



Proposed Draft Unicode Technical Report #30

CHARACTER FOLDINGS

Version	1.0
Authors	Asmus Freytag (asmus@unicode.org)
Date	2002-05-08
This Version	http://www.unicode.org/unicode/reports/tr30-1.html
Previous Version	none
Latest Version	http://www.unicode.org/unicode/reports/tr30/
Tracking Number	<u>1</u>

Summary

This report identifies a set of character foldings, in other words, operations that map similar characters to a common target. Such operations are used to ignore certain distinctions between similar characters. The report also provides an algorithm for applying these operations to searching.

Status

*This document has been approved by the Unicode Technical Committee for public review as a **Proposed Draft Unicode Technical Report**. Making this document available for public review does not imply endorsement by the Unicode Consortium. This is a draft document which may be updated, replaced, or superseded by other documents at any time. This is not a stable document; it is inappropriate to cite this document as other than a work in progress.*

Please send comments to the authors. A list of current Unicode Technical Reports is found on <http://www.unicode.org/unicode/reports/>. For more information about versions of the Unicode Standard, see <http://www.unicode.org/unicode/standard/versions/>.

The References provide related information that is useful in understanding this document. Please mail corrigenda and other comments to the author(s).

The foldings specified in this Technical Report are [intended] to cover Unicode, Version 3.2

[Notes to reviewers are indicated like this]

Contents

1. [Scope](#)
 2. [Introduction](#)
 - 2.1 [Relation to Normalization](#)
 3. [Definitions](#)
 - 3.1 [Notation](#)
 4. [Conformance](#)
 5. [Specifications](#)
 - 5.1 [Folding Algorithm](#)
 - 5.2 [Specification of Foldings](#)
 6. [Notes and Guidelines](#)
 - 6.1 [General Notes](#)
 - 6.2 [Problematic Foldings](#)
- [References](#)
- [Modifications](#)
-

1.0 Scope

This report identifies a set of character foldings, in other words, operations that map similar characters to a common target. Such operations are used to ignore certain distinctions between similar characters and are useful for "fuzzy" or "loose" searches, among other things. Each of the folding operations specified in this report has well-understood properties, and is appropriate in specific contexts. For example some identifiers need case folding, some do not. Some text searches need to preserve superscript form such as the *trademark symbol*™, while others do not. Not all of these folding operations may be appropriate in a give context. See [section 6.2](#) for some of the more problematic folding or expansion operations.

The report also a provides an algorithm for combining these operations for the purpose of searching or programmatic identifier matching. They combine canonical normalization with optional folding operations. This allows implementers to decide which folding option is useful for a particular purpose.

String foldings for programmatic identifier matching are related to search term foldings, but with the important difference that the set of allowable characters is restricted. Where required, identifier matches are addressed specifically below.

2.0 Introduction

For the purpose of fuzzy text matching, including both programmatic identifier matching and

general text searching, it is often necessary to selectively ignore otherwise meaningful distinctions between related characters, for example upper and lower case, removal of accent marks, etc. This process can be called *search term folding*. Depending on the operation, different foldings need to be applied, and possible interactions must be carefully managed. False positive matches may be acceptable more often than false negative matches.

2.1 Relation to Normalization

Search term folding is somewhat similar to normalization in that it also reduces several alternative representations into one, but there are important differences. [Normalization] is often intended for permanent transformation of data, but search term folding is by nature transient. As defined, the normalization forms offer two broad levels of distinctions (they either preserve or don't preserve compatibility distinctions).

The choice of which distinctions may be ignored for search term folding needs to be more specific; it depends on the nature of the operation. One size does not fit all.

For example, two of the normalization forms depend on compatibility mappings which replace character by their compatibility decompositions. Applying certain of these compatibility mappings may lead to unintended false positive matches, preventing their use in general text searches. In combination with whole word search it could even lead to unintended false negatives. (See [section 6.2 Problematical foldings](#)).

Furthermore, normalization and case folding are defined as separate and independent operations, but case folding often occurs together with other foldings in search term folding. In order to avoid inconsistencies, search term folding needs to address the interaction of case folding with the other steps in the algorithm.

Search term folding includes canonical normalization; however, whether the normalized text is in the composed (NFC) or decomposed form (NFD) is of secondary importance in terms of defining the foldings. Due to the transient nature of search term folding, the distinction between NFC and NFD is immaterial, as long as the two forms are not mixed.

3.0 Definitions

All terms not defined here shall be as defined in the Unicode Standard or in the online [\[Glossary\]](#).

Folding operation – a folding operation removes a distinction between related characters by mapping them to the same target.

For example, case folding removes the case distinction, by replacing upper and title case variants of a character with the lower case.

Multigraph expansion – a multigraph expansion replaces a digraph or higher multigraph, such as

double prime or *triple prime*, by its expansion into an equivalent series of single characters, in this case, two or three single *prime* characters.

Other foldings may replace individual characters by a character sequence.

[Ed. This definition may not capture the most useful sub-category of foldings – even if it were, it would need to be reworded. Left in here for now since it's referred to in the big table.]

3.1 Notation

The following notational conventions are used in this TR

xxxx..yyyy	indicates an inclusive range
xxxx, yyyy	indicates an alternative
<xxxx>	refers to a compatibility mapping tag as defined in [Aliases].
Xy	refers to a particular value for the General Category property defined in [UnicodeData], e.g. "Pd"
<+>	means a folding is contained in the Unicode data files, but its general use is not recommended
[CD]	Compatibility Decomposition

4.0 Conformance

The material in this report is entirely of an *informative* nature. Implementations that are conformant to the Unicode Standard are not required to support these specifications.

All specifications of algorithms are stated as logical algorithms—any implementations that achieve the same result are fully equivalent. In particular, implementations may use optimization techniques, such as normalizing and folding 'on demand'.

5.0 Specifications

5.1 Folding algorithm

5.1.1 Basic Folding Algorithm

The basic algorithm for search term folding can be stated as

- a. Apply optional folding operations
- b. Apply canonical decomposition
- c. Repeat a & b until stable
- d. Apply composition if necessary

where each step is applied on the whole string, and applies to the result of the preceding

operation. Implementations may be able to avoid step (c) by applying additional transformations.

5.1.2 Identifier Folding algorithm

For identifier folding, it is important to account for prohibited characters. This adds a new step (e). The modified basic algorithm then becomes:

- a. Apply optional folding operations
- b. Apply canonical decomposition
- c. Repeat a & b until stable
- d. Apply composition if necessary
- e. Eliminate or flag forbidden characters

Foldings in step (a) can be modified to also disallow certain characters by mapping them to forbidden characters, which are then caught in step (e). Implementations may be able to avoid step (c) by applying additional transformations.

5.2 Specification of folding operations

The following table summarizes the definition of the folding operations. Foldings that are *multigraph expansions* have been collected at the end of the table.

- The *description* column identifies the folding.
- The *source* column identifies the set of characters subject to the folding operation by referencing a set of code points, a set of general categories, or a compatibility mapping tag. All characters matching the source condition are subject to the given folding. Note that this column does *not* indicate the set of characters with which the source characters are equivalenced by the folding.
- The *target* column indicates the result of the folding, either by reference to an operation, or, in some cases, by providing the single Unicode character to which a whole set of source characters is folded.
- The *data file* column indicates which data file carries the character by character information to implement the operation referred to in the *target* column.

Descriptive Name	Source Characters	Target Characters	Data file specifying the mapping
Accent removal	Latin/Greek/Cyrillic characters with canonical decomposition	base characters of [CD]	[UnicodeData]
Diacritic removal (includes stroke, hook, descender)	Latin/Greek/Cyrillic characters with diacritics	related base character	[DiacriticFolding] TBD
Case folding	any	case fold according to CaseFolding.txt	[CaseFolding]
Canonical duplicates folding (e.g. Ohm - Omega)	0374, 037E, 0387, 1FBE, 1FEF, 1FFD, 2000, 2001, 2126, 212A, 212B, 2329..232A	[CD]	[UnicodeData]
Dashes folding	Pd	U+002D	[UnicodeData]

Greek letterforms folding	03D0..03D2, 03D5..03D6, 03F0..03F2, 03F4..03F5	[CD]	[UnicodeData]
Han Radical folding	2F00..2F5D, 2EF3, 2E9F	TBD	[RadicalFolding] TBD
Suzhou Numeral folding	3038..303A, 3021..3029	TBD	[SuzhouFolding]TBD
Hebrew Alternates folding	FB20..FB28	[CD]	[UnicodeData]
Jamo folding	3131..3183	[CD]	[UnicodeData]
Kana folding	Hiragana	Katakana	[KanaFolding] TBD
Ligature expansion Misc.	0587, 0675..0678, 0E33, 0EB3, 0EDC..0EDD, 0F77, 0F79, FB00..FB06, FB13..FB17, FB4F	[CD]	[UnicodeData]
Letterforms folding	Variants of letter forms 017F (long s)	related base form 0073	[LetterFolding] TBD
Math symbol folding		[CD]	[UnicodeData]
Native digit folding	Nd	substitute ASCII digit of same numeric property	[UnicodeData]
Non-break folding	<no-break>	[CD]	[UnicodeData]
Overline folding	FE49..FE4C	[CD] maps to: 203E	[UnicodeData]
Positional Forms folding - includes Arabic ligatures	<initial>, <medial>, <final>, <isolate>	[CD]	[UnicodeData]
Small forms folding	<small>	[CD]	[UnicodeData]
Space folding	Zs	U+0020	[UnicodeData]
Spacing Accents <+>	00AF,00B4,00B8,02D8..02DD, 037A,0384,1FBD,1FBE..1FC0, 1FFE,2017,203E,309B..309C	[CD]	[UnicodeData]
Subscript folding	<sub>	[CD]	[UnicodeData]
Superscript folding	<super>, plus modifier letters 02C0..02C1,06E5..06E6	[CD] with some additions	[SuperFolding] TBD
Symbol folding <+>	00B5, 2107,2135..2138	[CD]	[UnicodeData]
Underline folding	2017, FE4D..FE4F	005E	[UnicodeData]
Vertical forms folding	<vertical>	[CD]	[UnicodeData]
Width folding	<wide>, <narrow>	[CD]	[Width]
Multigraph expansions	[This breakdown may change]		
- Fraction expansion	<fraction>	[CD]	[UnicodeData]
- Circled	<circled>	[CD]	[UnicodeData]
- Parenthesized	2474..2487,249C..24B5, 3200..3243	[CD]	[UnicodeData]
- Dotted	2488..249B	[CD]	[UnicodeData]
- Ellipsis expansion	2024..2026	[CD]	[UnicodeData]
- Integral expansion	222C..222D,222F..2230	[CD]	[UnicodeData]
- Prime expansion	2033..2034,2036..2037	[CD]	[UnicodeData]
- Roman numerals	2160..2183	[CD]	[UnicodeData]
- Squared	<square>	[CD]	[UnicodeData]
- Squared (unmarked)	3358..3370, 33E0..33FE, 32C0..32CB	[CD]	[UnicodeData]
- Digraphs	0132..0133, 013F..0140. 0149, 01C4..01CC, 01F1..01F3, 1E9A	[CD]	[UnicodeData]
- Other multigraphs, e.g. c/o, TEL	203C, 2047..2049, 20A8, 2100..2101, 2103, 2105..2106,	[CD]	[UnicodeData]

2109, 2116, 2121		
------------------	--	--

Notes:

- some transformations (such as case, width, Kana) can be applied forward and backward, but the folding are defined in a preferred default direction.
- compatibility decomposition for Han Radicals is inconsistent and should not be used for Radical folding
- some targets (such as 203E) are also subject to another folding, other than case folding [*Ed.: check Arabic in light of the last bullet or add section on interaction of foldings*]

[Note to reviewers:

Generally, where new data tables are needed to implement a folding these are either left as TBD, or in a few cases existing datafiles from other sources, but similar to what would be needed are referenced. As of now, the table intends to cover the repertoire of Unicode 3.2]

6.0 Notes and Guidelines

Here are notes and guidelines.

6.1 General notes

Most important: Choose! Do not apply **all** of these folding all at once.

6.1.1 Case folding

The [[CaseFolding](#)] data file provides case folding information. For more information see Technical Report #21 [Case Mapping](#) [[CaseMapping](#)].

6.1.2 Diacritic folding

Diacritic folding goes beyond the decomposition and removal of accents, umlauts, cedillas etc. that is provided by the canonical decompositions, but also includes barred, slashed forms etc, as well as hooks, descenders, etc.

6.1.3 Letter forms

Greek letter forms should be folded for Greek text. They should **not** be folded for mathematical and scientific usage as doing so would conflate very distinct concepts. To give an example of common usage consider physics, which distinguishes angle encoded by *theta* and temperature encoded by *theta symbol*.

[Ed. note: Letterforms need to have 'final sigma' and 'final Hebrew' added to them.]

6.1.4 Kana Foldings

Japanese Katakana and Hiragana are two generally equivalent syllabaries, where each Hiragana syllable has a corresponding Katakana syllable. However, since Katakana are used to represent the pronunciation of foreign words in Japanese, there are more Katakana than Hiragana characters. Therefore, the folding is from Hiragana to Katakana.

There is one important difference in orthography between the syllabaries, affecting the way the long syllables are expressed. Hiragana uses an additional vowel, while Katakana uses a length mark. If this is taken into account, the folding is no longer context free.

[Ed. note: check these statements. Is there a preference for one direction (for the purposes of searches), given the 3.2 extensions are there still Katakana than can be mapped in Hiragana.]

6.1.5 Semantically neutral foldings

Semantically neutral foldings, could be defined as those foldings that simply remove a distinction that is more or less purely an artifact of the encoding itself. Under the right circumstances, these foldings are in principle candidates for permanent data transformation. This is strictly, and always true for the canonical decomposition and composition, but could also apply to text using the Arabic positional forms. If such a text is converted to use non-positional forms, but rendered via a standard Unicode rendering process, the appearance would be the same (except for deliberately odd combinations of positional shapes). In practice it is far more likely that the data containing the positional forms will display poorly on such a system.

The following foldings are semantically neutral

- Positional forms folding
- Vertical forms folding
- Canonical decomposition and composition, in particular Normalization Forms NFC and NFD

[this section may need further improvement]

6.1.6 Foldings based on tailored collation data

Foldings based on tailored collation data would fold characters that are 'nearly equivalent' in a particular language. For example, a locale-based folding for Swedish could follow common practice in Sweden and match the following pairs of character sequences, among others, based on equivalence in pronunciation.

ä	æ
ö	ø

ss	ß
y	ÿ
v	w

In other words, locale-based foldings would be different for some user groups using the same script (in this case Latin). The recommended way to implement locale-based searching based on sorting tables is found in [\[Collation\]](#).

6.1.7 Compatibility decompositions

Compatibility decomposition provides a fixed combination of several foldings and expansions. It is in fact the source of most of the foldings in the table in section 4.0. There are two ways to subdivide compatibility decompositions:

1. by compatibility decomposition type (the value in `<>` in the data file, e.g. `<super>`)
2. by explicitly limiting the range of *source* characters

The specifications in section 4.0 uses the first method, whenever the compatibility tag is well defined and meaningful. Where it is too broad, e.g., for the `<compat>` tag, foldings are further subdivided by defining specific ranges of source characters.

Using compatibility decomposition is convenient, since existing algorithms for Normalization may provide them. However, the full decomposition includes several foldings that may not be appropriate for the given purpose. By selectively not applying the decomposition to certain character ranges given in Section 4.0, one can in effect limit the compatibility decomposition to only the desired foldings.

6.2 Problematic foldings or expansions

Some foldings can have unintended consequences, including inadvertent changes in the semantics of the text. In most cases, it is best to be conservative and avoid problematic foldings altogether. There are two general exceptions to this rule. The first is the case of identifier matching. If a folding has a prohibited character as one of the output characters, it will not match any legal identifier. Therefore, for properly restricted inputs, one may safely use fixed combinations of foldings, such as NFKC. The other exception is the case of more extensive string pre-processing, discussed below.

6.2.1 Fraction expansion

Fraction expansion as defined in the compatibility decompositions can lead to a drastic change of the semantics of a string and can lead to term boundary issues for searching. For example: Expanding the fraction in this string: DIGIT 5 + VULGAR FRACTION ONE QUARTER turns it into DIGIT 5 + DIGIT 1 + FRACTION SLASH + DIGIT 4. This now will be found by a search for "51". Because of the semantics of FRACTION SLASH the expansion changed the numeric value from "5

and a quarter" into "51 over 4". Fraction expansion is therefore best avoided altogether.

By modifying the fraction expansion from the standard compatibility decomposition and inserting an appropriate space character, for example THIN SPACE, before the fraction, it is possible to prevent the expanded fraction from coalescing with preceding digits. However, if there are no preceding digits, no THIN SPACE must be added, or strings containing the expanded fraction would no longer match strings with already expanded fractions which presumably would not contain THIN SPACE characters. Finally, any space character would be subject to any space folding, which, if present would introduce a SPACE character, possibly affecting the search term.

6.2.2 Bullet expansions

If a circled bullet character is simply replaced by its contents, as when CIRCLED DIGIT 5 is replaced by DIGIT 5, the separation from the surrounding text is lost, and the DIGIT 5 could run together with adjacent numbers. For bullet characters using parenthesized or dotted letters or digit, this issue is somewhat mitigated by fact that the bullet itself contains punctuation. Bullet characters are commonly used like footnote marks to refer to other text, in other words, they do not occur just at the beginning of bulleted lines.

6.2.3 Spacing accents substitution

Spacing accents are mapped by compatibility decomposition to SPACE followed by a non-spacing accent. This inappropriately introduces a space character into the term, as well as introducing non-spacing marks where none were in the data before. The former is especially problematic, where the matching operation is affected by these spaces and combining characters.

6.2.4 Math folding

The set of compatibility decompositions includes the folding of letterlike mathematical symbols to their nearest ASCII or Hebrew equivalent. In particular the Hebrew characters used as letterlike symbols do not have RIGHT TO LEFT directionality and the set of such letters in mathematical usage is sufficiently restricted that such folding makes little sense in math, except in pure 'looks like' style searches.

6.2.5 Various "cluster" expansions

Unicode contains many clusters, *e.g.* square symbols, some of the letterlike characters that are made up of several characters. 'Decomposing' these may or may not be the right thing for search equivalence. Parenthesized characters and numbers would probably be immune to the term boundaries issues raised earlier, but the story is less clear for others.

6.2.5 Jamo expansion

For more information on Jamo expansion see [section 10.4 Hangul \(addition\)](#) in [Unicode 3.2].

6.2.6 Preserving semantics

To a limited extent, the problems surrounding bullet expansion can be mitigated by inserting a THIN SPACE around the expansion to set the expanded text off from the surrounding text. However, this cannot be applied together with any space folding, as otherwise the THIN SPACE may become SPACE and might be considered a search term delimiter.

References

- [CaseFolding] Data file <ftp://ftp.unicode.org/Public/UNIDATA/CaseFolding.txt>
- [Case Mapping] Mark Davis, Unicode Technical Report #21: Case Mapping, <http://www.unicode.org/unicode/reports/tr21>
- [Collation] Mark Davis, Unicode Technical Report #10: Collation, <http://www.unicode.org/unicode/reports/tr10>>
- [Aliases] <http://www.unicode.org/Public/UNIDATA/PropertyValueAliases.txt>
- [Charts] The online code charts can be found at <http://www.unicode.org/charts/> An index to characters names with links to the corresponding chart is found at <http://www.unicode.org/charts/charindex.html>
- [DiacriticRemoval] ***Ed. The data file containing the data for Diacritic removal is TBD.***
- [EAW] Unicode Standard Annex #11, *East Asian Width*. <http://www.unicode.org/unicode/reports/tr11>
For a definition of East Asian Width
- [FAQ] Unicode Frequently Asked Questions <http://www.unicode.org/unicode/faq/>
For answers to common questions on technical issues.
- [Glossary] Unicode Glossary <http://www.unicode.org/glossary/>
For explanations of terminology used in this and other documents.
- [Normalization] Unicode Standard Annex #15: *Unicode Normalization Forms* <http://www.unicode.org/unicode/reports/tr15/>
- [Kana] For an example of a data file implementing a kana folding, see http://oss.software.ibm.com/cvs/icu4j/icu4j/src/com/ibm/icu/impl/data/Transliterator_Hiragana_Katakana.txt?rev=1.5&content-type=text/x-cvsweb-markup
- [Width] For an example of a data file implementing a width folding, see http://oss.software.ibm.com/cvs/icu4j/icu4j/src/com/ibm/icu/impl/data/Transliterator_Fullwidth_Halfwidth.txt?rev=1.3&content-type=text/x-cvsweb-markup
- [Reports] Unicode Technical Reports <http://www.unicode.org/unicode/reports/>
For information on the status and development process for technical reports, and for a list of technical reports.
- [SuperFolding] ***Ed.: Superscript folding needs to cover not only characters given a <super> compatibility decomposition, but also other superscript characters, like modifier letters. A data file for the folding is TBD.***

- [TUS] *The Unicode Standard, Version 3.0*, (Reading, Massachusetts: Addison-Wesley Developers Press 2000) or online as <http://www.unicode.org/unicode/uni2book/u2.html>
- [U3.1] Unicode Standard Annex #27: *Unicode 3.1*
<http://www.unicode.org/unicode/reports/tr27/>
- [U3.2] Unicode Standard Annex #28: *Unicode 3.2*
<http://www.unicode.org/unicode/reports/tr28/>
- [UCD] Unicode Character Database.
<http://www.unicode.org/Public/UNIDATA/UnicodeCharacterDatabase.html>
For and overview of the Unicode Character Database and a list of its associated files
- [UXML] Unicode Technical Report #20: *Unicode in XML and other Markup Languages*
<http://www.unicode.org/unicode/reports/tr20/>
- [Versions] Versions of the Unicode Standard
<http://www.unicode.org/unicode/standard/versions/>
For details on the precise contents of each version of the Unicode Standard, and how to cite them.
- [XML] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, Eds., *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation 6-October-2000,
<<http://www.w3.org/TR/REC-xml>>

Modifications

Changes from Tracking Number 0

First version

Copyright © 2001–2002 Unicode, Inc. All Rights Reserved. The Unicode Consortium makes no expressed or implied warranty of any kind, and assumes no liability for errors or omissions. No liability is assumed for incidental and consequential damages in connection with or arising out of the use of the information or programs contained or accompanying this technical report.

Unicode and the Unicode logo are trademarks of Unicode, Inc., and are registered in some jurisdictions.