Annex Q [temporary ‘number’]
(informative)

Order-preserving subkey reduction

The template table of collation weights has four levels. When applied to a string, each level nominally produces a subkey that is about as long, in number of weights, as the number of characters in the string itself. There are, however, a number of ways to reduce the size of the subkeys without changing the ordering as determined by the nominal key. This international standard neither specifies normatively, and even less requires the use of, any subkey reduction technique. However, for conformity, any key size reduction method must preserve the order between strings as determined by the nominal key produced by the selected tailoring of the template table.

For illustration of what can be done in terms of subkey reduction, we here present two example reduction methods. Good implementations of these methods produce the reduced key directly, without producing the nominal key first.

Applied correctly to each nominal key, these reduction methods keep the order between the strings, since each is a strictly monotonically increasing mapping, i.e., if *nominal_key1*<*nominal_key2*, then reduce1(*nominal_key1*)<reduce1(*nominal_key2*), and reduce2(*nominal_key1*)<reduce2(*nominal_key2*).

Each subkey reduction method that results in a strictly monotonically increasing mapping can be applied to any level. And different reduction methods can be applied to different levels, as long as it is done consistently for all keys. E.g., example method 1 below can be used for levels 2 and 4, while using example method 2 below for level 3, while no reduction method need be applied to level 1.

1 Example key reduction method 1: interleaved counts and weights

This method can be applied for a single selected weight at each level this method is used, preferably a weight that is very commonly used at that level.

This method uses a count value, which may be stored in the subkey using different number of digits from what the weights uses at that level. The transformed value (see below) of the count need have no particular relation to any of the weight values. In this illustration we will use integer values between 00 and FF (in hexadecimal), the latter is the $maxcnt_{lvl}$. A practical implementation may of course use a wider range of values.

The reduction method works as follows, described in principle, not in implementation terms: Each maximal subrun of the for that level selected weight, including the subruns of length 0, is replaced by a transformed count value, as follows:

- swa_{lvl} is the selected reduction weight at level lvl.
- *wb* is the weight (including subkey separator (1) and key terminator (0)) that follows the (empty or not) maximal subrun of swa_{lvl} currently processed.
- *cnt* is the length (≥ 0) of the subrun currently processed.
- $maxcnt_{\mathrm{lvl}}$ is a value ≥ 0 and and is the largest value that can be stored as a count in the subkey, given the number of digits used for the storage of a (transformed) count in a subkey at that level.
- $guardcnt_{\mathrm{lvl}}$ is a value ≥ 0 and $\leq maxcnt_{\mathrm{lvl}}$, e.g. $maxcnt_{\mathrm{lvl}}$ **div** 2.
- $lowcntsize_{\mathrm{lvl}} = guardcnt_{\mathrm{lvl}} + 1$.
- $highcntsize_{\mathrm{lvl}} = maxcnt_{\mathrm{lvl}} - guardcnt_{\mathrm{lvl}} + 1$.
- If $swa_{\mathrm{lvl}} > wb$:

  Replace the subrun with the count/weight-pair ($guardcnt_{\mathrm{lvl}}$, swa_{lvl}) repeated (*cnt* **div** $lowcntsize_{\mathrm{lvl}}$) times, followed by the count (*cnt* **mod** $lowcntsize_{\mathrm{lvl}}$).
- If $swa_{\mathrm{lvl}} < wb$:

  Replace the subrun with the count/weight-pair ($guardcnt_{\mathrm{lvl}}$, swa_{lvl}) repeated (*cnt* **div** $highcntsize_{\mathrm{lvl}}$) times, followed by the count ($maxcnt_{\mathrm{lvl}}$ − (*cnt* **mod** $highcntsize_{\mathrm{lvl}}$)).

Note that if the limits are set reasonably high, the count/weight-pairs will be repeated zero times in practical cases. In the other extreme, if $maxcnt_{\mathrm{lvl}}$ is zero, the subkey is not changed.

The reason this method works only for one weight per level is that empty subruns otherwise would cause ambiguous key reductions.

This reduction method can be applied to level 2, where <BASE> can be expected to be a very common weight. It can also be used for level 3, where <MIN> (minuscule, i.e. lower-case, or case-less) can be expected to be very common. This reduction method can also be applied to level 4, where <PLAIN> is likely to be very common on level 4. Note that if the selected weight is not very common in a string, the resulting key by this method may be longer than the nominal key, since empty subruns of the selected weight must be replaced by another subrun that is non-empty.

An example reduction using this method on level 2 (selecting the weight <BASE>), on level 3 (selecting the weight <MIN>), and on level 4 (selecting the weight <PLAIN>). For readability, we will in this example write <a>, <b>, etc. for the weights of the letters, <-> for <BLANK>, <l> for <MIN>, <u> for <CAP>, <*> for <PLAIN>, <A> for <ACUTE>, <H> for the level 4 weight of a hyphen, and <P> for the level 4 weight of an apostrophe. The example character string is “Vice-Président’s”. The nominal key, according to the template table, is (as a number in R, expressed as a hexadecimal fractional number; <v> etc. really stands for digit sequences; spaces are used for alignment for clarity, they are in no way part the actual key):

```
0.<v><i><c><e>   <p><r><e>   <s><i><d><e><n><t>   <s> 0001
  <-><-><-><->   <-><-><-><A><-><-><-><-><-><->   <-> 0001
  <u><l><l><l>   <u><l><l>   <l><l><l><l><l><l>   <l> 0001
  <*><*><*><*><H><*><*><*>   <*><*><*><*><*><*><P><*> 0000
```

Do the reduction as described above (note that here: <-> (<l>) is smaller than any other level 2 (3) weight, but greater than 1 (the subkey separator weight), and <*> is greater than any other level 4 weight):

```
0.<v><i><c><e>   <p><r><e>   <s><i><d><e><n><t>   <s>   0001
  F8                      <A> 07                          0001
  FF <u>      FC <u> 0B                                   0001
  04         <H> 09                            <P> 01   0000
```

Note that “count” values are at the beginning and end of the subkeys, as well as between each non-selected weight. This key is significantly shorter than the nominal key, in this example as well as for most (not all) other strings that normally occurs in a collation.

2 Example key reduction method 2: each count integrated in a weight

This method can be applied for a set of weights at each level, preferably ones that are fairly commonly used at that level.

This method also uses a count value, but the transformed weight value must have the same number of digits as all other weights at that level uses. The transformed values (see below) for the count must have values that are in the neighborhood of the weight in the subrun that is replaced. This neighborhood of a weight must not overlap with any other weights or neighborhoods of weights.

The reduction method works as follows, described in principle, not in implementation terms: Each non-empty maximal subrun of a weight selected for reduction is replaced as follows:

- *wa* is the weight in the subrun.
- *wb* is the weight (including subkey separator (1) and key terminator (0)) that follows the non-empty maximal subrun of *wa* currently processed.
- *cnt* is the length (≥ 1) of the subrun currently processed.
- $minnbh_{wa}$ is the minimum value that constitutes the neighborhood of *wa*.
- $maxnbh_{wa}$ is the maximum value that constitutes the neighborhood of *wa*.
- $lownbhsize_{wa} = wa - minnbh_{wa} + 1$.
- $highnbhsize_{wa} = maxnbh_{wa} - wa + 1$.
- If $wa > wb$:
  Replace the subrun with the weight *wa* repeated (*cnt* **div** $lownbhsize_{wa}$) times, and, if (*cnt* **mod** $lownbhsize_{wa}$) is non-zero, followed by the weight ($minnbh_{wa}$ + (*cnt* **mod** $lownbhsize_{wa}$) − 1).
- If $wa < wb$:
  Replace the subrun with the weight *wa* repeated (*cnt* **div** $highnbhsize_{wa}$) times, and, if (*cnt* **mod** $highnbhsize_{wa}$) is non-zero, followed by the weight ($maxnbh_{wa}$ − (*cnt* **mod** $highnbhsize_{wa}$) + 1).

Note that if the neighborhood of *wa* consists of *wa* only, the subrun is not changed.

This reduction method can be applied to level 2, for <BASE>. It can also be used for level 3, e.g. for <MIN>, <CAP>, <HIRA>, and <KATA>. It can also be applied to level 4, for <PLAIN>. Note that even if the weight reduced is not common, the resulting subkey is never longer than the nominal subkey. A nontrivial neighborhood is needed around each of the selected weights for a shortening of the subkey to actually take place.

We do an example reduction using this method on level 2 (for <BASE>), on level 3 (for <MIN> and <CAP>), and on level 4 (for <PLAIN>). For this we make the following assumptions:

- <-> (level 2): value of weight: 0026; minimum of neighborhood: 0022; maximum of neighborhood: 002A.
- <l> (level 3): value of weight: 0005; minimum of neighborhood: 0002, maximum of neighborhood: 0007.
- <u> (level 3): value of weight: 0017; minimum of neighborhood: 0015, maximum of neighborhood: 001A.
- <*> (level 4): value of weight: 0F80; minimum of neighborhood: 0F00, maximum of neighborhood: 0FFE.

For the same example string as in the description of example reduction method 1, we still have the nominal key:

```
0.<v><i><c><e>   <p><r><e>   <s><i><d><e><n><t>   <s>   0001
  <-><-><-><->   <-><-><-><A><-><-><-><-><-><->   <->   0001
  <u><l><l><l>   <u><l><l>   <l><l><l><l><l><l>   <l>   0001
  <*><*><*><*><H><*><*><*>   <*><*><*><*><*><*><P><*>   0000
```

Do the reduction as described above:

```
0.<v><i><c><e>   <p><r><e>   <s><i><d><e><n><t>   <s>   0001
  0024                    <A>0028                       0001
  0015  0005     0015  0005 0005 0002                   0001
  0F03        <H>0F08                          <P>0F00  0000
```

If the lower neighborhood of <l> had been a bit bigger, we could have used only one weight (or two), instead of three, for the sequence of nine <l> weights.

This key is also significantly shorter than the nominal key, and this method never lengthens the key, since every subrun replaced is replaced by one that is shorter or at most as long as the original subrun. This method can also be applied to several weights at each level, which example method 1 cannot. On the other hand, example method 2 is a bit more complex than example method 1, since it needs to keep track of non-overlapping neighborhoods around some of the weights.