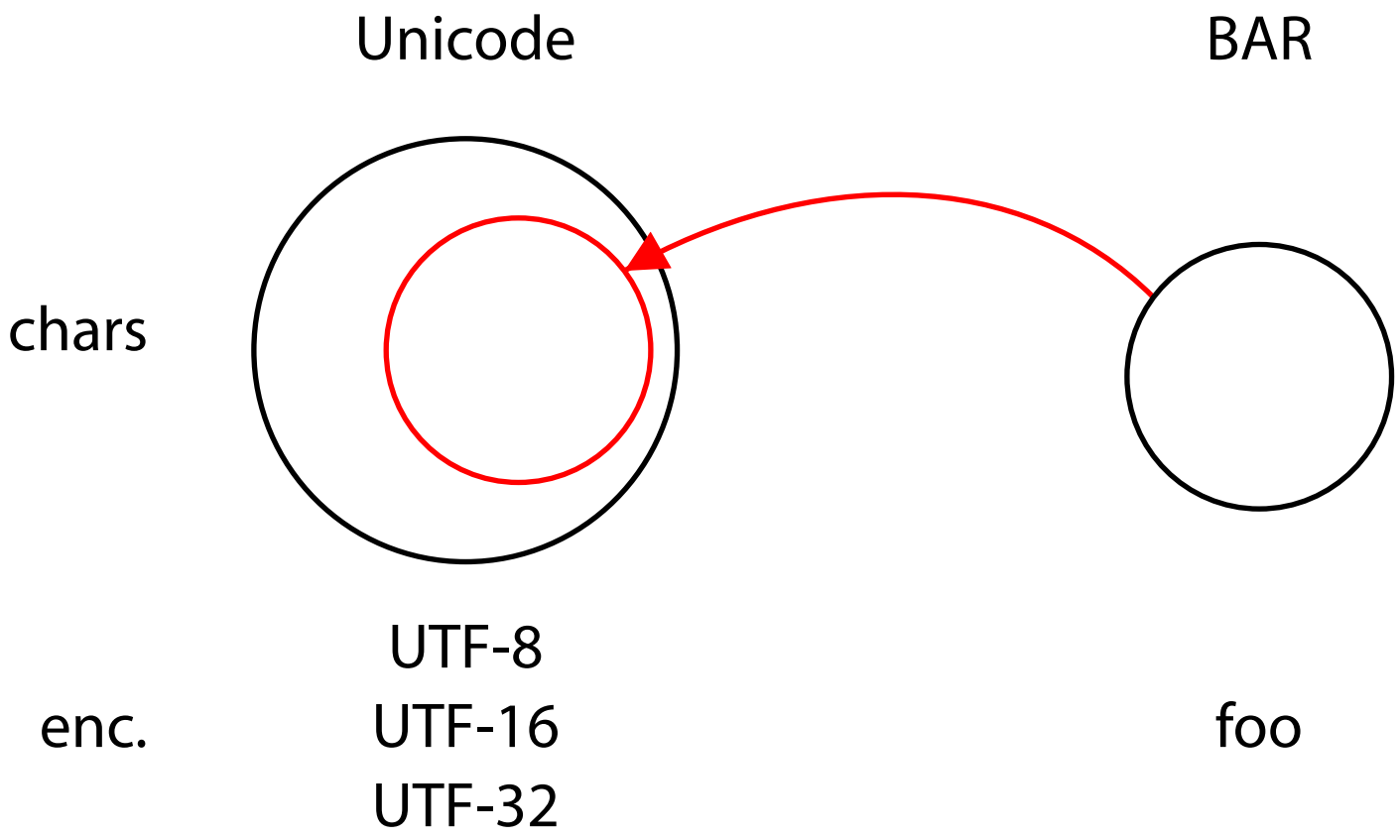
Transcoding



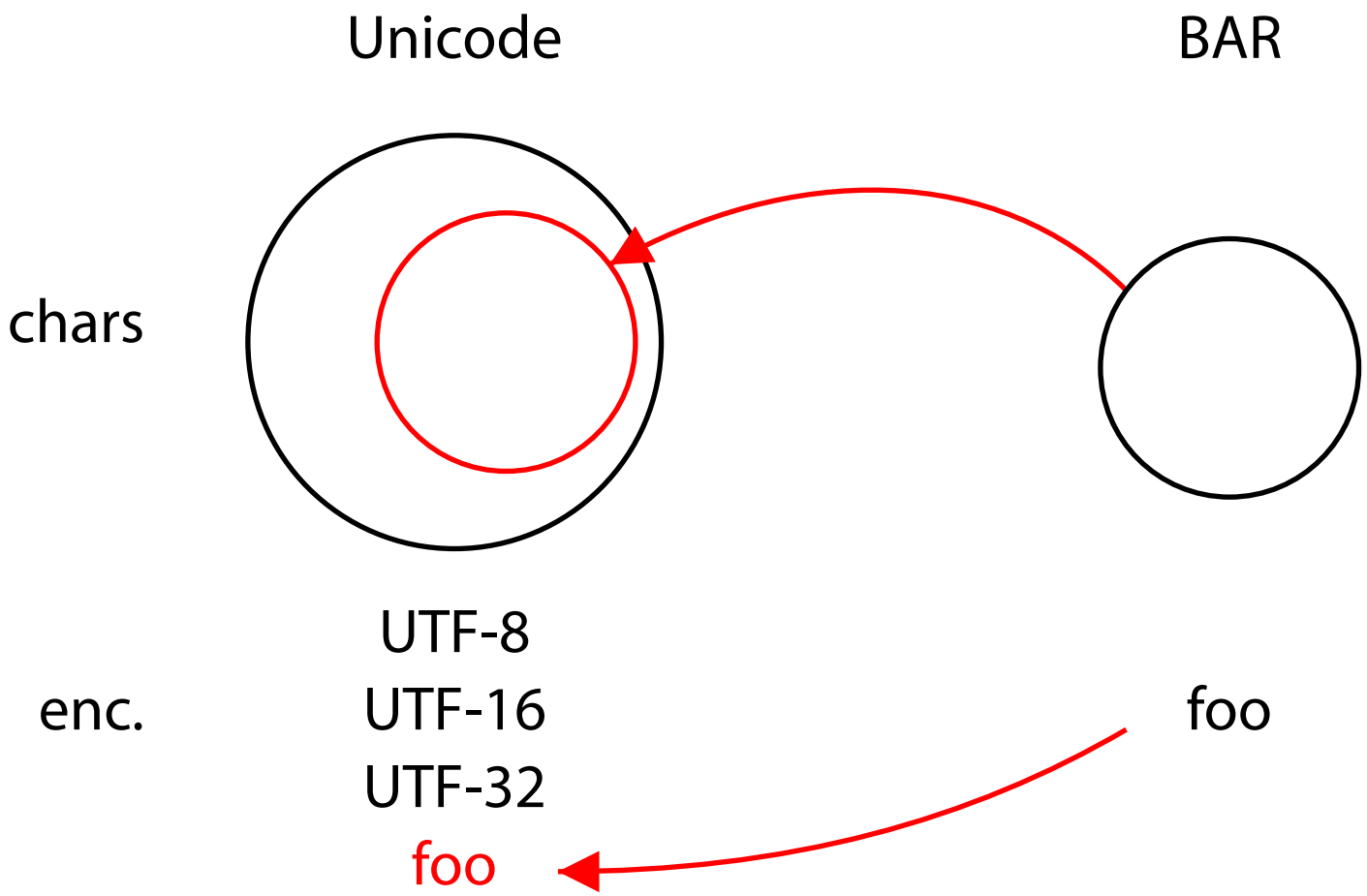
Transcoding



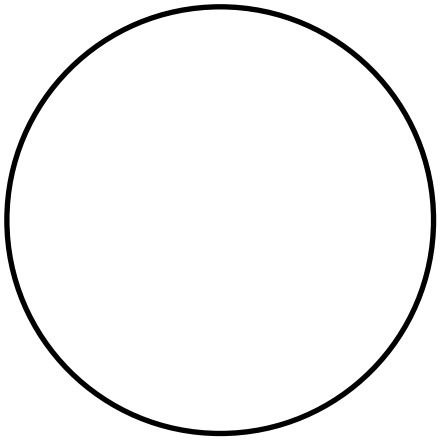
Transcoding



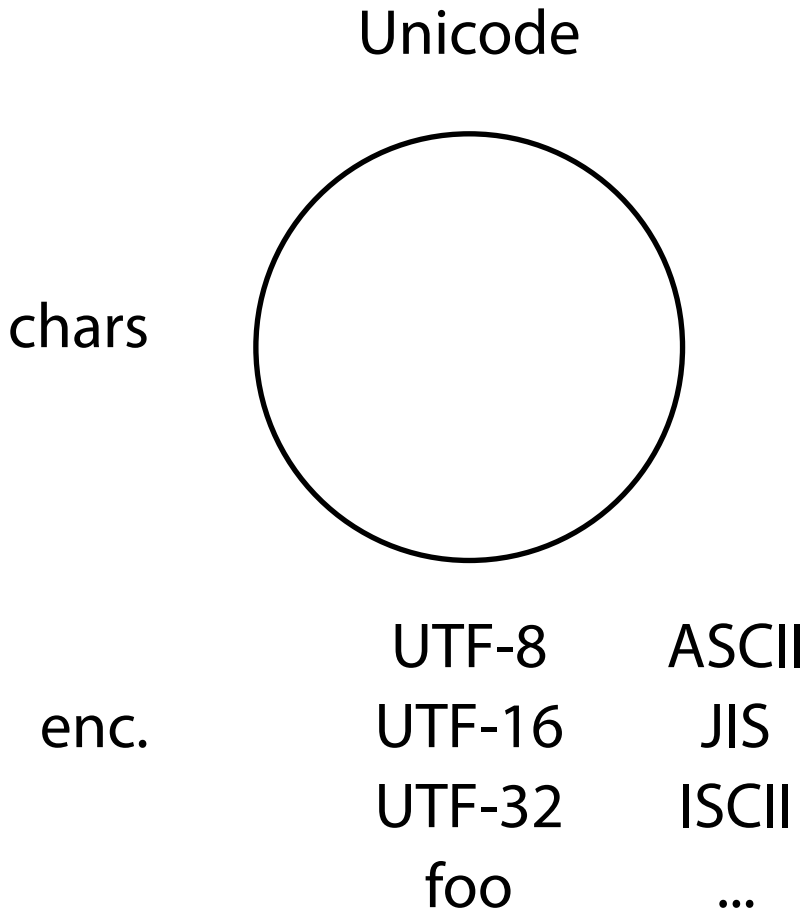
Transcoding



Transcoding



Transcoding



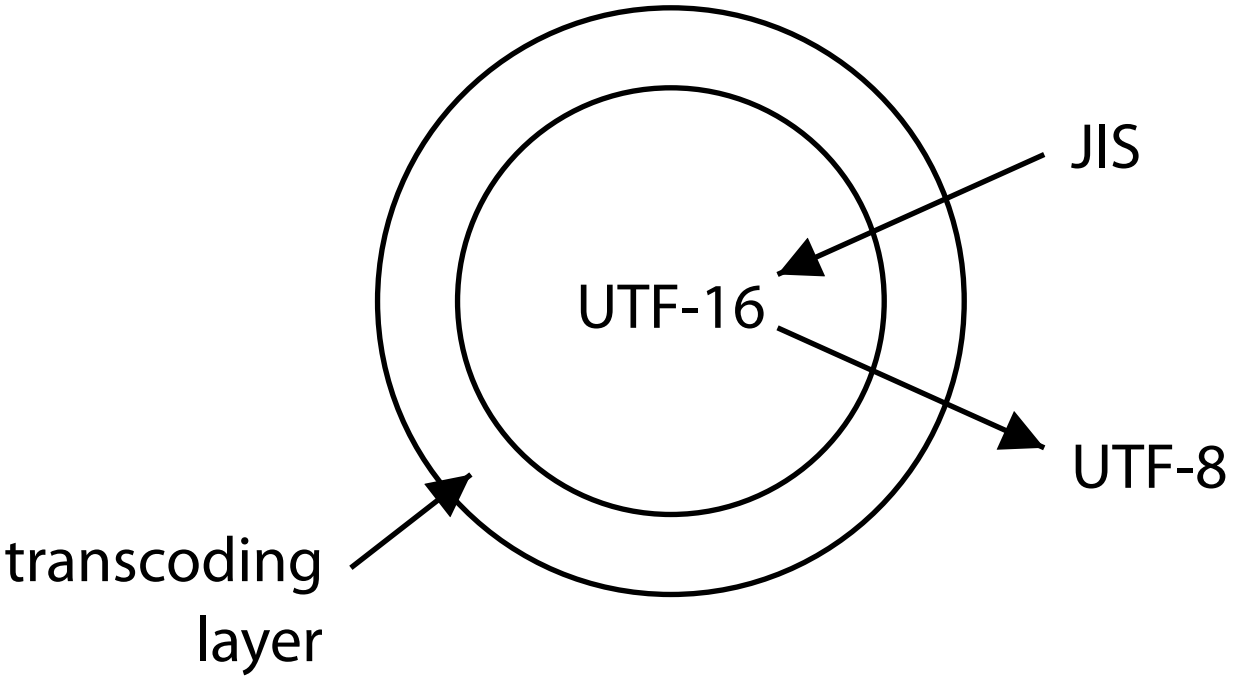
Character Hiragana MA (revisited)

- 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

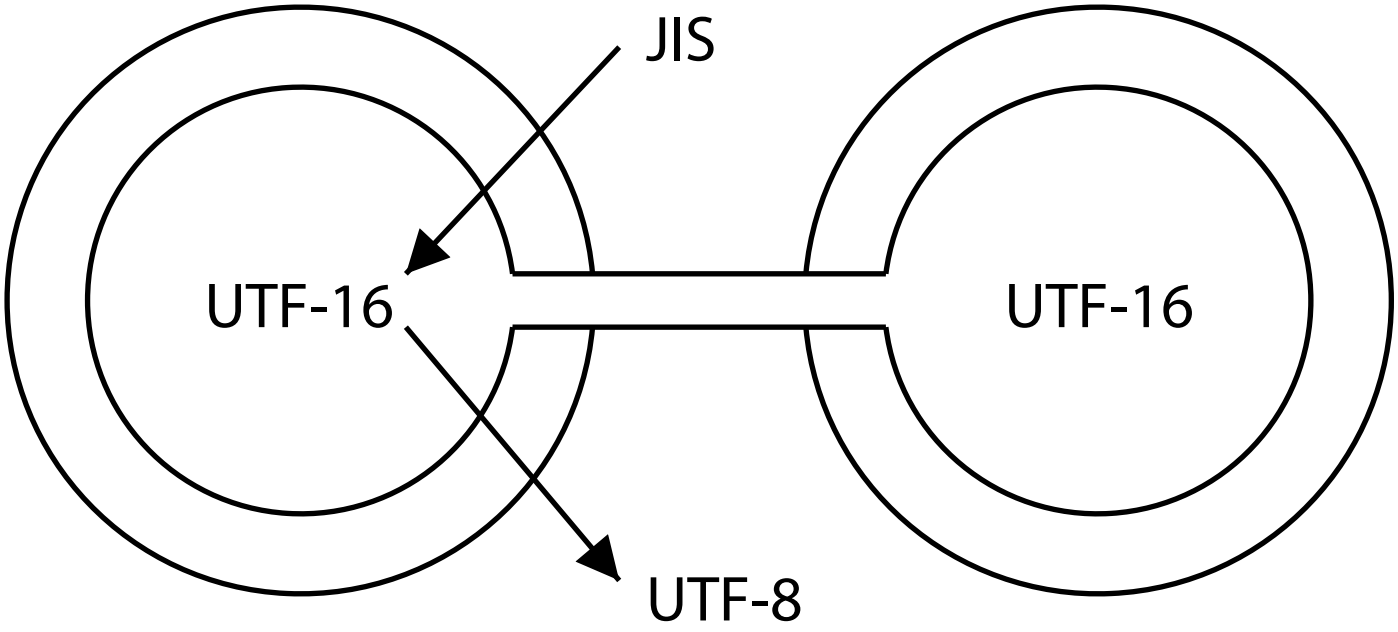
XML

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

Anatomy of an implementation



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
  - 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
- 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
- International Components for Unicode: http://oss.software.ibm.com/developerworks/opensource/icu/project/
- Mark Davis' site: http://www.macchiato.com
- Michael Everson's site: http://www.evertype.com
- *Unicode Demystified* by Richard Gillam (ISBN 0201700522), August 2002
- *Unicode Explained* by Jukka Korpela (ISBN 0-596-10121-X), June 2006