The 'end of word' problem in Sanskrit: report of the workgroup

The workgroup was established to consider the problem that in Sanskrit word junctions are often obscured or eliminated, and to propose XML/TEI markup to address this problem.

For the convenience of non-Sanskritists the discussion here uses Roman transliteration, but the difficulty arises within the Devanāgarī script in which Sanskrit is normally written. Devanāgarī is a syllabary, in which one syllable consists of any number of consonants (in practice, from zero to five) followed by one vowel followed optionally by *ṃ* or *ḥ* (*anusvāra* or *visarga*).[1] If a word ends in a consonant, it therefore has to share a syllable with the next word, so that *āsīd rājā* ('there was a king') is written **ā - sī - drā - jā**. To make matters worse, sandhi (phonological change at word boundaries) may fuse two consecutive vowels together, so that, even ignoring orthography, the words can no longer be divided — for example *tathā api* ('even so') becomes *tathāpi*, where the single vowel *ā* is shared by two inseparable words.

This presents no problems for display, but it poses very serious problems for any kind of analysis, and it has long been clear that we need to find some way round it. Various ad hoc 'solutions' have been used by people who have typed up Sanskrit texts, but none is at all satisfactory. It is clear that the most appropriate solution would be a form of specialised markup — hence XML and the Text Encoding Initiative.

The Sanskrit 'end of word' problem is in fact merely one language-specific form of a more general issue: that in many languages it is possible for a 'chunk' of text to represent a sequence of isolable segments, where those segments do not appear in their normal identifiable form. Examples include:

- a contraction (Old High German, English, many other languages)
- a compound word (Sanskrit, Avestan, many other languages)
- a group of words whose forms have been affected by 'euphonic' sandhi changes (Sanskrit, Breton)
- a group of words in which, for orthographic or other reasons, the word junctions are not indicated (Sanskrit, Japanese)

The workgroup accept the TEI Council's desire to avoid adding to the existing stock of elements if at all possible, and they propose that such cases should be dealt with by segmentation of the text with the existing element **<seg>**, using the new element **<choice>** to handle the implicit parallelism between actual text and proposed analysis. Thus, to take a simple example, they propose that contractions occurring sporadically in a text should be handled as follows:

```
We're all
<choice>
  <seg type="contraction">
    gonna
  </seg>
  <seg type="analysis">
    <seg>going</seg>
    <seg>to</seg>
  </seg>
</choice>
die!
```

For serious analysis of Sanskrit, however, it is necessary to incorporate into the markup an indication of the degree of thoroughness to which the analysis aspires. This is

[1] This is a slight simplification, deliberately omitting certain 'optional' features such as Vedic accents. These do not, however, affect the principle as stated here.

because many Romanised versions of Sanskrit texts already exist, and it will be simple to convert these automatically to Devanāgarī with partial markup built in: Roman *āsīd rājā* contains enough information to mark up the word junction that vanishes in Devanāgarī आसीद्राजा. The workgroup propose that three 'levels' of markup should be distinguished:

- **Level 1** is restricted to issues of orthography. Specifically, this means that markup is applied only to (a) non-final words ending in consonants (excluding anusvāra *ṃ* and visarga *ḥ*), and (b) sequences such as *rāmo 'pi*, normally written in Devanāgarī without spaces as रामोऽपि. Level 1 markup can be automatically created for texts originally encoded in Roman transliteration.
- **Level 2** includes all cases covered by level 1, but additionally applies markup to vowel sandhis of the type 'vowel + vowel ⇒ vowel'. Level 2 markup results in full separability of all the words of a text.
- **Level 3** involves complete analysis of all sandhis. This is not necessary if the aim is simply separation of words, but might be found useful in many cases. For example, the word *rāmaḥ* (nominative singular of the name of the god) may assume all of the following sandhi forms: *rāmaḥ*, *rāmaś*, *rāmaṣ*, *rāmas*, *rāma*, *rāmo*. In an index verborum it might be thought more sensible to include all these under the single form *rāmaḥ*; if that form can be automatically extracted from XML markup, so much the better.

Since it cannot be guaranteed that all parts of a text will be marked up to the same level, the level of each individual junction needs to be indicated. This can best be done by using the **type** attribute of the innermost **<seg>** elements, each of which except the last should specify **type="level1"**, **type="level2"** or **type="level3"**:

```
<choice>
  <seg type="wordgroup">
    sarvametadyathāvedya
  </seg>
  <seg type="analysis">
    <seg type="level1">sarvam</seg>
    <seg type="level3">etat</seg>
    <seg type="level2">yathā</seg>
    <seg>āvedya</seg>
  </seg>
</choice>
```

A mechanism also has to be defined to permit markup at more than one level to coexist for a single **<seg>** element, since this could well result from the 'upgrading' of a text. This is accomplished by bracketing the alternatives with **<choice>**…**</choice>**:

```
<choice>
  <seg type="wordgroup">
    sarvametadyathāvedya
  </seg>
  <seg type="analysis">
    <seg type="level1">sarvam</seg>
    <choice>
      <seg type="level1">etad</seg>
      <seg type="level3">etat</seg>
    </choice>
    <seg type="level2">yathā</seg>
    <seg>āvedya</seg>
  </seg>
</choice>
```

Compounds can be dealt with in an exactly analogous way:

```
<choice>
  <seg type="compound">
    sarvavidvajjanāpriyam
  </seg>
  <seg type="analysis">
    <seg type="level1">sarva</seg>
    <choice>
      <seg type="level1">vidvaj</seg>
      <seg type="level3">vidvat</seg>
    </choice>
    <seg type="level2">jana</seg>
    <seg>apriyam</seg>
  </seg>
</choice>
```

Conclusions

To sum up our recommendations, we propose

A A list of attribute values, to be called **junctionType**. These values for the **"type="** attribute of the **<seg>** tag are relevant for all languages, and consist of **"contraction"**, **"compound"**, **"wordgroup"** and **"analysis"**. The values indicate what kind of junction is to be analysed, or — in the case of **"analysis"** — that the respective **<seg>** element contains that analysis.

B A second list, to be called **"sanskritAnalysis"**, containing the values **"level1"**, **"level2"** and **"level3"**. This list specifies the levels of analysis used for Sanskrit.

Since our proposals involve no new elements, merely recommending patterns of usage for existing attributes, it follows that any document in which they are correctly used will conform to the current TEI release (P4). In the forthcoming release (P5), however, it is intended that the use of one or more schema languages will replace the current reliance on DTDs, making it possible to specify such usages formally. We therefore propose that list (A) should be included in the P5 version of the TEI Guidelines as the language-general set of attribute values for the analysis of junctions, and list (B) as an example of how specialised attribute values may be used to create language-specific markup for junctions. List (B) would then encourage other users, for instance Japanologists, to create their own language-specific values for **"type="**.

The two lists could be contained within two RELAX NG named patterns called **"junctionType"** and **"sanskritAnalysis"**. These patterns would form part of a general named pattern called **"junctions"**. Below we give a small RELAX NG schema which shows how the patterns could be used as a part of the declaration of the **<seg>** element. The schema allows for the validation of the examples which appear in this report:

```
start = choice
choice = element choice { seg+ }
seg = element seg { junction?, (text | choice | seg)+ }
junction = junctionType | sanskritAnalysis
# Users who are interested in other languages than Sanskrit are
# encouraged to integrate their own patterns into the 'junction'
# pattern, e.g.
# junction = junctionType | sanskritAnalysis | japaneseAnalysis | ...
junctionType = attribute type { "contraction" | "compound" | "wordgroup" |
    "analysis" }
sanskritAnalysis = attribute type { "level1" | "level2" | "level3" }
```

We request the TEI council (assuming that they approve our proposals) to incorporate some such schema in an optional module of P5, and to use examples such as those contained in this report to document the use of that module. Users who are interested in junction markup should be encouraged to rely on this module as a part of their customisation of the general TEI annotation scheme.

John D. Smith
November 15, 2004