Design of a Sanskrit Reader Assistant

Gérard Huet Inria Paris

**Abstract**

We give the design rationale of a semi-automatic computer tool that assists its users in understanding classical Sanskrit text, without presupposing an extensive knowledge of the grammar of the language. The Sanskrit Heritage Web service comprises three main components: a hypertext dictionary, a morphology generator, and an interactive segmenter cum tagger using a graphical interface. We describe the interaction between these components, and the main logical invariants that specify their behavior. We discuss possible extensions of this work towards semantic analysis, corpus management, and teaching. Finally, the main short-comings (incompleteness, overgeneration) are addressed and their possible solutions discussed.

# Background

According to historical linguistics, languages evolve with time and space, and thus a given language is defined as the set of attested sentences in a specific corpus, in a given span of time over a given geographical area. Grammars are only secondary artefacts, and the set of sentences they sanction must be measured against its Gold Standard corpus. This view of historical linguistics was actually fostered by the Western discovery of Sanskrit, and the ensuing development of the Indo-European comparative linguistics. In the 20th century, universal linguistics was investigated, and formal generative grammars were designed as computational artefacts, such as context-free grammars. One thus speaks of the language associated with a grammar, as the set of sentences potentially generated by the formal mechanism. With this new view, one may design parsers, that will take a candidate sentence, and attempt to construct a formal structure that represents its linguistics analysis. Still more recently, deep neural networks have been used to train a parser from a data-base of annotated corpus, leading to recognizers that actually attempt to synthesize a grammar from the learning dataset.

With the advent of computers, a computational view of written text emerged.

Digital libraries of literature were developed, and the TEI (Text encoding initiative) produced standards for structured representation of linguistic data. Then software tools were developed for comparison of versions

of texts and computer assistants used by philologists for the establishment of critical editions. On the computational side, algebraic methods were investigated as a foundation of the emerging new discipline of Informatics. The theory of formal languages and automata developed formalisms such as regular and context-free languages, and algorithms permitting recognition of formal text and inter- translation. These methods were then extended to the recognition of written natural language utterances and their semantic analysis, paving the way for computer inter-language translation. This became a major subfield of Informatics, namely Computational linguistics. In complement, specific techniques were studied for speech recognition and handwriting recognition.

Sanskrit computational linguistics started developing around 2000. Sanskrit is considered by comparative linguists as an Old-Indo-Aryan language, in the Indo-European family, descending from Vedic Sanskrit. Other descendants are Prakrits, the ancient vernacular languages of Northern India, which themselves evolved into modern North-Indian languages. But Classical Sanskrit has an important specificity. It is actually a semi-formal language, since it has been described in great details by a formal grammatical treatise, the *Aṣṭādhyāyī* of Pāṇini, as early as the 5th century B.C. Indeed, *saṃskṛta* means *refined* or *perfected*, in contrast with *prakṛta* which means *natural*. Since this descriptive grammar is very complete and accurate, it became *de facto* prescriptive, and thus the language stopped its evolution, which essentially remained within the bounds of Pāṇini’s grammar.

The *Aṣṭādhyāyī* is often characterized as a *generative* grammar. Indeed, its main part gives a generative view of morphology, through procedures of elaboration [*vṛtti*] of word forms from roots [*dhātu*]: inflected words [*pāda*] are obtained by declension [*sup*] from nominal stems [*prātipadika*] and by conjugation [*tiṅ*] from roots and their auxiliary modes stems [*sanādi*]. Nominal stems themselves are issued from roots in two recursive layers: primary stems are built from roots by *kṛt* suffixing, secondary stems are built from stems by *taddhita* suffixing. Finally, a very productive process of compounding [*samāsa*] contracts contiguous words into a single one. On the other hand, another section of *Aṣṭādhyāyī*, dealing with sentences [*vākya*], is analytical in character: it defines sensible enunciations as the ones consistent with semantic roles of words [*kāraka*], in a model of situations constrained by functional dependencies [*ākāṅkṣa*]. This rich linguistic tradition is an indispensable foundation of Sanskrit computational linguistics.

# The Sanskrit Heritage Dictionary

Our experiments with Sanskrit computational linguistics started with the de- sign of an electronic Sanskrit to French dictionary called the Sanskrit Heritage Dictionary (Dictionnaire de l’Héritage du Sanskrit). This hypertext dictionary, which has been publicly available on Internet since 2004 through its index at [sanskrit.inria.fr/DICO/index.fr.html](https://sanskrit.inria.fr/DICO/index.fr.html), has a dual purpose. Initially meant for human users as a small hypertext encyclopedia of Indian traditional culture, it evolved into a linguistic digital database

proper for computer processing of the language. Its overall design has been documented in [[3](#_bookmark6), [4](#_bookmark7), [8](#_bookmark11), [13](#_bookmark16)]. From the point of view of the Indian grammatical tradition, it plays the role of a database of roots [*dhātupāṭha*], of non-derivable nominal stems [*uṇādikośa*], and of semantic glosses, comparable to traditional thesauri such as the *Amarakoṣa*.

This dictionary is thus a dual resource: as the source of morphology generation by computer, and as a glossary appropriate for a human user to articulate meanings from formal analysis, taking into account usage, literature, and more generally cultural artifacts of ancient India. Thus, it is essential to record not just the etymological meaning of words [*abhidhā, mukhya*], but also their idiomatic usage [*rūḍha*], their figurative usage [*lakṣaṇā*], their metaphorical usage [*gauṇa*], their suggestive usage [*vyañjana*], their implicature [*dhvani*]. The dictionary also acts as a repertory of frozen expressions, illustrating frequent collocations. A number of citations is also provided, illustrating typical styles with a selection of important sayings [*subhāṣita*] and ritual invocations [*mantra*]. They currently number 998. Among them, 983 are given with their analysis, as a link to the Heritage\_citations branch of the tagged corpus repository described in section 7 below.

The source of the dictionary is a structured text file, which is parsed into a set of lexical entries tagged with parameters, in order to be further processed by the morphological generator described in the next section. Currently the dictionary has 35245 entries, over 658 verbal roots.

# Morphology generation

## Generalities

This essential component is a *pada* generator, inspired from the Pāṇinian notion. It allows generation of forms from lexicalized stems, both for substantives and verbs. It basically follows the methodology sketched above, but with a few discrepancies that it might be useful to explain.

First of all, we must understand the phonemic encoding of speech. Speech [*vāc*] is a sound signal, a vocal enunciation. Articulated speech or language [*śabda*] is a meaningful linguistic utterance, pairing a signifier [*dhvani*] and a signified [*artha*], its meaning. Phonetics [*śikṣā*] was studied in India from high antiquity, and recognized as a technical area of knowledge [*vedāṅga*]. By the time of Pāṇini, Vedic texts had been subjected to lexical analysis [*padapāṭha*] in treatises [*prātiśākhya*] expressing speech in abstract discretized form. The 50 discrete units of speech [*varṇa*] correspond roughly to phonemes (vowels being further distinguished by duration). These units form syllables [*akṣara*] which themselves compose morphemes, words, and discourse amenable to representation in writing with syllabic scripts such as *devanāgarī*.

Pāṇini describes linguistic operations with strings of symbols, mixing genuine phonemes belonging to the phonetic production, and formal markers [*anubandha*] that represent operations on these strings. Morphological synthesis operate

on phonemic forms [*prakṛti*] via affixation of symbolic suffixes [*pratyāya*] that mix genuine phonemic production with *anubandha* operators. It is this interveaving of object language (Sanskrit) with meta language (the command language of the Pāṇinian operations controled by *anubandha*s) that is the main difficulty in understanding the Pāninian *sūtra*s.

For instance, one may form from root *kṛ* (to do) an agent noun of its action according to *sūtra* {III,1,133}, that prescribes use of primary suffix *ṇvul*, consisting solely of *anubandha*s. Marker *ṇ* specifies bringing the root stem to its highest vocalic degree [*vṛddhi*] by a further rule that will produce morpheme *kār*. Marker *vu* is defined as abbreviating the morpheme *aka* according to *sūtra* {VII,1,1}, which is thus affixed to the base *kār* to produce *kāraka* “he who does”. Marker *l* specifies the initial position of the accent.

Markers are also used to classify phonemes in the various classes that are relevant to the grammar operations: vowels, consonants, nasals, sibilants, etc. The whole grammar is introduced by a set of 14 *sūtra*s of a specific form, con- sisting each of a sequence of *varṇa*s ended with a marker. This permits to define *condensed statements* [*pratyāhāra*] formed with one *varṇa* and one marker, and denoting the set of all phonemes starting with the given *varṇa* up to the marker, erasing intermediate markers. Thus, we may designate all consonants by *hal*, all sibilants by *śar*, etc. The same idea is used for e.g. designating all 18 personal ending suffixes for verbs by *tiṅ*. Another important Pāṇinian concept is *recur- ring presuppositions* [*anuvṛtti*] that permits hierarchical sharing of conditioning conditions for *sūtra*s.

These abbreviating notations were essential to make the grammar compact in view of its human memorization. However, they are not essential for com- puter treatment, since the memory requirements are relaxed, and thus more explicit descriptions may be used instead. For instance, in our implementation phonemes are represented by natural numbers, morphemes and words are repre- sented by lists of such, etc. As for *anubandha*s, they are replaced by functional invocation of the corresponding operation. Thus, our implementation is not a strict emulator of *Aṣṭādhyāyī*, and it favors modular compositionality over code compression. Let us list a few more discrepancies.

We slightly extended the *varṇa*s in order to have a finer classification leading to a more uniform treatment. For instance, we have 2 varieties of phoneme *j*, in order to allow for forms such as *mṛṣṭa*, past participle of root *mṛj*, as opposed to regular formations such as *bhakta*, past participle of *bhaj*. Similarly, we have 3 varieties of phoneme *h*, to accommodate respectively *bṛḍha*, *dugdha* and *naddha*, obtained uniformly by alliteration [*sandhi*] operations on this extended phonemic set.

A more significant discrepancy is that we do retroflexion on the fly during morphological synthesis, rather than as a terminal operation effected during the final phonetic smoothing of the *tripādī* section. Another innovation is the notion of phantom phoneme. This mechanism is necessary for the proper treatment of roots starting with *i*, *u*, or their lengthened versions *ī* and *ū*, when they are prefixed by preverb *ā*, in order to allow the proper recognition of sentences like *ihehi* ‘come here’. This mechanism has been explained in [[6](#_bookmark9), [10](#_bookmark13), [1](#_bookmark4)].

What matters is not to follow *Aṣṭādhyāyī*’s computations to the letter, but

to compute stems [*prātipadika*], word forms [*pada*] and other morphemes such as preverbs [*upasarga*] and other particles [*nipāta*], in such a way that at analysis time we may reconstruct from them word forms consistent with the Pāṇinian treatment. An important concern is that the generative process must be finitistic, i.e. must only generate a finite set of forms, which will be used as a filtering database at analysis time. Thus, it imposes for instance that compound words must be broken up in their constituents, which will allow their reconstruction at analysis time, since the grammar does not impose a limit on their number of components. Actually, compounding in Sanskrit is beyond morphology, and interferes with syntax. Compounds are syntactic macros for rigid nominal phrases, as is explicit from their canonical paraphrase [*vigraha*].

The standard construction of compounds glues two nouns [*subanta*] by join- ing the stem [*prātipadika*] of the first one to the second one. The first component is obtained by erasure [*lopa*] of the declension suffix of the first word. Knowledge of this declension suffix is necessary in the Pāṇinian system, because it is used to elaborate the paraphrase [*vigraha*] that defines its meaning as an abbreviation for a specific noun phrase. In effect, *Aṣṭādhyāyī* does not just generate forms, but it generates [*śabda*] in its double aspect of phonetic utterance and meaning. It may be thought of as generating Sanskrit signs from the speaker’s intention [*vivakṣā, tātparya*] [[12](#_bookmark15)]. But when we analyze Sanskrit text, only the phonetic part of the sign is available, and still in an impoverished condition, since accent is missing in writing, whereas it is recursively computed in an explicit manner according to the derivation rules. On the other hand, this means that our generator does not need to make the accent computation, since this would not help us to recognize text, where the information is lost.

## Generative power of morphology

Let us first consider inflexion of nominal stems, irrespectively of whether these stems are lexicalized, or themselves issued from generative processes.

On the side of nominals, we implemented the various declension schemes, regular as well as exceptional, and this led to quite a few distinct paradigms, 116 at the time of writing. But it should be remarked that the retroflexion of phonemes *s* and *n* is automatically accounted for by the context condition of the suffixing operations, and thus we have a unique regular paradigm for stems in

*-a*, generating *rāmeṇa* as well as *devena* – our paradigms are functional phonetic schemes and not merely tables of suffixes.

We profit of this generation pass to engender other morphemes, usable to assemble various constructions. For instance, we store the bare stems themselves, in order to provide the first component of nominal compounds, by elision [*lopa*] of the declension suffix [*sup-pratyaya*] of the prior element [*pūrvapada*]. Thus, production of the inflected forms of *rājan* will populate the bank of Iic stems with the form *rāja*, so that it may be recognized at segmenting time as the proper first component of a word form such as *rājapuruṣaḥ*. At the same time, we generate in another database the first component of the *cvi* inchoative constructions in *-ī* or *-ū*. In this way entry *drava* (liquid) will engender form *dravī*, in order to recognize e.g. *dravībhūtaḥ* (liquefied) used with past participle of the auxiliary root *bhū*

(to become). Also, we engender pseudo-ablative forms of the *tasil* construction, in order to recognize e.g. indeclinable *viśvatas* (from all sides) from pronoun *viśva* (all).

We also pave the way for the recognition of exocentric [*bahuvrīhi*] compounds by providing extra forms for their second component. Thus, neuter substantive *ambara* (cloth, neuter) produces a masculine form *ambaraḥ*, usable only as second element of a compound such as *pītāmbaraḥ* (he who wears a yellow garment). A similar mechanism is implemented to recognize invariable [*avyayī-bhāva*] compounds such as *nirmakṣikam* (exempt from flies) from feminine stem *makṣikā* (fly). Please note that such special forms as *ambaraḥ* and *makṣikam* are not, and should not be recognized as, *bona fide* stand-alone inflected words (*pada*).

Let us now consider the verbal system. Our notion of verbal root is close to that of Whitney [[24](#_bookmark27)]. For instance, we group in one entry *kṛ* the present forms *karoti* and *kṛṇoti*, even though they belong to two different present classes (*gaṇa*), and thus occur in two distinct entries in the Pāṇinian *dhātupāṭha*. The justification is that these *gaṇa*s are just generators for the present system. This allows us not to get artificial ambiguities with forms outside of the present system (future, perfect, passive, etc.) As a consequence, we have far fewer roots (647 at present) than usual *dhātupāṭha*s (that have of the order of 2200). This is also because we do not consider *dhātupāṭha* entries that have no obvious attested form in the available corpus – admittedly a rather subjective criterion, until enough corpus is digitalized, and our computer tools are robust enough to scan it entirely and spot their possible occurrences.

We developed generation paradigms for the major tenses/aspects/moods of classical Sanskrit. First, the present system (accounting for present [*laṭ*], imperfect [*laṅ*], optative [*liṅ*] and imperative [*loṭ*]), in the two voices (*parasmaipada* and *ātmanepada*). The allowed present classes [*gaṇa*] are indicated in the lexicon, together with the present form of the third person singular [*ti*]. Thus, for entry *gam*, the lexicon specifies ‘[1] pr. (*gacchati*)’. This specific form is used as a consistency check that indeed the conjugation generator produces this exact form.

We also systematically generate forms for the two forms of future (*lṛṭ* and *luṭ*), for perfect [*liṭ*], and for the passive forms of the present system (*karmaṇi*). We also provide for the periphrastic perfect forms of roots of class 10 by generating the form in [*-ām*] prefixing the perfect forms of auxiliaries *kṛ*, *bhū* and *as*. We have also implemented generative paradigms for the 7 varieties of aorist and injunctive (resp. *luṇ* and *leṭ*), but we do not generate all the forms for all the roots, only the combinations that are deemed productive enough to appear in corpus (this is a subjective criterion, induced from mention in Whitney roots [[24](#_bookmark27)], and occasionally completed when a missing form is spotted). We also implemented schemes for benedictive/precative mood (optative aorist *āśirliṅ*), conditional (preterit of future *lṛṅ*) and subjunctive (*leṭ*), but are generating such rare forms only for roots for which there is attested usage, mostly in mantras.

We generate a considerable number of primary nominal derivatives from roots

[*kṛdanta*]. We have implemented generative schemes for most participles: present participles (active, middle and passive), past passive participles [*kta*], past active participles [*ktavat*], gerundives (future passive participles) in their three forms in *-ya*, *-īya* and *-tavya*, future active and middle participles, perfect participles (active and middle). All such constructions yield adjectival stems, which in turn generate their forms in all genders, numbers and cases, leading to a very extensive lexicon of inflected words. All these forms are recorded with proper tags, allowing for recognition of their origin, and thus of their proper functional dependencies. Besides those, the remaining nominal *kṛdantas* (essentially agent nouns and action nouns) are not systematically generated, but are deemed lexicalized.

Finally, we generate absolutive forms in *-tvā* and *-ya*, as well as forms in *-am* (so-called *ṇamul*) for roots that admit this construct for iterative meaning. We also provide infinitive forms, both in their usual form in *-tum* (so-called *tumun*), and in their special forms in *-tu*, usable with auxiliary forms of *kāma* or *manas*, in order to recognize compounds such as *draṣṭukāmaḥ* and *draṣṭumanā*.

Secondary conjugations (*sanādi*) are not systematically generated. The lexicon indicates the secondary conjugations available for each root. Their forms are generated from the lexicalized 3rd person singular indicated in the lexicon, due to the great variability of such forms. The lexicon also documents whether nominal entries give rise to denominative verbs, and what is their formation scheme and voice.

A specific technology was developed for storing such an enormous amount of forms (several millions) marked with their morphological parameters, using specific data-structures inspired from automata theory and allowing maximum sharing of information. The Zen toolkit, a set of Ocaml[1](#_bookmark0) data structures and algorithms adapted to the task, is documented in [[5](#_bookmark8), [7](#_bookmark10)]. The name ‘Zen’ was chosen to evoke the simplicity of the informatics concepts used in its design, based on the functional programming paradigm. The whole library fits in 5000 lines of Ocaml code.

# Segmentation by sandhi analysis

The Zen toolkit was then extended to develop finite-state transducers adapted to the phonetic smoothing (sandhi) between contiguous words. Thus, it is possible to represent the phonetic smoothing that occurs at the junction of word forms (and during compound formation) by a finite automaton operating on the segmented representation (*padapāṭha*). This is possible because external (i.e. inter-*pada*) sandhi is a finitely representable regular relation). Furthermore, such relations are invertible, and thus we may revert the process, and compute all possible segmentations from a given continuous enunciation (*saṃhitā*). The corresponding non-deterministic transducer operates directly on the databanks produced by the morphological generator, properly annotated with prediction points where the context may trigger a sandhi rule.

1[https://ocaml.org](https://ocaml.org/)

This algorithm and its completeness analysis are given in [[9](#_bookmark12)]. This technology was generalized to a general paradigm of non-deterministic computation called Effective Eilenberg machines by Benoît Razet [[15](#_bookmark18), [16](#_bookmark19), [22](#_bookmark25)].

## The phases transducer

The segmenting transducer is a two-level automaton. The lower level consists in individual lexicons of forms, one for each lexical category. The upper level connects these lexicons according to a transition diagram of *phases*. Figure [1](#_bookmark1) gives a simplified version of the phases automata, with 10 phases.

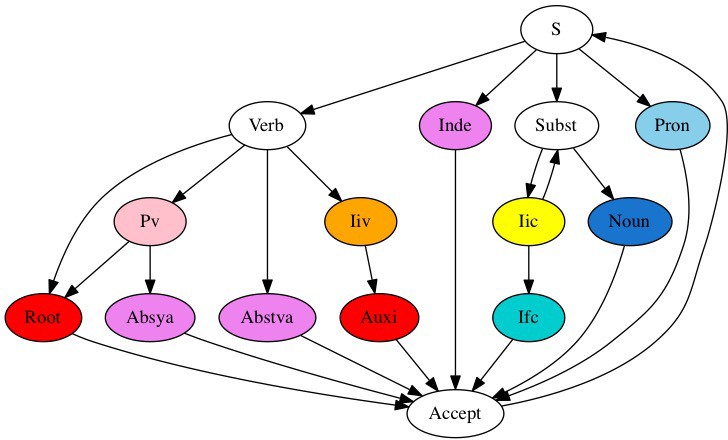


Figure 1: A simplified segmenter with 11 phases

The diagram may be read from top to bottom as the various ways one may recognize an inflected word (*pada*), going from the starting state S to the Accept state. Nominals (*subantas*) start at node Subst. Non-compounds are recognized at phase Noun. Compounds are recognized by going through phase Iic, which corresponds to the data bank of bare stems. The second component of the compound will be found either by going through phase Noun, or through phase Ifc, that recognizes forms that can occur only as right components of compounds, like we discussed above for the formation of *bahuvrīhi* and *avyayībhāva* compounds. The Ifc phase also contains forms that are specified as such in the grammar, like *-kāraḥ* (maker\_of). Multiple component compounds are recognized by looping through the Iic phase.

Finite verbal forms (*tiṅanta*s) are recognized at phase Root for root forms. Forms of verbs with preverbs are recognized by going through phase Pv, which

corresponds to attested concatenations of preverb particles [*upasarga*]. Absolutive root forms in *-tvā* are recognized at phase Abstva, whereas absolutive forms in *-ya* (corresponding to roots prefixed with preverbs) are recognized at phase Absya. Forms obtained through the *cvi* construction, such as *dravībhavati* are recognized through the sequence Iiv then Auxi (that contains finite forms of auxiliary verbs). Finally, indeclinables and pronominal forms are recognized respectively at phases Inde and Pron.

Sentences may then be recognized by recognizing their successive words, using the upward going arrow from Accept to S. The diagram may be used to generate a continuous enunciation from its list of words, interpreting each down-going arrow as sandhi, or conversely as generating the *padapāṭha* from a continuous enunciation, interpreting the arrows as *sandhiviccheda*.

## Treatment of preverbs

Each root in the lexicon is indicated with the preverbs (*upasarga*) that it may combine with. Each such preverb is a hyperlink to the corresponding verbal entry, making it easy to navigate quickly to the relevant information, back and forth. Each verbal entry has in turn its own list of which extra *upasarga*s it accepts. When the lexicon structure is compiled into grammatical information, as a first pass in the tool-building process, the transitive closure of this in- formation is computed, so that the system knows for each root what are the allowed sequences to use in front of conjugated forms (*tiṅanta*s), and for each such sequence what combinations of present class (*gaṇa*) and voice (*pada*) are permitted. This allows such compound preverb strings [*upasargamālā*] to be recognized in a holistic way at phase Pv rather than piecemeal, which saves on possible ambiguities.

Thus, we do not have to store forms given with their preverbs, we only generate root forms. But we are obliged to be more generous on such forms than what the bare root alone allows. For instance, root *viś* is used stand-alone only in active voice (*parasmaipade*): *viśati*, but admits middle voice (*ātmanepade*) when used with *upasarga ni-*: *niviśate*. Thus, we generate form *viśate* in the *tiṅanta* bank, and at recognition time we shall recognize *niviśate*, but will filter out *viśate* if stand-alone. This is consistent with *sūtra* (I,3,17), and more generally the mechanism accommodates faithfully section (I,3) of the grammar.

A difficulty occurs for sandhi. Preverbs are attached to root and *kṛdanta* forms by a tighter glueing operation than inter-word sandhi, incurring retroflexion. Thus, root *nī*, with present *nayati*, admits form *praṇayati* when used with *upasarga pra-*. This induces, at recognition time, the need to use a different internal sandhi operation when following the link between nodes Pv and Roots in the diagram above.

## Completing the diagam

Vocative forms are treated specially, they have their own phase, and do not really participate to the sentence structure, they are rather part of the discourse structure. Furthermore, nominal stems in *-a*, by far the most frequent ones, have their vocative forms identical to their bare form used as first component of compounds [*pūrvapada*]. This ambiguity is avoided by imposing a chunk- final restriction, or extra punctuation after vocatives and interjections.

Then a specific mechanism takes care of privative [*nañsamāsa*] compounds. This induces splitting the nominal banks according to their initial *varṇa* (consonant vs vowel). This is possible while keeping the sharing of the banks, at no extra cost.

For roots of class 10 as well as causatives and denominative verbs, a periphrastic formation of perfect [*ṇamul*] is used, followed by perfect forms of the auxiliaries *kṛ*, *bhū* and *as*, like in sentence *janayāṃcakāra kaṭaṃ devadattaḥ* (Devadatta made a mat). This induces two more phases in the segmenting automaton. A similar construction is the *cvi* inchoative construction mentioned above, and the use of certain particles like *alam*, *tiras*, etc. as specific prefixes [*gati*] to certain root forms.

We now get an extended graph of transitions of the segmenting transducer, represented in Figure [2](#_bookmark2) below. The complete graph used in the current version has of the order of 50 phases, accommodating transitions from preverbs to *kṛdanta*s, but also constructions that are less common, such as *draṣṭukāmaḥ*.

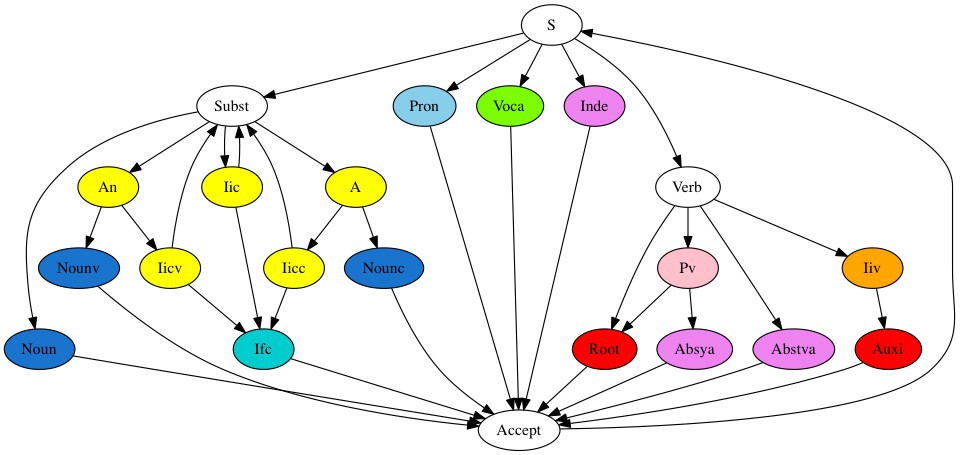


Figure 2: An extended segmenting transducer

# Graphic interface

The Sanskrit Reader companion is a Web service that accepts Sanskrit input in various formats, and computes the shared forest of all segmentation solutions in

the form of an interactive interface, where each word or morpheme recognized in the process is represented graphically as a colored rectangle, aligned with the input string. The colors of the rectangles are the colors of the automaton phases as depicted on Figure [2](#_bookmark2): *tiṅantas* are red, *subantas* are blue, *prātipādikas* are yellow, etc. If a segment is mandatory, in the sense of being a component of every solution, it stands alone in its column, and is marked with a blue check sign. Otherwise, a segment is marked with 2 signs: one green check sign, and one red cross sign. Both are clickable buttons linked to orthogonal actions. Clicking on the green button will make the segment mandatory, by erasing all solutions not going through it. Clicking on the red button will eliminate the segment, erasing all solutions going through it.

Each rectangle is itself mouse sensitive, and if clicked will display the morphological parameters leading to the generation of the item. This allows the user to check the dubious choices. The display gives the morphological tags in a pop-up window, where the lexical stems are hypertext links to the relevant entry in the lexicon.

The full description and analysis of this graphic interface is available in [[2](#_bookmark5)]. An example is shown in Figure [3](#_bookmark3) below. The user has entered the string “śvetodhāvati” and he is at the point of wondering which blue segment to select, yielding either “the white one runs” or “the dog runs towards here”.

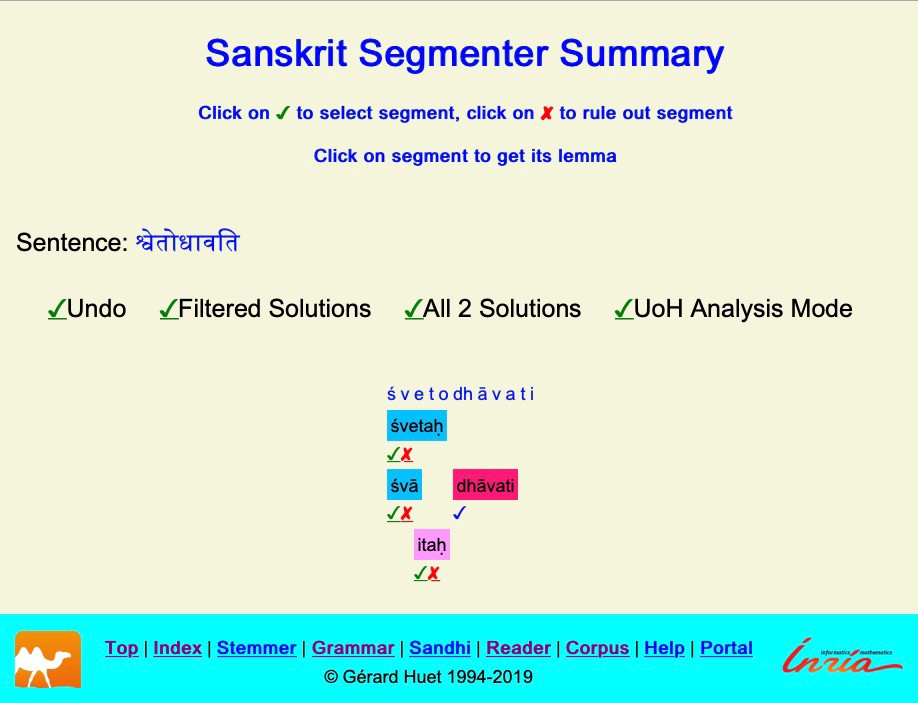


Figure 3: A typical ambiguity

# User-aid

A common condition of failure of the segmenter is to be unable to reconstruct a correct chunk of input from sandhi combinations of the generated forms. In the interface, this appears as a black rectangle. This rectangle may be clicked (on an ace-of-spade underlying symbol) in order to access a very useful tool, called User-aid. It helps you in three different kinds of ways. First, it allows you to revise the chunk, if there is a mis-spelling, remove unclear parts or even elide it completely. Else you may try to solve the chunk by providing a stem together with its case, gender and number (assuming the chunk is a nominal form). But the system offers you possible choices for this, by guessing possible lemmatizations. Furthermore, when the corresponding stem is an entry in the Monier-Williams dictionary, it indicates this possibility by making the stem a link to the corresponding entry.

When the choice is validated, the stem is recorded with its gender in a local cache, used from then on as a local supplement to the generating lexicon. That is, in following uses of the reader, all forms of this new nominal stem will be recognized. This is very useful, since e.g. a specialized Āyurveda technical term is likely not to be recorded in the Heritage dictionary, but once encountered in a text it has good changes to reappear later, perhaps in a different case and number disguise.

Of course, this facility is available only for users who have installed the ser- vices on their localhost Web server, since the caching database is a private persistent structure.

# Corpus manager

One recent component of our suite of tools is the Corpus Manager. It allows to store the final result of an analysis in a persistent file in a hierarchical library structure representing the morphologically-annotated corpus. Three levels of users have been defined: reader, annotator, and manager. The reader can only visualize the annotated corpus. An annotator may build new sections of the corpus, using the Sanskrit Reader tool, and update revisions, in his own area of competence. A manager creates new branches in the corpus repository, and gives annotator rights to competent individual scholars. The whole scheme is implemented using the Git infrastructure.

This effort is described in [[14](#_bookmark17)]. This facility is still under design, and is not robust enough to be put in production for large-scale corpus processing. It is used mostly at present for displaying the dictionary’s quotations in analyzed form, and for organizing texts used as teaching material.

# Dependency analysis

The Sanskrit Reader badly over-generates. It sees *śleṣa* everywhere, and makes no attempt at making sense of the word combinations exhibited in the solutions.

Therefore, we want now to filter out the non-sensible solutions, in the hope to selecting a unique solution, preferably the intended one.

The analytic section of *Aṣṭādhyāyī* gives a semantic analysis of sentences in terms of expectancies [*ākāṅkṣā*], similar to dependencies from the tradition of Structural Syntax. Each word in the sentence has a role [*kāraka*] and if these mutual roles are semantically consistent the sentence passed the consistency check. *kāraka*s are linked to the cases of the substantives by a many-to-many relation, so there are choices there.

An experimental dependency parser based on simple constraints solving was implemented in 2006 [[11](#_bookmark14)]. It gives good results on simple sentences, but is not usable for long complex sentences. This tool is still available on the Sanskrit Heritage site, but it is becoming deprecated with the efforts of Amba Kulkarni at University of Hyderabad with her Saṃsādhanī parser [[19](#_bookmark22), [20](#_bookmark23), [17](#_bookmark20), [18](#_bookmark21)].

It is now possible to call this dependency parser from the Heritage site reader. When the number of segmentation solutions has been reduced enough, one may invoke Saṃsādhanī to determine the “best” solution from the point of view of semantic consistency and statistical estimation. The two sets of tools are thus complementary at this stage.

# Evaluation

## Lexical completeness

Since the segmenter is lexicon-directed, its completeness depends on the vocabulary completeness of the dictionary, together with the completeness of the morphological derivation scheme, and the completeness of the sandhi relation description.

The Heritage dictionary is currently accommodating 647 roots. Some roots are declared in multiple classes [*gaṇa*], so the total of distinct present systems is 792. This is much less than the 2147 entries of the extensive root catalogue Dhāturūpaprapañca, but it covers fairly completely the non-Vedic forms of Whitney roots. The verbal system is generously generative, as well as participles, as explained above. Every root is associated with a list of possible preverb combinations, yielding a total of around 2400 verbs.

Other nominal primary derivatives then participles, absolutives and infinitives, mostly action nouns, agent nouns and abstract nouns are not defined as generative schemes, but are lexicalized on a by-need basis.

The main limitation concerns secondary derivatives [*taddhitas*], which must be lexicalized at present, as well as the frequent compounds built with particles *su-*, *dur-*, *sa-*, *vi-*, etc.

Similarly, *prādi* compounds are not generative. These are exocentric [*bahu- vrīhi*] constructs, whose first member is a preposition [*karmapravacanīya*]. At present, they are not systematically generated, and must be lexicalized. When their *uttarapada* (2nd constituent) is a *kṛdanta*, there may be ambiguity with the role of the preposition as a preverb (*upasarga*) co-denotional with the *kṛdanta*.

These are completely different constructions. For instance, take *nāyakaḥ*, an agent noun of root *nī*, meaning “leader”. It may be preceded by *upasarga pra-*, leading to *kṛdanta praṇāyakaḥ*, meaning “commander”. But it may also form with the same preposition a *prādi* compound *pranāyakaḥ*, now an adjective meaning “without chief”. Here we are lucky, since the retroflexion induced by the closer co-denotative junction is an indication of their difference. But in many cases we shall get the same *prātipadika*. Thus, we have a genuine ambiguity in a word such as *nirvācyaḥ*. Should it be recognized as a compound, meaning “what should not be discussed”, or as a participle, meaning “what should be explained”?

Note that this problem is not an ambiguity of Sanskrit, nor a problem for Pāṇini, who clearly distinguishes the two by their accent. It is an ambiguity of the writing convention of Classical Sanskrit, where the accent is not marked. This is a source of difficulty for beginning students, especially when they use a dictionary such as Monier-Williams’, since the two forms are far away from each other in the entries ordering We have to understand better the productivity conditions of *prādi* compounds in order to understand the situations of ambiguity which demand proper lexicalization. In the example above, *nirvācyaḥ* returns two solutions, the lexicalized *prādi* compound *nirvācyaḥ*1, and *nirvācyaḥ*2, analyzed as a gerundive participle of verb *nirvac*.

Other compounds may be split between regular and non-regular compounds. Regular compounds use the stem form of their first component, and glue the two components using the standard inter-pada external sandhi. Those are recognized by our transducer, going through the Iic phase in the diagram above. However, iteration of compounding will be flattened in the segmenting phase. Thus (A-B)-C and A-(B-C) will be both recognized as the linear sequence A-B-C: one needs an additional level of analysis in order to favor one of the two constructions. The exocentric [*bahuvrīhi*] compounds are recognized as adjectives in all genders, because we generate in the Ifc banks the missing gender forms of their right component. Thus, neuter *ambara* (cloth) will generate a masculine pseudo-form *-ambaraḥ* in lexicon Ifc, usable to recognize e.g. *pītāmbaraḥ* (he who wears a yellow cloth), but precluding the recognition of wrong *\*ambaraḥ*. Finally, privative compounds (whose first component is *a-* or *an-* according to whether the second component starts with a consonant or a vowel) are recognized by phases A and An respectively in Figure 2 above.

Irregular *aluk* compounds, whose first component has kept its inflected form, such as *kaṇṭhekāla*, must be lexicalized. Same for compounds whose glueing operation is different from the standard external sandhi, such as *pṛṣodara*, where the final *t* of its first component *pṛṣat* has been elided. These irregular compounds are not productive, and they are in finite number. Note that similarly in Aṣṭādhyāyī there are open lists of such exceptions provided by collections [*gaṇa*] such as *pṛṣodarādi* in *sūtra* (6,3,109).

There is no pretense of covering Vedic corpus. Only a few Vedic forms are lexicalized, or accommodated by specific extensions of morphology, in order to cover the main Vedic mantras. Similarly, occasional irregular Epic forms are occasionally recorded.

## Sandhi completeness

The covering of sandhi rules is extensive, and corresponds very closely to Pāṇini’s treatment. Actually the sandhi rules, in the systematic form of a quadruplet (*x, u, v, w*) where *x* is the left context, *u* is a suffix of the left word, *v* is a prefix of the right word, and *w* is the string replacing *uv*, are all documented in the computer program with the sequence of Paninian sūtras justifying them.

A special sandhi relation links the preverbs to the roots and their primary derivatives. This tighter glueing incurs retroflexion. It is harder to relate the corresponding rules to Aṣṭādhyāyī, since some roots like *nī* are presented in *dhātupāṭha*s with markers preparing this retroflection in proper contexts. We made a separate analysis, for the pairs (root,preverb) allowed by the lexicon, and came with an admittedly ad-hoc set of rules, which in practice seems sufficient. Only very rarely some new retroflexion context is discovered, and a new rule added.

There is a restriction concerning vocatives, which must be ended by a space in the input stream. This is justified by the frequent ambiguity between bare stem and vocative form of stems in *-a*, and justified by prosody considerations.

A special mechanism for allowing *sa* for pronoun *saḥ* in proper contexts has been implemented. But no special sandhi provision is enforced for dual forms.

## Possible remedies to incompleteness issues

The main source of incompleteness of the segmenter concerns secondary derivatives [*taddhitānta*s]. Some of the *taddhita pratyāya*s involve raising their stem to the upper grade [*vṛddhi*]. Recognizing them would involve identifying these stems in order to build the proper vṛddhified banks, and identifying the relevant sandhi rules that apply for the glueing of the *pratyāya*. The main problem is that the grammar defines the range of these *taddhita* affixes in terms of the meaning of the stems they apply to, and this is not easily interpretable computationally.

## Applicative uses of the platform

Overall, completeness is satisfactory for classical corpus, with a rough recall of 90% for a text like Gīta. The problem is the poor precision, since every possible decomposition of the input is returned. For a moderately complex sentence, the number of solutions easily runs in millions. Hopefully, the graphical user inter- face manages to represent all the segmentations in one shared two-dimensional graphic that fits in one Web page, and that a competent user may resolve in a few clicks. The segmenter is very efficient, and the result is returned usually instantly. Only long chains of compounds may result in a noticeable delay.

The current system is useful for tagging corpus, and for teaching purposes. Tagging still needs a competent human annotator, but we are experimenting with statistical optimization (using the manually tagged corpus of Oliver Hellwig’s DCS) and hope to be able to reduce the number of plausible solutions to a few candidates, which fed through Amba Kulkarni’s Saṃsādhanī parser would result in a unique analysis.

Actually, compounds are analyzed only as sequences of bare nominal stems followed by an inflected nominal. These are only *pre-compounds*, in that they represent only the frontier of several possible binary trees, and ignore their type (exocentric, etc.) A more complete tagging of the text, restoring compound structure, could benefit from the recent work of Pavankumar Sutuluri and Amba Kulkarni on generating compounds[[23](#_bookmark26), [21](#_bookmark24)].

The main current usage is by people using it as an aid to translate Sanskrit text. It is notably useful for beginners, who can use the tools without extensive knowledge of morphology and sandhi. They may even process devanāgarī input imported from corpus libraries without having to learn the script.

Experiments for teaching purposes have been conducted, for instance by Pr Varalakshmi from the Sanskrit Academy at Osmania University in Hyderabad. She used our tools for remote teaching at the Silicon Andhra University.

# Practicalities

## The Sanskrit Inria Web server

The Heritage platform is available as a family of Web services hosted at URL

[sanskrit.inria.fr](http://sanskrit.inria.fr/). No registration is needed to use these tools, which neither display commercials nor record users’ interactions. A comprehensive Reference Manual is available online at sanskrit.inria.fr/manual.html.

It is possible to download the whole package as a stand-alone software for workstations running versions of UNIX such as Linux or Apple’s MacOS. The installation procedure is documented in the user’s manual.

The entire source code, under a pleasing literate programming format, is available as a 780 pages pdf document sanskrit.inria.fr/DOC/Heritage\_ Platform.pdf.

## Installing your own server

The development repository is freely accessible at gitlab.inria.fr/huet/ Heritage\_Platform, together with its companions gitlab.inria.fr/huet/ Zen (the Zen toolkit) and gitlab.inria.fr/huet/Heritage\_Resources (its linguistic resources). These Git repositories may be cloned for stand-alone installation on users’ own servers.

# Conclusion

We presented a comprehensive platform for computer processing of Sanskrit text. We gave the main lines of its design, and discussed its shortcomings.

This effort was funded by Inria, in the framework of a research cooperation with the Sanskrit Studies Department at University of Hyderabad. The main contributors of the software development are Gérard Huet for the Heritage

Dictionary, Zen toolkit, morphology generators and sandhi analyzer, Pawan Goyal for the graphical interface, Monier-Williams alignment and user-aid facility, Idir Lankri for the Corpus Manager, and Amba Kulkarni for the Saṃsādhanī parser and its alignment with the Heritage Platform.

# References

* + 1. P. Goyal and G. Huet. Completeness analysis of a Sanskrit reader. In M. Kulkarni and C. Dangarikar, editors, *Recent Researches in Sanskrit Computational Linguistics*, pages 130–171. D.K. Printworld, 2013.
    2. P. Goyal and G. Huet. Design and analysis of a lean interface for Sanskrit corpus annotation. *Journal of Linguistic Modeling*, 4(2):117–126, 2016.
    3. G. Huet. Structure of a Sanskrit dictionary. Technical report, XIth World Sanskrit Conference, Torino, 2000.
    4. G. Huet. From an informal textual lexicon to a well-structured lexical database: An experiment in data reverse engineering. In *Working Conference on Reverse Engineering (WCRE’2001)*, pages 127–135. IEEE, 2001.
    5. G. Huet. The Zen computational linguistics toolkit. Technical report, ESSLLI Course Notes, 2002.
    6. G. Huet. Towards computational processing of Sanskrit. In *International Conference on Natural Language Processing (ICON)*, 2003.
    7. G. Huet. Zen and the art of symbolic computing: Light and fast applicative algorithms for computational linguistics. In *Practical Aspects of Declarative Languages (PADL) symposium*, 2003.
    8. G. Huet. Design of a lexical database for Sanskrit. In *Workshop on Enhancing and Using Electronic Dictionaries, COLING 2004*. International Conference on Computational Linguistics, 2004.
    9. G. Huet. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *J. Functional Programming*, 15,4:573– 614, 2005.
    10. G. Huet. *Themes and Tasks in Old and Middle Indo-Aryan Linguistics, Eds. Bertil Tikkanen and Heinrich Hettrich*, chapter Lexicon-directed Seg- mentation and Tagging of Sanskrit, pages 307–325. Motilal Banarsidass, Delhi, 2006.
    11. G. Huet. Shallow syntax analysis in Sanskrit guided by semantic nets constraints. In *Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries*, ACM, New York, NY, USA, 2007.
    12. G. Huet. Sanskrit signs and Pāṇinian scripts. In A. Kulkarni, editor, *Sanskrit and Computational Linguistics*. D.K. Printworld, New Delhi, 2016.
    13. G. Huet. Sanskrit lexicography, past and future. In L. Wei, editor, *Research on the Language and Script in Buddhist Sutras*. Hangzhou Buddhist Academy, 2019.
    14. G. Huet and I. Lankri. Preliminary design of a Sanskrit corpus manager. In G. Huet and A. Kulkarni, editors, *Computational Sanskrit & the Digital Humanities*. D.K. Printworld, New Delhi, 2018.
    15. G. Huet and B. Razet. The reactive engine for modular transducers. In

K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, editors, *Algebra, Meaning and Computation, Essays dedicated to Joseph A. Goguen on the occasion of his 65th birthday*, pages 355–374. Springer-Verlag LNCS vol. 4060, 2006.

* + 1. G. Huet and B. Razet. Computing with relational machines. Mathematical Structures in Computer Science, special issue in honor of Corrado Böhm, October 2015.
    2. A. Kulkarni. A deterministic dependency parser with dynamic program- ming for Sanskrit. In *Proceedings, Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 157–166. Charles University, Prague, 2013.
    3. A. Kulkarni. *Sanskrit parsing based on the theories of Śābdabodha*. Indian Institute of Advanced Study (DK Printworld distr.), 2019.
    4. A. Kulkarni, S. Pokar, and D. Shukl. Designing a constraint-based parser for Sanskrit. In G. N. Jha, editor, *Proceedings of the 4th International Sanskrit Computational Linguistics Symposium*. Springer-Verlag LNAI 6465, 2010.
    5. A. Kulkarni and K. V. Ramakrishnamacharyulu. Parsing Sanskrit texts: Some relation specific issues. In *Recent Researches in Sanskrit Computational Linguistics*, pages 130–171. D.K. Printworld, 2013.
    6. K. V. R. Pavankumar Satuluri and A. Kulkarni. Order of operations in the formation of Sanskrit compounds with special reference to introduction of samāsānta element and deletion of case endings. *Journal of Oriental Institute, Vadodara*, 66(4):77–86, 2016-17.
    7. B. Razet. *Machines d’Eilenberg Effectives*. PhD thesis, Université Denis Diderot (Paris 7), 2009.
    8. P. Satuluri and A. Kulkarni. Generation of Sanskrit compounds. In *Proceedings of ICON 2013,* pages 77–86, Noida, India, December 2013.
    9. W. D. Whitney. *Roots, Verb-forms and Primary Derivatives of the Sanskrit Language*. Motilal Banarsidass, Delhi, 1997. (1st edition 1885).