**DRAFT Unicode technical note NN**

# Auxiliary character decompositions for supporting Hangul

Kent Karlsson
2006-09-24

## 1 Introduction

The Hangul script is very elegantly designed. There are just a small number of letters (28, plus a small number of variant letters introduced later, but the latter have fallen out of use) and even a featural design philosophy for the shapes of the letters.

However, the incarnation of Hangul as characters in ISO/IEC 10646 and Unicode is not so elegant. In particular, there are many Hangul characters that are not needed, for precomposed letter clusters as well as precomposed syllable characters. The precomposed syllables have arithmetically specified canonical decompositions into Hangul jamos (conjoining Hangul letters). But unfortunately the letter cluster Hangul jamos do not have canonical decompositions to their constituent letters, which they should have had. This leads to multiple representations for exactly the same sequence of letters. There is not even any compatibility-like distinction; i.e. no (intended) font difference, no (intended) width difference, no (intended) ligaturing difference of any kind. They have even lost the compatibility decompositions that they had in Unicode 2.0. There are also some problems with the Hangul compatibility letters, and their proper compatibility decompositions to Hangul jamo characters. Just following their compatibility decompositions in *UnicodeData.txt* does *not* give any useful results in any setting.

In this paper and its two associated datafiles these problems are addressed. Note that **no changes** to the standard Unicode normal forms (NFD, NFC, NFKD, and NFKC) are proposed, since these normal forms are stable for already allocated characters. Instead the decomposition mappings in the first datafile associated with this paper can be used as a correction tailoring of the normative decomposition mappings, using additional and replacement canonical and compatibility decomposition mappings listed in the data file *AuxiliaryHangulDecompositions.txt*. **These decomposition mappings are not proposed for *UnicodeData.txt*.**

The Korean standard KS X 1001 allows for Hangul syllables to be represented as particular sequences of otherwise non-conjoining Hangul letters, but interpreted in a conjoining way if the syntax is fulfilled. To arrive at the proper interpretation for these sequences via a table of compatibility-like decomposition mappings, multiple decomposition mappings are needed for each compatibility Hangul letter. One of the multiple decomposition mappings for a Hangul compatibility character is used, depending on the character's position in the syntactic sequence instance. These context dependent decomposition mappings are listed in *AuxiliaryKSX1001Decompositions.txt*. This is to be used on top of the auxiliary decomposition mappings of *AuxiliaryHangulDecompositions.txt*.

Most applications can ignore the context dependent decomposition mappings completely. Indeed the best use of *AuxiliaryKSX1001Decompositions.txt* is by character encoding conversion programs, using the context dependent decomposition mappings as well as the standard (arithmetically specified) and auxiliary canonical compositions (specified in *AuxiliaryHangulDecompositions.txt*). Note though, that such legacy data, that actually use such KS X 1001 sequences and not only Hangul syllable characters, may be fairly rare.

## *1.1 Letter Hangul jamo characters*

> "In the winter of our year 1443-4, our King [Seycong] originated and designed the twenty eight letters of the Correct Sounds. The letters are simple and fine and very easy to learn; their shifts and changes in function are endless; and there are no [Korean] sounds that cannot be written."
>
> *[Ceng Inci, in Hwunmin Cengum Haylyey, 1446; as translated in 'The Korean Language' by Ho-Min Sohn, Cambridge University Press,1999; the clarifications in brackets are mine.]*

A *single-letter Hangul jamo* character represents a basic conjoining Hangul letter, or a variant of such a letter. There are 17 basic consonant letters, here given in the order as first presented:

ㄱ, ㅋ, ㅇ, ㄷ, ㅌ, ㄴ, ㅂ, ㅍ, ㅁ, ㅈ, ㅊ, ㅅ, ㅎ, ㆆ, ㅇ, ㄹ, ㅿ

(the glyph for ㅇ (yesieung) should have a clear 'shoot' on top), and 11 basic vowel letters, again in the order as first presented:

ㆍ, ㅡ, ㅣ, ㅗ, ㅏ, ㅜ, ㅓ, ㅛ, ㅑ, ㅠ, ㅕ

(originally the short dashes now attached to the long dashes were dots, and dot is now often drawn as a short vertical dash). Some of the consonant letters have variants that were invented after the invention of Hangul (the following variants have shortened right or left leg):

ㆻ and ㆽ as variants of ㅈ,

ㆼ and ㆾ as variants of ㅊ,

ㅥ and ㅦ as variants of ㅅ,

From the very beginning, in the document introducing the then-new script, letter combinations were used for denoting sounds in Korean, Chinese, Mongolian, and Japanese. For instance: doubled consonants for tense pronunciation, ㅇ (ieung) below for light pronunciation (that particular letter combination did not catch on), and vowel letter combinations for diphthongs as well as single vowel sounds.

There may be a small number of historic variant letters that are still missing as encoded characters, e.g. 45 degree turned mium (ㅁ). There are also two Korean combining punctuation characters that are missing as coded characters.

The variants, as well as a few of the original basic letters, have fallen out of use. For example, ㅇ (YESIEUNG, a small circle with a 'shoot' on top), for the sound "ng", has in the trailing position been replaced by ㅇ (IEUNG, a small circle), originally silent, as it still is in the leading position. ㆆ, ㅿ and ㆍ represent sounds not present in modern official Korean, though the sounds are still present in some dialects. As noted above certain letter combinations stand for separate consonantal or vowel sounds that do not have their own letter. This is similar to what is often done on the Latin script, like "sh" and "ng" and others. Originally, there were also two tone marks, applicable to a syllable block. The tone marks have fallen out of use, as have the (pre-Hangul) Korean punctuation marks.

The letters for a syllable are grouped into a "syllable block", typographically the size of a Hàn ideograph. In practice, there are at most (in total, however represented, see below) three consonant letters in a consonants cluster, and at most (in total, however represented) three vowel letters in a vowel cluster. Note that some vowel combinations

look very much the same (like e.g. A-I and I-EO, EU-YO and YU-EU), especially as they are normally typeset so close that the letters lightly touch. But for each such same-looking pair of vowel pairs, only one of those vowel pairs is used.

The encoding as characters in Unicode/10646 for Hangul jamo employ a little coding trick to make the determination of syllable boundaries simple: the consonants are encoded twice, leading and trailing. The leading consonants have the Unicode property *Hangul_Syllable_Type* = **L**, the training consonant clones have *Hangul_Syllable_Type* = **T**, while the vowels have *Hangul_Syllable_Type* = **V**. Other possible ways that could have been used to encode syllable blocks include:

(a) Using an initiator/separator/terminator character before/between/after syllables. A similar approach is sometimes used for the Hangul compatibility letters, see below, using an initiator.

(b) Using combining characters for the Hangul letters that follow the first one in a syllable. This was the original Unicode design for Hangul, and it is somewhat similar to the approach chosen; compare also the *combining Latin letters above* that have been encoded.

Note that the current model, with conjoining jamo characters, can be seen as base + combining characters for a Hangul syllable block, if we employ the following definition of 'conjoining jamo':

- A *Choseong Jamo* character that immediately follows a Choseong Jamo character shall be treated as a combining character.
- A *Jungseong Jamo* character that immediately follows a Choseong Jamo character, a Jungseong Jamo character, or a precomposed Hangul syllable character that has no Jongseong part (Hangul syllable type **LV**) shall be treated as a combining character.
- A *Jongseong Jamo* character that immediately follows a Jungseong Jamo character, a Jongseong Jamo character, or a precomposed Hangul syllable character shall be treated as a combining character.
- A *precomposed Hangul syllable* character that immediately follows a Choseong Jamo character shall be treated as a combining character.
- All other instances of a Choseong Jamo character, a Jungseong Jamo character, a Jongseong Jamo character, or a precomposed Hangul syllable character shall be treated as a base character (i.e. not combining).

This makes a Hangul syllable block (including combining characters applied to it) to be a special case of combining sequence. Note that the lead consonants cannot be regarded as true base characters, since the lead part may consist of several lead consonants (see below), and should be followed by at least one vowel conjoining jamo. Similarly, the vowel and trail consonant jamos cannot be regarded as true combining characters.

## *1.2 Basic syntax of Hangul syllables*

A well-formed Hangul syllable (one with a lead consonant and a vowel) has the following syntax (here disregarding precomposed Hangul syllable characters, but see below):

Hangul-syllable **::=** L+ V+ T* M*

where L stands for a leading consonant jamo character, V stands for a vowel jamo character, T stands for a trailing consonant jamo character, and M stands for a combining

mark, in particular a (historic) Hangul tone mark (U+302E, U+302F). Note that there may be several lead consonants, several vowels, and several trailing consonants. The tone mark (if any, or more generally, the sequence of combining characters) applies to the entire preceding syllable, not just the last part of it, since the Hangul syllable components, including the precomposed Hangul syllable characters, are conjoining characters, not really base characters. The tone mark glyphically appears at the left of a syllable, so for a L L V T M syllable, where M stands for a Hangul tone mark, the glyph for M is to be rendered to the left of the (possibly composed) glyph for L L V T, not to the left of the (sub)glyph for T.

Typographically, the leading consonants are put in the top left part or the top part of a syllable block depending on the shape of a following vowel or vowel cluster, normally left to right; the vowels are put under, to the right of, or under and to the right of the leading consonants, left to right or top-down; and trailing consonants are put at the bottom of the syllable block, normally left to right. An exception is when IEUNG is after consonant (other than IEUNG) or a doubled consonant: IEUNG is then placed under juet that consonant or doubled consonant before it (see the sample glyphs for U+111D, U+112B, U+112C, U+1158 and their counterparts in the final consonant block for how such compositions are laid out). Note that I-ARAEA is laid out vertically, i.e. ARAEA is a horizontal vowel, except after ARAEA.

In addition, two filler jamos are included in the encoded repertoire: HANGUL CHOSEONG FILLER (Lf), i.e. lead filler, and HANGUL JUNGSEONG FILLER (Vf), i.e. vowel filler. They do not stand for any letters, so that they have general category **Lo** is misleading. Indeed, they also have the property of being *Default_Ignorable_Codepoint* (but they are not ignorable when finding Hangul syllable block boundaries, nor for collation). (In a combining characters approach, the lead filler would have been replaced by a no-break space, and the vowel filler would just not be needed at all). They are used as placeholders for missing letters, where there should be at least one letter. Note that there has to be at least one lead consonant and at least one vowel in a well-formed Hangul syllable according to the syntax above. There is no *jongseong* (trail) filler, so the fillers are not intended as placeholders *inside* a consonants cluster or vowels cluster (but we will allow a choseong filler between a non-ieung choseong consonant and a choseong IEUNG, but only for a hypothetical special case). The two Hangul jamo filler characters can in many cases be inserted automatically into one or two non-well-formed Hangul syllables to fullfill (L+|Lf) (V+|Vf) T* M* (where neither L nor T are fillers; see The Unicode Standard version 4.0), but the result might not be what was intended. The fillers are not part of the orthography. They are used only to fullfill the representation recommendation also for ‘incomplete’ syllables which are often necessary when explaining about word roots and other elements that play important roles in Korean grammar.

What has been presented so far here is fully sufficient for representing any text in Hangul: historical (except for as yet missing historic variants, if any), modern, and future (unless new letters are invented, which has already happened in at least one case, see [8], and gain use).

Note that Hangul is a very elegantly designed *alphabetic* script. Typographic features, such as cluster ligatures, variant (sub)glyph selection (each letter need to have a few handfulls of variant glyphs to be used depending on which syllable the letter is in), and syllable layout should be handled by rendering and font mechanisms. Modern font technologies (like Opentype with Uniscribe, AAT or Opentype with ATSUI, Pango, and Graphite) make this much more technically accessible than before. Hangul is not an ideographic script at all, even though the typographic design of the syllable blocks look a bit like ideographs, they are quite distinct but fit well to be mixed with Hàn ideographs.

The following subsections are about coded character additions that are *unnecessary* for the representation of Hangul, but have been added for various other reasons, such as compatibility with older standards. They generally introduce a number of difficulties for all processes handling Hangul in a non-trivial way.

## 1.3 Letter cluster Hangul jamo characters

Letter cluster Hangul jamo characters represent either clusters of two or three consonants (as introduced above), or clusters of two or three vowels (as introduced above). Cluster jamo characters for several (not all) consonant and vowel clusters *occurring*, or has occurred, in Hangul texts are allocated in Unicode/10646.

Asked for the list of Hangul letter clusters occurring in all known Korean text (published/written since the invention of Hangul) by Microsoft Korea, a group of Korean linguists came up with:

- 34 lead consonant clusters *in addition to* the 67 (90−23) lead consonant clusters already allocated;
- 28 vowel clusters *in addition to* the 55 (66−11) vowel clusters already allocated;
- 59 trail consonant clusters *in addition to* the 65 (82−17) trail consonant clusters already allocated.

This does not take into account the Hangul jamo letter variants that may still not be allocated in Unicode 5.0.

The letter cluster jamos work as L, V, or T respectively according to their *Hangul_Syllable_Type* in the syllable syntax given above. But as mentioned, one can preferably represent the sequence of consonants or sequence of vowels using single-letter Hangul jamo characters. The letter cluster Hangul jamos are thus not needed for representing Hangul texts, and no more letter cluster jamos should be allocated.

Note that all of the letter cluster jamos are completely unnecessary. Not even the doubled consonants constitute letters of their own, though that is sometimes claimed. The Hangul design document of 1446 is quite clear on the matter. That is, ssangkiyeok, for instance, in *not* another letter that looks like two kiyeok, it really *is* two kiyeok. The same goes for the kapyeoun- combinations: they are really compositions with ieung at the end (written below the 'kapyeoun-ed' letter or double letter), they are introduced that way in the design document of 1446.

There is a confusing similarity between ieung and yesiung, especially in modern fonts, and that the function of yesieung has been subsumed by ieung (in a trail position). Therefore, some of the -IEUNG named composite jamos may actually have a YESIEUNG as a component. This does not hold for CHOSEONG SSANGIEUNG, as that is clearly exemplified in the design document of 1446, it is really doubled IEUNG.

For the other IEUNG composites, there are two reasonable alternatives:

1. Treat all of them as IEUNG. That is consistent with the naming.
2. Treat all of them as YESIEUNG. That is what they most probably were intended to be, also for leading instances as yesieung (ng) *is* used leadingly.

Either way does not, in the end, matter all that much, since the "other" choice can always be expressed anyway. Therefore in the associated datafiles, option 2 has been applied.

It has been suggested that ieung, in a lead position, may have had special meaning. In particular that ieung may follow, but not be rendered under, another consonant. However,

there is no support in the design document of 1446 for IEUNG to be used that way. But it can be encoded as <L, HANGUL CHOSEONG FILLER, HANGUL CHOSEONG IEUNG> (in which case the filler is not ignorable).

## *1.4 Hangul compatibility letters*

The Hangul compatibility letters and half-width letters encode the consonants and some of the consonant clusters only once each, no duplicate encoding for lead and trail. The Hangul compatibility letters are normally rendered as spacing characters without any conjoinment. They may be used when talking about Hangul letters in isolation, when writing Hangul texts in 'linear form' or when using KS X 1001 sequences to represent Hangul syllables that do not have a precomposed syllable character in KS X 1001.

Writing Hangul in 'linear form' is obsolete, and 'linear Hangul' cannot be used in IDNs, since NFKD (and NFKC) is wrong for it. But, interestingly, HANGUL LETTER EU, may in 'linear form' be given in a U-like shape, see reference [6], page 39. The same reference page also has an above-script u-like diacritic to indicate that two vowel letters are combined to form a single vowel. Those two characters should be allocated in Unicode. Finally, leading IEUNGs are omitted in 'linear Hangul.

Each (full- or half-width) Hangul compatibility letter should normally be considered as a free-standing form, with compatibility mappings of the forms C → L+ Vf (or possibly even C → Lf Vf T+) and W → Lf V+; i.e. to Hangul syllables. In the associated datafile, the full-width forms are given "<free>" decompositions, since there is another representation for them: as Hangul syllables using CHOSEONG FILLER or JUNGSEONG FILLER, as given here. The latter is more general, but due to the prevalent use of (full-width) Hangul compatibility letters, the latter should be preferred when available. The "<free>" decompositions mappings are very close to auxiliary canonical decompositions, but the Hangul compatibility letters are not conjoining.

However, that leaves the (HALFWIDTH) HANGUL FILLERs useless. Indeed, they should not be rendered at all, despite that they have been given the property Lo. Note that these FILLERs are also given the property of *Default_Ignorable_Codepoint*.

The compatibility Hangul FILLER characters, U+3164 (HANGUL FILLER) and U+FFA0 (HALFWIDHT HANGUL FILLER), are there for compatibility with Korean standard KS X 1001. If at all interpreted (which is *not* expected for most applications), these filler characters work very differently from the jamo filler characters. When, and very much if, converting from Hangul compatibility letter sequences (according to KS X 1001) to proper conjoining Hangul letters (which may then be composed to precomposed Hangul syllable characters; there are more of them in Unicode/10646 than in KS X 1001), the Hangul compatibility syllables have the (generalised) syntax:

Hangul-compatibility-syllable ::= Hf (C+|Hf) (W+|Hf) (C+|Hf) M*

where C is a (possibly half-width) Hangul compatibility consonant letter or compatibility consonant letter cluster, W is a (possibly half-width) Hangul compatibility vowel letter or compatibility vowel letter cluster, and Hf is a (possibly half-width) compatibility Hangul FILLER character.

It may be reasonable to convert input in compatibility Hangul characters to a string of conjoining jamos (Hf (C+|Hf) (W+|Hf) (C+|Hf) M* → L+ V+ T* M*). This may be done in particular when converting from an encoding based on KS X 1001 (and conversely generate such sequences when converting to such an encoding). Note that the sequence starting Hf is to be removed (note again the FILLERs are *not* letters, despite their general category as Lo), so is a trailing Hf, while a lead Hf should be converted to a Lf, and a

vowel Hf should be converted to a Vf. Similar conversions (with other syntax) may be done as part of the working of a Hangul IME. Sequences of compatibility Hangul letters are not normally expected to display as conjoined Hangul syllables, but display as a sequence of freestanding letters. Note that the standard normal forms NFKD and NFKC do *not* do this conversion but return (in all views) incorrect results for strings containing these characters.

## 1.5 Hangul syllable characters

A lot of (far from all) Hangul syllables have a character of their own in the range AC00-D7A3. The allocated ones each have an arithmetic canonical decomposition into two (*choseong*, *jungseong*) or three (*choseong*, *jungseong*, *jongseong*) Hangul jamo characters in the ranges 1100-1112, 1161-1175, and 11A8-11C2. Some of the precomposed Hangul syllable characters thus decompose into a sequence of jamos where some of them are letter cluster jamos.

The Hangul syllable characters alone can represent modern Hangul words written in a way that is currently orthographically acceptable. They cannot represent all historic Hangul words (Middle Korean), nor can they represent a closer-to-pronunciation writing of all modern words. However, all Korean words can elegantly be represented by sequences of single-letter Hangul jamo characters plus optional tone mark.

Let SBase = 0xAC00, LBase = 0x1100, VBase = 0x1161, TBaseM1 = 0x11A7 (one less than the lowest code point for a modern Hangul jamo trail consonant (clusters)), VCount = 21, TCountP1 = 28 (one more than the number of trailing Hangul consonant (clusters) that occur in modern Korean), NCount = VCount * TCountP1 (588).

The arithmetic decompositions (Unicode 5.0) for precomposed Hangul syllable characters are as follows:

Each Hangul precomposed syllable character of *Hangul_Syllable_Type* **LV** has a canonical decomposition into **L** and **V** Hangul jamos:

| **LV** | | **L in** 1100–1112 | **V in** 1161–1175 |
|---|---|---|---|
| *s* | → | LBase + (*s* – SBase) **div** NCount) | VBase + ((*s* – SBase) **mod** NCount) **div** TCountP1) |

Each Hangul precomposed syllable character of *Hangul_Syllable_Type* **LVT** has a canonical decomposition into a **LV** Hangul syllable character and a **T** Hangul jamo:

| **LVT** | | **LV** | **T in** 11A8–11C2 |
|---|---|---|---|
| *s* | → | SBase + (((*s* – SBase) **div** NCount) * NCount) | TBaseM1 + ((*s* – SBase) **mod** TCountP1) |

Note: This description is slightly different from that in The Unicode Standard 3.0 and 4.0, but the net result for normalisation is the same. This description, with the intermediary step with **LV** for **LVT** syllable characters makes *recomposing* easier to describe, as it falls in line with the other tabular canonical decompositions, which are unary or binary, but never tertiary (or more) nor tertiary with a twist. Note that this limitation to binary canonical decompositions also holds for *AuxiliaryHangulDecompositions.txt*, so that the data (almost) fits with the normalisation algorithm described in UAX 15. (The “almost” part is covered below.)

The arithmetic decompositions imply arithmetic compositions for **L** and **V**:

| **LV** | | **L in** 1100–1112 | **V in** 1161–1175 |
|---|---|---|---|
| SBase + ((*a* – LBase) * NCount) + ((*b* – VBase) * TCountP1) | ← | *a* | *b* |

Similarly for **LV** and **T**:

| **LVT** | | **LV** | **T in** 11A8–11C2 |
|---|---|---|---|
| *c* + (*d* – TBaseM1) | ← | *c* | *d* |

Note that for maximal final composition these arithmetic composition rules should be applied to Hangul jamo substrings that are already maximally composed for modern letter combinations according to the tabular canonical composition rules for Hangul jamo (presented in this paper; the data is found in the accompanying file *AuxiliaryHangulDecompositions.txt*). See subsection 2.2 below.

### 1.6 Full syntax for Hangul syllables, including Hangul syllable characters

A Hangul syllable, allowing for precomposed syllable characters, has the following syntax (see page 86 of The Unicode Standard version 4.0, here with adjustment for tone marks):

Hangul-ext-syllable ::= L+ V+ T* M* | L* LV V* T* M* | L* LVT T* M*

where LV is a precomposed consonants-vowels Hangul syllable character, LVT is a precomposed consonants-vowels-consonants Hangul syllable character.

### 1.7 Circled and parenthesised Hangul letters and syllables

All of the parenthesised or circled Hangul characters should be treated as compatibility characters with a compatibility mapping to a proper syllable sequence of Hangul single-letter characters. I.e., they should not have mappings to just a single Hangul jamo letter even for the single-letter characters of this kind; but include a filler jamo to make a well-formed Hangul syllable instead or use compatibility letters in the decomposition.

## 2 Hangul jamo auxiliary canonical decomposition mappings

At one point (Unicode version 2.0) the letter cluster jamos had compatibility decompositions into single-letter jamos. These were removed due to problems with maintaining precomposed Hangul syllable characters in NFKC. Now there is, unfortunately, no longer any (canonical or compatibility) decomposition of the letter cluster jamos into single-letter jamos in *UnicodeData.txt*. This leads to multiple representations of exactly the same piece of Hangul text that cannot be normalised to the same string of code points with any of the standard normal forms. A better way to deal with this problem is to give auxiliary canonical decompositions into single-letter jamos for each letter cluster character. These are given in the *AuxiliaryHangulDecompositions.txt* datafile. The decomposition mappings that stay as they are listed in *UnicodeData.txt* are not mentioned in the auxiliary decompositions datafile.

Using the auxiliary canonical decomposition mappings, original text using Hangul syllable characters (but no Hangul jamo characters) are maintained in a modified NFC which is augmented with the auxiliary canonical decompositions. Whenever possible the cluster Hangul jamos should be treated as having these canonical decompositions into the corresponding sequence of single-letter Hangul jamos. These decompositions should also be used in inverted form (compare NFC and NFKC canonical combination step) when a maximally precomposed form is desired. However, the auxiliary decomposition mappings of *AuxiliaryHangulDecompositions.txt* **shall not** be used for computing any of the Unicode normal forms.

Spell checking, rendering, collation, identifier comparisons, etc. should use the auxiliary decompositions for an NFC-like modified normal form. Note, however, that for rendering, if a precomposed Hangul syllable is preceded by a CHOSEONG jamo, or the precomposed Hangul syllable is of type LVT and is followed by a JONGSEONG jamo or is of type LV and is followed by a JUNGSEONG jamo or a JONGSEONG jamo, then it is better to decompose the precomposed Hangul syllable character before the rendering.

### 2.1 Maximally decomposed Hangul forms

When computing a maximally decomposed form (compare NFD and NFKD) using the auxiliary decompositions presented here, the process works as the normal algorithm for computing NFD or NFKD respectively, except that the data in *AuxiliaryHangulDecompositions.txt* is used to override the some of the data in *UnicodeData.txt*.

### 2.2 Maximally composed Hangul forms

When computing a maximally composed form (compare NFC and NFKC) using the auxiliary decompositions presented here, the process works as the normal algorithm for computing NFC or NFKC respectively, except that the data in *AuxiliaryHangulDecompositions.txt* is used to override the some of the data in *UnicodeData.txt*, plus the following modification is done to the composition step.

One algorithmic description needs to be added, for determining a unique maximally composed auxiliary normal form for Hangul. Purely going from left to right when doing the compositing does not give maximal composition to precomposed characters. For instance, that would *not* maintain strings written *purely* in Hangul syllable characters, recreating the given Hangul syllable characters. So instead, the jamo and Hangul syllable characters are composed as follows:

1. First, compose Hangul jamo substrings according to the auxiliary tabular canonical decompositions. (Note that the composition process works from the start of the substring, composing two characters to one when possible, see UAX 15.)
2. Repeat step one; however, this is needed only if the first iteration produced <U+1169, U+1168> (<HANGUL JUNGSEONG O, HANGUL JUNGSEONG YE>) or produced <U+116E, U+1168> (<HANGUL JUNGSEONG U ,HANGUL JUNGSEONG YE>).
3. Finally, compose according to the arithmetic canonical decompositions of Hangul syllable characters (see section 1.5 above).

It is possible to reformulate the composition, and the decomposition mappings, so that these extra steps are not needed. However, that would mean allocating two more two-letter jamos (O-YEO, U-YEO), and replace the arithmetic decomposition of the precomposed Hangul syllable characters by a tabular specification of the (new) decomposition mappings.

## 3 Hangul compatibility letter auxiliary compatibility decomposition mappings

Compatibility Hangul letters (U+3131–U+318E) should be regarded as having the auxiliary compatibility decomposition mappings in *AuxiliaryHangulDecompositions.txt*. These decompositions shall not be used inverted.

Collation, case insensitive comparisons, spell checking, and some other processes should use the auxiliary decompositions for computing an NFKC-like normal form, **but they must not be used for NFKD nor NFKC**.

### 3.1 Full-width Hangul compatibility letters

The full-width Hangul compatibility letters should be interpreted in a free-standing context. They should then decompose into “incomplete” Hangul syllables, including an appropriate jamo filler character. Such decompositions are given in the datafile

*AuxiliaryHangulDecompositions.txt*. These decompositions are very close to canonical decompositions, except that the (full-width) compatibility letters are not conjoining.

### 3.2 Half-width Hangul compatibility letters

The compatibility mappings of the half-width compatibility letters are ok as they are in the *UnicodeData.txt* file. But when decomposed to the full-width form, the auxiliary decomposition mappings should then be used for further decomposition to Hangul jamos. If a more composed form is desired, then the auxiliary canonical decomposition mappings for the Hangul jamo characters are used in reverse (see above).

### 3.3 Other Hangul compatibility characters

The Hangul parenthesised and Hangul circled characters should in many cases (collation, for instance) be regarded as having the compatibility decompositions given in the datafile *AuxiliaryHangulDecompositions.txt*.

### 3.4 Sort order for Hangul

Ordering (sort order) of Hangul syllables should be based on a weighting scheme that orders cluster characters as sequences of single letters, i.e. as if they were maximally canonically decomposed. The clustering as such, as used for collation, should be achieved by other mechanisms (see reference [7]) than using letter cluster characters. Even so, the reason for allocating the letter cluster characters may have been one of the original reason for having these cluster characters, and why they did not get the canonical decompositions that they should have had from early on. The clustering is important to get the correct sort order for Hangul strings, both for historical clusters that do not have a letter cluster character, and in relation to e.g. Hàn characters that may be given in a string after a text in Hangul.

## 4 Hangul KS X 1001 conversion decompositions

KS X 1001 has a compatibility Hangul syntax: Hf C W (C|Hf); an initiator, lead consonants, vowels, and trail consonants. Note that Hf is used in such a way that, if this syntax is striclty followed, one can determine if an instance of a C is used as a lead or a trail consonant. This syntax can, without introducing any ambiguity, be generalised to Hf (C+|Hf) (W+|Hf) (C+|Hf). This kind of syllable analysis should *not* be commonly done for Hangul compatibility letters. However, if this syntax is analysed for, one gets the four parts:

a. starter part (a compatibility FILLER), which works as an initiator,
b. lead consonants part (at least one consonant letter (cluster) or a single FILLER),
c. vowels part (at least one vowel letter (cluster) or a single FILLER), and
d. trail consonants part (at least one consonant letter (cluster) or a single FILLER).

### 4.1 Full-width and half-width Hangul compatibility letters

The full-width Hangul compatibility letters should in the context of KS X 1001 Hangul composition (generalised) syntax be interpreted as having mappings directly to jamo letters (no fillers). These special decompositions could also be useful in a Hangul IME, that converts compatibility letters typed on a keyboard into Hangul jamos. The interpretatiuon may be slighly different, since one would not want to impose the exact KS X 1001 syntax on such an application. In other cases their auxiliary <compat> decompositions should be used instead. The KS X 1001 decompositions are syntactic position context dependent. They are given in the datafile *AuxiliaryKSX1001Decompositions.txt*.

The HANGUL FILLER has four conversion decomposition contexts in addition to the free-standing one (<free> or <compat> in *AuxiliaryHangulDecompositions.txt*):

<starter>, <lead>, <vowels>, and <trail>

The <vowels> context is mistakenly the only one given in the *UnicodeData.txt* file for its <compat> decomposition of this character.

The Hangul compatibility consonant letters have two conversion decomposition contexts in addition to the free-standing one (<free> or <compat> in *AuxiliaryHangulDecompositions.txt*):

<lead> and <trail>

The *UnicodeData.txt* <compat> decompositions mistakenly for these vary between the <lead> (which was used, apart from the tag, when a single choseong was available) and <trail> decompositions.

The Hangul compatibility vowels letters have one new decomposition context in addition to the free-standing one (<free> or <compat> in *AuxiliaryHangulDecompositions.txt*):

<vowels>

Note that the *UnicodeData.txt* <compat> decomposition for these mistakenly are the same (apart from the label) as the <vowels> decomposition.

The new decompositions in *AuxiliaryHangulDecompositions.txt*, which give the decompositions for the free-standing context, together with the contextual decompositions in *AuxiliaryKSX1001Decompositions.txt* should be used together as replacement to the *UnicodeData.txt* decompositions, **except for computing standard normal forms**. The contexts are <free> (for free-standing) (or <compat> for the half-width forms), <lead>, <vowels>, <trail>, and <starter> (for KS X 1001 conversion).

# 5 Use of the auxiliary decomposition mappings

The KS X 1001 *AuxiliaryKSX1001Decompositions.txt* data should be used only by character encoding conversions, Korean IMEs, or similar.

The use of *AuxiliaryHangulDecompositions.txt* is not so straightforward. As has been mentioned a number of times above, they cannot be used for the standard normal forms, since those are stabilised for sequences of already allocated characters. But we can define auxiliary Hangul normal forms. The use of which will be described below.

- NFD-Hangul is produced in the same way as NFD, but using the auxiliary canonical decompositions overriding some of the decomposition mappings of *UnicodeData.txt*.
- NFC-Hangul is produced in the same way as NFC, but using the auxiliary canonical decompositions overriding some of the decomposition mappings of *UnicodeData.txt*, and the algorithm adjustment described in section 2.1. (NFC-Hangulplus: recompose using also the <free> decompositions as a last recomposition step?)
- NFKD-Hangul is produced in the same way as NFKD, but using the auxiliary canonical and compatibility decompositions overriding some of the decomposition mappings of *UnicodeData.txt*.
- NFKC-Hangul is produced in the same way as NFKC, but using the auxiliary canonical and compatibility decompositions overriding some of the decomposition mappings of *UnicodeData.txt*, and the algorithm adjustment described in section 2.1.

(NFC-Hangulplus: recompose using also the <free> decompositions as a last recomposition step?)

Since the different representations of Hangul strings that have the same NFD-Hangul form are really equivalent spellings, and they do not necessarily have the same NFD form, there are multiple representations of exactly the same spelled Hangul syllables. Indeed, the HANGUL LETTERs also constitute a way of writing certain so-called incomplete Hangul syllables, representations that are confusable with using Hangul jamos and Hangul jamo fillers. To prevent spoofing, certain representation should be rendered properly, while other (not canonically equivalent) representations of exactly the same Hangul syllable should be rendered in a blotted out fashion to indicate that that representation, if rendered properly, would look just like the same syllable represented differently.

One needs to draw a line between what to render properly and what to render blotted out. A decision for the Unicode standard should be made for which representations are ok for rendering and for security sensitive applications, and which are not, so that all implementations can draw the line in the same way. We will here look into some alternatives for how to draw that line (for the Unicode standard, not for individual implementations to decide differently).

It is not quite clear if using the jamo filler characters is in principle confusable with using the Hangul compatibility letters. Is, for instance, <U+1100, U+1160> in principle confusable with U+3131? It is conceivable that <U+1100, U+1160> is intended to be displayed differently from U+3131. But below, it is assumed that they are confusable and that the U+ 3131 representation is preferred (since it is prevalent) and the other should be blocked as a possible spoof. The half-width Hangul letters are assumed to be distinguishable from the full-width ones.

To make it easy to refer to full-width compatibility Hangul letters below, the Hangul letters are given "<free>" (for free-standing) decompositions (instead of <compat> decomposition mappings) in the datafile *AuxiliaryHangulDecompositions.txt*.

Using Hangul syllables with the sequence <U+115F, U+1160> (<CHOSEONG FILLER, JUNGSEONG FILLER>), or syllables which should have that sequence (containing just trail consonants), should be regarded as a possible spoof and should get a blotted out display, as well as any extraneous jamo fillers (e.g. multiple fillers, or fillers inside of a choseong sequence (except for a special case between a non-IEUNG and IEUNG) or inside a jungseong sequence). Thus the following cases should result in a blotted out glyph, or just plain rejection, for the Hangul syllable (where choseong includes the choseong filler and jungseong includes the jungseong filler):

- <U+115F, U+1160> in the syllable,
- <choseong, U+115F> or <U+115F, choseong> in the syllable, except for <non-ieung and non-filler choseong, U+115F, HANGUL CHOSEONG IEUNG> (for preventing the IEUNG to be displayed below the preceding choseong or double choseong),
- <jungseong, U+1160> or <U+1160, jungseong> in the syllable,
- more cases that should result in blotting out, or just plain rejection, are given below.

Let "*syll*" stand for a (not necessarily well-formed) Hangul syllable (maximal non-empty substring fulfilling L* V* T*, L* LV V* T*, or L* LVT T*).

Note that a *syll* **shall** be displayed as a **single** syllable block. Dividing it up into several syllable blocks would be an error, and doing so would indeed be a security issue.

In each of the following alternatives, if *syll* (in its entirety) is canonically equivalent to a precomposed Hangul syllable, that representation is ok, and should be displayed using the font's glyph for that precomposed Hangul syllable. Other ok *syll*:s need jamo glyph composition, and not ok *syll*:s should get a blotted out display as a possible spoof combination of characters.

Note that Unicode so far has had no recommendation for which combinations of Hangul jamos are ok, except to say that Hangul syllables, however constructed, are to be displayed in a single block and define the boundaries of syllables in a character sequence. No word on which among several logically equivalent (but not canonically equivalent) representations to use.

Here are some possible alternatives *for the standard*; they are **not** implementer's choices:

**Alternative 1.** Compute (a) NFC-Hangul(*syll*), and compute (b) NFC(*syll*).

If (a) and (b) have the same sequence of code points and that sequence of code points is not the same as the NFKC-Hangul of any of the Hangul characters with a <free> decomposition, and it does not misuse jamo fillers (see above), then the *syll* may be ok. Otherwise: treat the *syll* as a possible spoof (blotted out display or otherwise blocked).

*This alternative prefers using multi-letter jamos whenever possible. While it "allows" use of all already allocated multi-letter jamos, it is also for that reason very clumsy.*

**Alternative 2.** Compute (a) NFD-Hangul(*syll*) plus do auxiliary compose to characters named in *Jamo.txt* (i.e., other auxiliary compositions excluded), and compute (b) NFD(*syll*).

If (a) and (b) have the same sequence of code points and that sequence of code points is not the same as the NFKD-Hangul plus auxiliary composition to characters named in *Jamo.txt* of any of the Hangul characters with a <free> decomposition, and it does not misuse jamo fillers (see above), then the *syll* may be ok. Otherwise: treat the *syll* as a possible spoof (e.g. blotted out display or otherwise blocked).

*This alternative prefers using single-letter jamos, except for those that can occur in precomposed Hangul syllable characters. It is slightly clumsy since it can leave certain multi-letter jamos outside of precomposed Hangul syllable characters in NFC.*

**Alternative 3.** Compute (a) NFC-Hangul(*syll*) plus do auxiliary decompose for jamo characters (without decomposing the precomposed Hangul syllable characters), and compute (b) NFC(*syll*).

If (a) and (b) have the same sequence of code points and that sequence of code points is not the same as the NFKC-Hangul of any of the Hangul characters with a <free> decomposition, and it does not misuse jamo fillers (see above), then the *syll* may be ok. Otherwise: treat the *syll* as a possible spoof (blotted out display or otherwise blocked).

*This alternative prefers using single-letter jamos, except for those that actually occur in a precomposable Hangul sub-syllable.*

**Alternative 4.** Compute (a) NFC-Hangul(*syll*) and (b) NFC(*syll*); but if that leaves any jamos in the representation of (a): recompute to get (a) NFD-Hangul(*syll*) and (b) NFD(*syll*).

If (a) and (b) have the same sequence of code points and that sequence of code points is not the same as the NFKC-Hangul (if there is no recomputation) or NFKD-Hangul (if there is recomputation) of any of the Hangul characters with a <free> decomposition, and it does not misuse jamo fillers (see above), then the *syll* may be ok. Otherwise: treat the *syll* as a possible spoof (blotted out display or otherwise blocked).

*This alternative prefers using single-letter jamos, except for those that actually occur in a precomposable (and here precomposed) full Hangul syllable. This is the most elegant alternative, while still allowing for the prevalent use of precomposed Hangul syllable characters as well as HANGUL LETTERs.*

**The last alternative (4) is the alternative preferred by the author.** It is elegant in that it uses precomposed Hangul syllables or Hangul compatibility letters whenever possible, but only for full syllables that can be represented that way, and otherwise uses just single-letter jamos). There is a technical implementation feasibility reason given below.

There are some additional requirements for proper display in a syllable block:

- at most three lead consonant letters
  - displayed to the left of eachother, however,
  - note that a ieung is displayed below the preceding lead letter or doubled lead letter (yesieung has no such special behaviour, note also that yesieung should have a clear stem on top)
- at most four vowel letters, out of which
  - the first ones are at most two horizontal vowel letters (displayed below each other, as well as below any lead consonants), followed by
  - at most two vertical vowel letters (displayed to the left of the preceding vowel letters as well as to the left of lead consonants), followed by
  - at most two horizontal vowel letters (displayed below all the preceding vowel letters), however,
  - note that araea is counted as a horizontal vowel, except after another araea
- at most four trail consonant letters
  - all displayed below all preceeding characters in the syllable, however,
  - note that a ieung is displayed below the preceding trail letter or doubled trail letter (yesieung has no such special behaviour, note also that yesieung should have a clear stem on top)

Other combinations, mostly too many letters of the same kind together, should lead to blotted out display (at least partially, since that will almost be the effect anyway of having too many (shrunk) letter glyphs in too small space. Note that each Hangul jamo may need up to (depending on font) about 15 or so glyph variants, to fit in different positions in a syllable block.

Note that these additional requirements are practical to implement only for spoof prevention alternative 4 above. Though of course possible for the other alternatives too, it then becomes far too unwieldy complicated and unreliable in implementation. Recall also that the letters of a Hangul syllable must be displayed in a single syllable block. Otherwise there is another possibility for spoofing. This speaks strongly for alternative 4.

# 6 Acknowledgements

Many thanks to Jungshik Shin for untangling my misunderstandings about Hangul, and for comments on earlier drafts of this Unicode technical note. Dae Hyuk Ahn, Ienup Sung, and Mark Davis have also provided me with comments on Hangul decomposition. Any remaining errors in the above text are of course mine.

# 7 References

[1] Han'gŭl matchumpŏp (The hangŭl spelling conventions), Ministry of Education, Mun'gyobu, Seoul, 1988. (Reference from Jungshik Shin.)

[2] Han'gŭl match'umpŏp t'ongi'iran (A proposition for unified han'gŭl spelling conventions), Chosŏnŏ Hakhoe (Korean Language Association), Seoul, 1933. (Reference from Jungshik Shin. There may possibly be different versions of this document.)

[3] Kaejŏnghan Chosŏnmal kyubŏmjip (A revised collection of Korean language norms), Kugŏ sajŏng wiwonhoe (Korean Language Assessment Committee), Sahoegwahak Ch'ulp'ansa, P'yŏng'yang, 1988. (Reference from Jungshik Shin.)

[4] The Korean Alphabet, its history and structure, ed. Young-Key Kim-Renaud, University of Hawai'i Press, 1997, ISBN 0-824-81989-6.

[5] The Korean Language, Ho-Min Sohn, Cambridge University Press, 1999, ISBN 0-521-36123-0 or 0-521-36943-6. (Section 6.3 gives a translation to English of the 1444 design document for the Hangul alphabet.)

[6] Hangul, Korea background series, Korean overseas information service, Seoul, Korea, 1973.

[7] Ordering Rules for Hangul – CTT suggestion, Kent Karlsson, 2004, ISO/IEC JTC1/SC2/WG2, N2715, 2004-03-09, http://std.dkuug.dk/JTC1/SC2/WG2/docs/n2715.doc.

[8] The Korean alphabet of 1446 – Expositions, OPA, The visible speech sounds, Annotated translation, Future applicability; Hwun Min Ceng Um, Sek Yen Kim-Cho, Humanity Books and AC Press, New York, 2002, ISBN 89-428-1587-1. (Reproduces, translates and analyses (in English) the 1446 official design document for the Hangul alphabet.)

[9] A history of Korean Alphabet and Movable Types, Ministry of Culture and Information, Republic of Korea, 1970. (Part 1 reproduces the 1444 official design document for the Hangul alphabet.)

[10] The Korean Language, Structure, use and context, Jae Jung Song, 2005, ISBN 0-415-32802-0.