L2/03-203

The arithmetically specified decompositions of precomposed Hangul syllables

Kent Karlsson
<kentk@cs.chalmers.se>, <kentk@chl.chalmers.se>

2003-06-12

1 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 arithmetically specified canonical decomposition into two (*choseong*, *jungseong*) or three (*choseong*, *jungseong*, *jongseong*) Hangul jamo characters in the ranges 1100-1112, 1161-1175, and 11A8-11C2 respectively.

2 Current arithmetically specified decompositions (and compositions)

The current (Unicode 4.0 and Unicode 3.0) arithmetically specified canonical decompositions are described in the Unicode 3.0 and 4.0 books. It directly decomposes precomposed Hangul syllable characters into two or three Hangul Jamo characters. This differs from all other decompositions not only in being arithmetically specified, but also in that all other decompositions are to one or two other characters, never three or more.

3 Proposed arithmetically specified decompositions

The new arithmetically specified canonical decomposition mappings proposed here *do **not** change the full canonical decomposition* for any precomposed Hangul syllable (and they do not change any of the maximal compositions, as used by the normalisation algorithm). So there is no change in the normal forms as a result of this change. There *is* a nominal change in the decomposition steps performed, but no implementation of Unicode normalisation need be changed for this. However, if this proposal is accepted, APIs or UIs presenting one-step decompositions should be changed to reflect the change in one-step decompositions for precomposed Hangul syllable characters.

The proposed formulation have just decompositions into two characters, never three characters. This formulation thus fits much better with the general approach to canonical decomposition as used in Unicode, and fits better with how composition works in UAX 15 (Unicode normalisation algorithm). Thus the arithmetically specified decompositions are not so much of a special case that separate caveats need be kept in the description in UAX 15 regarding Hangul syllable composition. However, the arithmetically specified compositions should be done after the tabularly specified ones. This is not strictly needed for the standard normal forms, but would be necessary if additional (canonical) decompositions are given for multi-letter Hangul jamo conjoining characters (which could be possible in a tailored normalisation).

First, some definitions (essentially same as the ones used for the present arithmetically specified decompositions, some name fixes for clarity have been done).

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 suggested arithmetically specified decompositions 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)

4 Proposed arithmetically specified compositions

The proposed arithmetically specified decompositions imply (proposed) arithmetically specified compositions for **L** and **V**:

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

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

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

These compositions can be used in the specifications of NFC and NFKC. They can also be used in actual implementations, though less efficient than composing three jamo in one go. However, implementations of normalisation are free to use the direct composition of three (suitable) jamos to a precomposed Hangul syllable, since none of the normal forms are affected by the suggested change of decomposition mappings.

5 Proposed changes to UAX 15

Since this change is “post 4.0”, the best place to put it (until next version) is in UAX 15, as a new annex on arithmetically specified decompositions.

Further, UAX 15 says:

> “**D4.** A character X can be *primary combined* with a character Y if and only if there is a primary composite Z which is canonically equivalent to the sequence <X, Y>.”

While subtly formulated to cover also the case of LVT Hangul syllable characters which, apart from singleton decomposition mappings, currently have a decomposition mapping into other than two other characters. Singleton decompositions are not used for composition. However, with the change in decomposition mappings suggested above, the formulation for “D4” can be reformulated into a more direct one:

> “**D4.** A character X can be *primary combined* with a character Y if and only if there is a primary composite Z which has a decomposition mapping to the sequence <X, Y>.”

Further, the following formulation for D3 in UAX 15:

> "**D3.** A *primary composite* is a character that has a canonical decomposition mapping in the Unicode Character Database (or has a canonical Hangul decomposition) but is not in the §6 Composition Exclusion Table."

should, independently of the rest of this proposal, be reformulated as:

> "**D3.** A *primary composite* is a character that has a canonical decomposition mapping in the Unicode Character Database or has an arithmetically specified (by Unicode, see annex X below) canonical decomposition, but is not in the §6 Composition Exclusion Table."

Finally, in annex 10 of UAX 15, the text there should be replaced by the following text. One could also include code for the "hangulSyllableType" function and the "HangulSyllableType" class. Note that the current code (as the suggested one) already uses the arithmetic compositions suggested above. Note also that the current code for arithmetic decompositions can be seen as an optimisation of the suggested decompositions.

Common Constants

```
static final int
    SBase = 0xAC00, LBase = 0x1100, VBase = 0x1161, TBaseM1 = 0x11A7,
    VCount = 21, TCountP1 = 28,
    NCount = VCount * TCountP1;   // 588
```

Hangul Arithmetic Decomposition

```
public static String decomposeHangul(int c)
{
    // This arithmetic can be optimised. Compare code for getHangulName.

    if (hangulSyllableType(c) == HangulSyllableType.LVT)
    {
        int lv = SBase   + (((c  – SBase) / NCount) * NCount);
        int t  = TBaseM1 +  ((c  – SBase) % TCountP1);

        int l  = LBase   +  ((lv – SBase) / NCount);
        int v  = VBase   + (((lv – SBase) % NCount) / TCountP1);

        return CharacterToString(l)+
               CharacterToString(v)+
               CharacterToString(t);
    }
    else if (hangulSyllableType(c) == HangulSyllableType.LV)
    {
        int l = LBase +  ((c – SBase) / NCount);
        int v = VBase + (((c – SBase) % NCount) / TCountP1);

        return CharacterToString(l)+
               CharacterToString(v);
    }
    else
    {
```

```
        return CharacterToString(c);
    }
}
```

Arithmetic decompositions must be done before tabular decompositions (in case a tailoring has tabular decompositions for Hangul Jamo, or this is used as a part of a compatibility decomposition for pre-3.0 Unicode data, which had compatibility decompositions for several Hangul Jamo characters).

Hangul Arithmetic Composition

```
public static String composeHangul(String source)
{
    int len = source.length();
    if (len == 0)
        return "";


    StringBuffer result = new StringBuffer();

    // Hangul is in the BMP, so we need not worry about higher planes.

    char prev = source.charAt(0);            // get first char

    for (int i = 1; i < len; i++)
    {
        char curr = source.charAt(i);

        if ('\u1100' <= prev && prev <= '\u1112' && // "modern" L
            '\u1161' <= curr && curr <= '\u1175')   // "modern" V
        {
            // make syllable of form LV
            prev = (char)(SBase + ((prev–LBase) * NCount) +
                                  ((curr–VBase) * TCountP1));
        }
        else if (hangulSyllableType(prev) == HangulSyllableType.LV &&
                 '\u11A8' <= curr && curr <= '\u11C2') // "modern" T
        {
            // make syllable of form LVT
            prev += curr – TBaseM1;
        }
        else
        {
            // no arithmetic composition possible, move on
            result.append(prev);
            prev = curr;
        }
    }
    result.append(prev);  // don't loose last char in string
    return result.toString();
}
```

Arithmetic decompositions must be done after tabular compositions (in case a tailoring has tabular compositions for Hangul Jamo).

Hangul Character Names

Hangul arithmetic decomposition is also used to form the character names for the Hangul syllables. While the sample code that illustrates this process is not directly related to normalization, it is worth including because it is so similar to the decomposition code.

```
static private String[] JAMO_L_TABLE =
{
    "G", "GG", "N", "D", "DD", "R", "M", "B", "BB",
    "S", "SS", "",  "J", "JJ", "C", "K", "T", "P", "H"
};

static private String[] JAMO_V_TABLE =
{
    "A",  "AE",  "YA", "YAE", "EO", "E",   "YEO", "YE", "O",
    "WA", "WAE", "OE", "YO",  "U",  "WEO", "WE",  "WI",
    "YU", "EU",  "YI", "I"
};

static private String[] JAMO_T_TABLE =
{
    "",   "G",  "GG", "GS", "N",  "NJ", "NH", "D", "L", "LG",
    "LM", "LB", "LS", "LT", "LP", "LH", "M",  "B", "BS",
    "S",  "SS", "NG", "J",  "C",  "K",  "T",  "P", "H"
};

public static String getHangulName(char c)
{
    if (hangulSyllableType(c) == HangulSyllableType.LV ||
        hangulSyllableType(c) == HangulSyllableType.LVT)
    {
        // This code uses arithmetically optimised "decomposition".
        // (The test above hasn't been optimised, though.)
        // Note the empty string at index 0 of JAMO_T_TABLE.

        int SIndex = c - SBase;
        int LIndex =  SIndex / NCount;
        int VIndex = (SIndex % NCount) / TCountP1;
        int TIndex =  SIndex % TCountP1;

        return "HANGUL SYLLABLE "+
                 JAMO_L_TABLE[LIndex]+
                 JAMO_V_TABLE[VIndex]+
                 JAMO_T_TABLE[TIndex];
    }
    else
    {
        throw new IllegalArgumentException("Not a Hangul Syllable: "+s);
    }
}
```