Completeness Analysis of a Sanskrit Reader

Pawan Goyal & Gérard Huet

INRIA Paris-Rocquencourt,
BP 105, 78153 Le Chesnay Cedex, France

Abstract. We analyse in this paper differences of linguistic treatment of Sanskrit in the Sanskrit Heritage platform[1] and in the Paninian grammatical tradition.

1 General methodology

The general assumption behind the design of the Heritage Sanskrit Reader is that sentences from Classical Sanskrit may be generated as the image by a regular relation R of the Kleene closure W^* of a regular set W of *words* over a finite alphabet Σ. Think of W as the vocabulary of (inflected) words (*padas*) and of R as sandhi.

The computerized lexer underlying the Heritage Reader essentially proceeds by inverting relation R over the candidate sentence w in order to produce a finite sequence $w_1, w_2, ...w_n$ of word forms, together with a proof that $w \in R(w_1 \cdot w_2 ... \cdot w_n)$. The word forms w_i must be justified being valid word forms of Sanskrit (i.e. *padas*), and some justification must be offered that the combination of such word forms makes sense. The first justification consists in exhibiting w_i as the lemmatization of some root stem, according to valid rules of morphology. The second justification consists in giving some dependency analysis of sentence w using assignments of semantic roles for the individual w_i's consistent with their morphological analysis. Both kinds of justifications must be ultimately related to the traditional methods of Sanskrit grammar (*vyākaraṇa*). That is, that each w_i corresponds to some w'_i, obtainable by a valid Paninian derivation sequence, and that the concatenation of the sequence $w'_1, w'_2, ...w'_n$ yields by some valid Paninian derivation a final sequence of phonemes w' equivalent in a strong sense to the original w.

Thus, to fix ideas with a concrete trivial example, the sentence (in Roman transliteration):
rāmogrāmaṃgacchati, equivalently represented as:
rāmogrāmaṅgacchati[2], may be analysed as the sequence:

[1] `http://sanskrit.inria.fr`

[2] The normalization procedure implemented in the Heritage Reader, where the *anusvāra* before a consonant is replaced by the homogeneous nasal of the consonant, corresponds to the following rule in *Pāṇini*'s grammar:

8.4.58 *anusvārasya yayi parasavarṇaḥ*: In *saṃhitā*, if *ṃ* is followed by a consonant in *pratyāhāra yay*, it is replaced by the homophonic nasal of the consonant.

[*rāmaḥ* { nom. sg. m. }[*rāma*]⟨aḥ|g →og⟩]
[*grāmam* { acc. sg. m. }[*grāma*]⟨m|g →ṅg⟩]
[*gacchati* { pr. [1]ac. sg. 3 }[*gam*]⟨⟩]
where the various forms are lemmatizations of respectively the basic stems (*prātipadikas*) *rāma* and *grāma* and the root *gam*. The root *gam* (to go) being transitive (*sakarmaka*), and its form *gacchati* being in the active voice, we may assign the agent role (*kartṛ*) to the nominative form *rāmaḥ* and the goal role (*karman*) to the accusative form *grāmam*, yielding the interpretation "Rāma goes to the village". It should then be a routine exercise (for a *vyākaraṇa* specialist) to use the sequence of lemmas and phonetic rewrite rules as a guide to the generation of a Paninian derivation of the original sentence.

The method looks hardly original, in view of modern computational linguistics, and it could be applied *mutatis mutandis* to many natural languages, with Σ representing the set of phonemes of the language, W being the vocabulary of (inflected) words and R being some phonetic smoothing (called *sandhi* in Sanskrit). Often, at least in Western languages such as English or French, the phonetic treatment is given separately as a layer of speech understanding, words are strings over an alphabet of written characters, isolated by blanks in the written sentence, and more importance is assigned to a structural notion of *syntax* where word order is important. The basic assumption is that there is a clear hierarchy of levels of linguistic description (traditionally called phonetics, phonemics, morphology, syntax, semantics, pragmatics), corresponding to more or less modular computational processes, and that the interfaces between these processes are the essential components allowing generation in one direction, and analysis in the reverse direction in a general model of universal linguistics. It could thus appear that the treatment of Sanskrit is just slightly biased, in having a significant *sandhi* component because of the faithfulness of its written representation to oral enunciation, and in having a shallow *syntactic* component because of the richness of its morphology. Indeed, the treatment of Sanskrit linguistics by Paul Kiparsky [19] is essentially a precise elaboration of such processes, at least in the synthesis direction. One is thus hardly surprised at the general presentation of the methodology.

However, things are not that simple. For one thing, if this methodology was straightforward, one wonders why the traditional methods of linguistics analysis (*vyākaraṇa*) do not explain sentence formation in this simple, intuitive, modular fashion. Indeed, the traditional Paninian presentation of Sanskrit grammar by the trinity of the 3 sages (*trimuni*) of ancient India is an extremely complex interweaving of formal transformations operating at all levels in parallel, and it has been copiously criticised by linguists of the 19th century such as Whitney. However, there exists to this date no grammatical description of Sanskrit that is as precise and as complete as the traditional one, and Sanskrit grammar experts well aware of modern linguistics theories such as George Cardona and Peter Scharf [23] maintain that mixing of levels is unavoidable for accounting for the complexity of Sanskrit grammar. Thus it remains a challenge to provide a modular description of computational linguistics processes adequate to recognize

correct Sanskrit sentences, and there is to this date no better justification of such than to relate their operations to that of the tradition. We shall attempt to meet this challenge in the present paper, and to provide a rationale for the completeness of the Heritage Reader with respect to a well-identified subset of classical Sanskrit.

2 Morphology

2.1 Limiting the recursion

The first difficulty we are facing is in the definition of the set W of *words* (*padas*). It cannot simply be explained as the set of inflected forms of a finite set of "atomic" words. For instance, it is possible to form compounds of arbitrary length, making the set of substantive bases (*prātipadikas*) infinite. Furthermore, the process of turning a substantive base into a denominative verb is productive[3], even for compounds, although this last possibility is infrequent. If we examine the diagram given in [20], which purports to explain the recursive structure of Sanskrit morphology as authorized by the traditional grammar, it seems hopeless to invert it directly in order to obtain a complete parser. By contrast, the (simplified) finite-state diagram underlying the Heritage transducer given in Figure 1 reduces drastically the allowed recursions. For one thing, it forbids denominative verbal forms, except for the conjugated forms of a finite set of lexicalized verbal stems. Also, it makes many assumptions on compounding which must be explained and justified.

Let us explain briefly Figure 1, in order to relate it to the traditional terminology. Each colored oval, called a *phase* of the underlying automaton, corresponds to a finite set of forms. Going from top (node S) to bottom (node Accept) consists in following the arcs, selecting forms from the intervening phases, and glueing them with sandhi, in order to obtain one word (*pada*). The sub-diagram below the Subst node builds substantival forms (*subantas*). The sub-diagram

[3] As per Pāṇini's grammar, the main *sūtra*s corresponding to the nominal verbs ordain 6 different suffixes:

- *kyac*: 3.1.8 *supaḥ ātmanaḥ kyac*. Ex: *ātmanaḥ putram icchati →putrīyati*.
- *kāmyac*: 3.1.9 *kāmyac ca*. Ex: *ātmanaḥ putram icchati →putrakāmyati*.
- *kyaṅ*: 3.1.11 *kartuḥ kyaṅ salopaḥ ca*. Ex: *śyena iva ācarati →śyenāyate*.
- *kyaṣ*: 3.1.13 *lohitādiḍājbhyaḥ kyaṣ*. Ex: *alohito lohito bhavati →lohitāyate*.
- *ṇiṅ*: 3.1.20 *pucchabhāṇḍacīvarāt ṇiṅ*. Ex: *puccham udasyati →utpucchayate*.
- *ṇic*: 3.1.21 *muṇḍamiśraślakṣṇalavaṇavratavastrahalakalakṛtatūstebhyaḥ ṇic*. Ex: *miśraṃ karoti →miśrayati*.

Suffix *ṇiṅ* is specific only to 3 nominal bases, *kyaṣ* is specific to the *lohitādigaṇa*, while the other suffixes are applicable to any nominal base. Among these suffixes, the suffix *ṇic* is the most productive and there is a *gaṇasūtra: tatkaroti tadācaṣṭe*, which specifies this suffix after any nominal base in the sense of 'he does that', 'he behaves like', 'he says that'.

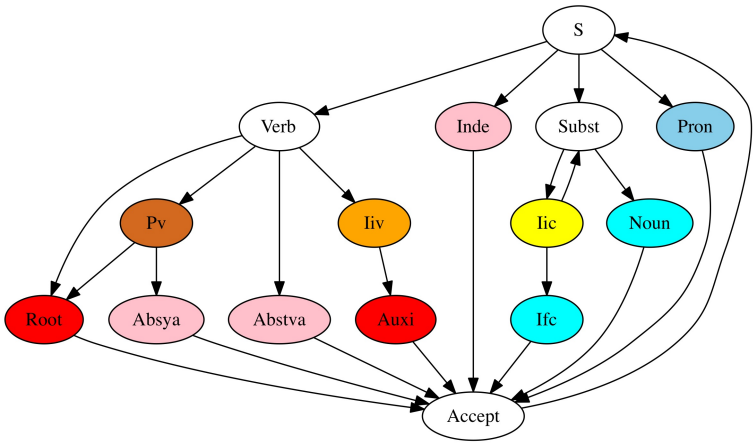

Fig. 1. A simplified lexical analyser.

below the Verb node builds verbal forms (*tiṅantas*). The Inde node builds indeclinable forms, the Pron node builds pronominal forms. Phase Noun contains declined forms of autonomous atomic substantive and adjective stems, found in the lexicon with their attested gender parameters. Phase Ifc contains the non-autonomous ones, usable only as the right-hand component of a compound, such as root substantival forms like *-gam* (gone to). Phase Iic contains bare stems of nouns (*prātipadikas*) such as *deva-*. Phase Root contains conjugated forms of roots, phase Pv contains attested sequences of preverbs (*upasargas*) such as *ā, sam, nirā, samabhivyā*, etc. Phase Auxi contains conjugated forms of the three auxiliary roots *bhū*, *as* and *kṛ*, phase Iiv contains periphrastic forms of a substantive formed with an *-ī* suffix (technically designated as *cvi* in Paninian terminology). The path Iiv-Auxi allows generation of compound verbal forms such as *pavitrīkaroti* (he purifies). Phase Inde contains a mix of indeclinable forms, such as grammatical words (*ca, vā, api*, etc.), and adverbs (*yathā*). Absolutives in *-tvā* (*kṛtvā*) are stored in Abstva, whereas absolutives in *-ya*, necessarily prefixed with a preverb, are put in phase Absya. Infinitive forms (*kartum*), which may or may not be prefixed with a preverb, appear both in Inde and in Absya.

The diagram of Figure 1 is a simplification of the actual automaton transition graph of the Sanskrit Heritage lexical analyser, that has numerous other phase databanks to account for constructions such as periphrastic perfect, but it suffices to give a general idea and start the discussion.

2.2 Morphology generation

Each of the colored nodes of the diagram of Figure 1 represents a data bank of forms. These data banks are filled in during a series of preliminary passes. First the lexicon is analysed, to gather stems and their morphological parameters, such as permitted genders of nominal stems, allowed present classes (*gaṇa*) and

attested preverbs for roots. In a second phase, more stem generation occurs for roots, accounting for the various tenses/moods (*lakāras*), as well as absolutives and infinitives, but also participles (adjectival *kṛdantas*) in 10 varieties. Finally, inflexional morphology paradigms derive the inflected forms according to the morphological parameters, some of which being read from the lexicon, some of which being defined in specific tables.

It is not simple to relate our paradigmatic derivations and the operations of Paninian grammar. Some stem operations relate to morpho-phonetic operations well identified in the central rules (*sūtras*), such as taking the phonetic grades of *guṇa* or *vṛddhi*. Some are dispersed in various parts of the grammar, such as stem substitutions. Still others, such as retroflexion, occur in the final section (*tripādī*) of the *Aṣṭādhyāyī*, which comprises a list of rewrite rules which are applied iteratively at the end of the derivation process, in some kind of "phonetic smoothing" phase. Since the forms stored in our data banks are to be matched to the surface realization of the sentence, this phonetic smoothing must be applied at the time of generation of these forms. Some careful analysis must be done in order to understand the mutual interaction of the retroflexion rules and of the external sandhi rules (which are used in segmentation in the Heritage reader). This analysis will be given in section 4.4 below.

The main discrepancy between the Paninian processes and our reader operations is in the order of rewritings. Often the *sūtras* relevant to a given operation are dispersed in different sections of the *Aṣṭādhyāyī*, and thus it is hard to give the trace of the needed *sūtras*. However, in many cases, one can recognize the *sūtra* operations from the computer program. A complete analysis, in the specific case of future passive participles in *-ya* (*yat, kyap, ṇyat*), is given in Appendix A.

2.3 Variations in sandhi

One related difficulty concerns the precise definition of the sandhi relation labeling implicitly the arcs of the diagram. For most arcs, it is what Western linguists call "external sandhi". However, for the arcs issued from the Pv phase, it adds retroflexion rules, necessary to explain verbal forms of verbs (or participles) prefixed by preverb sequences. Furthermore, the forms stored in the various data banks (Noun, etc.) use such retroflexion rules in the formation of their stems and of their inflexions.

The correctness and completeness of our method involves thus a careful assessment of the mutual interaction (feeding, bleeding) between the rules of the morpho-phonetic processes of stem formation and inflection, the rules of external sandhi, and the final *tripādī* rules.

3 Specific problems of compounding

3.1 Pre-compounds vs compounds

Another set of difficulties concerns compounding. First of all, what our analyser recognizes by using the loop going through phase IIc in Figure 1 is not strictly

speaking a compound (*samāsa*), but what we call a *pre-compound* - a mere sequence of Iic bare stems ended in an inflected substantival form. Several possible compounds may be mapped to a unique pre-compound, representing the common frontier of the corresponding binary trees. Thus the sequence *ānandamayakośaḥ* will parse as a unique pre-compound *ānanda-maya-kośaḥ*, which has two possible interpretations as compositions of binary compounding, one of which is the intended one *(ānanda<maya)<kośa* (in Gillon's notation [7]). This has advantages (a unique pre-compound parse instead of C_n parses of an $n+1$-component compound, where C_n, the n-th Catalan number, is an exponential function of n, as explained in [13]), and drawbacks (some ulterior computation will be needed in order to resolve the ambiguity if one wants to get a handle on its meaning). Only at this ulterior stage one will be able to relate the precise compound construction to the relevant *sūtras*.

3.2 Forms allowed as left components of compounds

Heritage Treatment

Another problem is that the left component of a compound is not always a bare stem - it may be a gendered stem (e.g. *durgāpūjā*). It may even be a fully inflected form, like in so-called *aluk* compounds such as *gavāmayana* ("cows' travel" or year), where *gavām* is the genitive plural of *go*. We consider this last construction as non-productive, and assume that the finite set of such attested frozen forms is lexicalized. Other possible left components of compounds are particles which are not bare stems of stand-alone substantives, such as *su, vi, dur*. They must be accommodated in the Iic bank.

Aṣṭādhyāyī **Treatment**

The left component of a compound may undergo some operations as enumerated in Pāṇini's grammar and therefore, is not always a bare stem. Firstly, we have the problem of *aluk* compounds, where Iic can be a fully inflected form. The rules corresponding to *aluk* compounds are discussed in Pāṇini from 6.3.1 to 6.3.24. For example,

6.3.4 *manasaḥ sañjñāyām* : *aluk* applies to a *tṛtīyā*, which occurs after *manas*, when a constituent in combination follows and the derivate denotes a *sañjñā*. Thus, *manasādattā*. The Iic *manasā* is the inflected form of *manas* in instrumental case (*tṛtīyā*).

6.3.7 *vaiyākaraṇākhyāyāṃ caturthyāḥ*: *aluk* applies to a dative form (*caturthī*), which occurs after *ātman*, when a constituent in combination follows and the derivate denotes a name assigned by the grammarians. Thus, *ātmanepadam*. The Iic *ātmane* is the inflected form of *ātman* in *caturthī*.

Similar to the rules as enumerated above, the rules for *aluk* compounds are very specific and generate a finite set of compounds. Thus, this formation is non-productive and as a design decision in the Heritage system, we consider these compounds as frozen expressions and the finite set of these forms is lexicalized.

Another problem is that in some cases, the Iic stems may undergo some changes and may not be the same as the bare stem. These cases correspond to Pāṇini's rules 6.3.25 to 6.3.139. For example:

6.3.52 *pādasya padājyātigopahateṣu*: The word *pāda* is replaced by *pada*, when *āji*, *āti*, *ga* or *upahata* combine to follow. Thus *padopahataḥ*, *padagaḥ*.

6.3.70 *kāre satyāgadasya*: Augment *mum* is introduced to *satya* and *agada*, when *kāra* combines to follow. Thus *satyaṁkāraḥ*.

Similar to these rules, the rules in this section ordain various changes in the Iic of a compound. While most of the rules in this section are non-productive and the cases like 6.3.52 are being taken care of by declaring additional Iic stem *pada* for the word *pāda*, some of the rules which are productive and may require special treatment are:

6.3.34 *striyāḥ puṃvat bhāṣitapuṃskādanūṅ samānādhikaraṇe striyām apūraṇīpriyādiṣu*: In the place of a feminine word, the corresponding masculine form is substituted, when the feminine word has an actual corresponding masculine and does not end in the suffix *ūṅ*, if followed by another feminine word, but not when this word ends in a *pūraṇa* suffix or belongs to the list headed by *priyā*. Thus, *darśanīyabhāryā* and not *darśanīyābhāryā*.

6.3.61 *ikaḥ hrasvaḥ aṅyaḥ gālavasya*: The final of a nominal which ends in a vowel belonging to the *pratyāhāra ik*, with the exception of *ṅī*, is optionally replaced by its short counterpart in the opinion of Gālava. Thus, *grāmaṇiputraḥ* and *grāmaṇīputraḥ*.

A careful assessment of the rule 6.3.34 reveals that we actually take care of the words generated using this rule without any special treatment. The Iic stems which may qualify for this rule are those feminine stems, which have been derived from a masculine stem and as a result of this rule application, these stems will be substituted by their masculine stems. In the Heritage system, the corresponding masculine forms will always be stored in the Iic banks and thus, analyzing these compounds will not raise a special problem.

The case with the rule 6.3.61 is different because the Iic stem is shortened and may not be present in our Iic bank. However, this is an optional rule and some statistics from the corpora is required to figure out the productive nature of this optional shortening.

3.3 Forms allowed as right components of compounds

Heritage Treatment

Still another problem concerns the forms allowed as right components of compounds. Sometimes, and this is the case of the important class of *tatpuruṣa* compounds, there is gender preservation of the right component. We are thus assured of finding the corresponding form in the Noun database, which contain forms of words consistent with their lexicalized gender. But this is not true of the exocentric (*bahuvrīhi*) compounds, where a substantive may be lifted to an adjective used in a different gender, such as *caturmukhaḥ* "who has four heads", since *mukhaḥ* is not an autonomous form of neuter stem *mukha*. This forces us

to add forms such as *mukhaḥ* to the Ifc database. A similar problem arises with the so-called *avyayībhāva* compounds, where an adverb "turns to indeclinable" its right component, usually by coining a neuter-like form, like in *yathāśraddham* (according to your convictions), where the right component is similar to the accusative form of a hypothetical neuter stem *śraddha* induced from the originally lexicalized feminine stem *śraddhā*.

Aṣṭādhyāyī Treatment

Pāṇini's *sūtra* 2.2.24 (*anekam anyapadārthe*) states that many syntactically related *padas* combine in a *bahuvrīhi* compound and the compound denotes the meaning of something other than its own constituents. Therefore, it acts as an adjective and like other adjectives in Sanskrit, can take any gender.

In the case of *avyayībhāva* compounds, the Ifc is changed to neuter (shortened)[4] and the compound gets the *sañjñā* '*avyaya*'[5] and therefore, the nominal case ending is deleted[6]. Some exceptions where the nominal case ending is not deleted are stated in the following rule:

2.4.83 *na avyayībhāvāt ataḥ am tu apañcamyāḥ*: A case ending occurring after an *avyayībhāva* compound is not deleted when the compound ends in '*a*', instead the case ending is replaced by '*am*' except when it is in ablative (*pañcamī*).

The generation of *avyayībhāva* compounds can be implemented by adding two phases. The first phase corresponds to the possible left-hand sides which can be listed as a finite set of words, corresponding to the *sūtra*s 2.1.5 to 2.1.19, such as {*adhi, pari, anu, antar, upa, su, dur, nir, ati, iti, tat, prati, yathā, sa, yāvat, apa, bahir, prāk, ā, abhi, pāre, madhye*} and the set of numbers. The second phase corresponds to the forms of the second component, which must be constructed from ordinary *prātipadika*s by various changes, as described in the grammar. If the *prātipadika* ends in a long vowel, it is changed to the corresponding short vowel (neuter stem). Some other stem changes are described in the rules 5.4.107 to 5.4.112. For example:

5.4.107 *avyayībhāve śaratprabhṛtibhyaḥ*: For the *prātipadika*s listed in the *śaradādigaṇa*, the suffix *ṭac* is added. Thus, *śarad* can be stored as *śaradam*, obtained by the following sequence:

śarad + *ṭac* $\xrightarrow{5.4.107}$ *śarada* $\xrightarrow{2.4.83}$ *śaradam*

5.4.111 *jhayaḥ*: For the *prātipadika*s ending in a consonant belonging to *pratyāhāra jhay* (the first four letters of the sound classes), a suffix *ṭac* (*a*) is optionally added. Thus, the right hand side corresponding to *sarit* can be stored as two forms {*sarit, saritam*}, where *saritam* is obtained by the same process as described above for *śaradam*.

In effect, the forms of the second component correspond to the following 2-stage method:

---

[4] 2.4.18 *avyayībhāvaḥ ca* and 1.2.47 *hrasvaḥ napuṃsake prātipadikasya*

[5] 2.1.5 *avyayībhāvaḥ*

[6] 2.4.82 *avyayāt āpsupaḥ*

1. Stem changes from long to short and other changes as prescribed in the rules 5.4.107 to 5.4.112.
2. If the stem obtained from the first step ends in '*a*', add the suffix '*am*' to it.

3.4 Intra-compound sandhi

Finally, we have to make sure that the components of a compound are glued together according to the sandhi rules that are assumed by our transition transducer. In general, external sandhi is used. However, it may happen that retroflexion occurs, as in *rāmāyaṇa*, which is therefore not obtainable by external sandhi from its components *rāma* and *ayana*. We shall assume that such frozen forms, usually occurring only in proper names, are not the result of a productive process, and can be obtained by proper lexicalization. But another class of compounds, called *prādi* compounds, take retroflexion regularly. This is specially frequent with participles prefixed with preverbs, but also with other primary substantival formations (*kṛdantas*) from roots. In order to deal with those, we have to introduce new data banks of participial forms, accessible from yet another copy of preverb sequences, with retroflexion allowed on the corresponding transition, as shown in the more complete state diagram of Figure 2 below. This extension demanded an extension of our morphological generator, which had to provide derivational morphology and not just flexional morphology. The *aṣṭādhyāyī* treatment of the retroflexion across word and compound boundaries has been dealt with exclusively in section 4.4 below.

Such intricate problems of compounding have already been discussed at length in other papers. We shall concentrate in the following on the justification that the morpho-phonetic devices used in our parser are indeed consistent, from both points of view of correctness and of completeness, with the rules defined in the *Aṣṭādhyāyī*, specially in view of their different order of application.

4 Completeness of the segmenter

4.1 Basic Zen technology

The Heritage platform is based on a functional toolkit for computational linguistics called Zen [8]. This technology builds on a central applicative data-structure of *tries* or lexical trees. Tries are a commonly used data structure for representing lexicons. One may see a trie as a deterministic finite state automaton recognizing the lexicon as a regular set of words. The maximal sharing of this data structure obtained by representing the underlying tree as a dag (directed acyclic graph) yields the corresponding minimal automaton. The Zen toolkit generalizes this observation to the case of general (i.e. possibly infinite) regular sets by representing the control graph of finite state machines such as finite automata and transducers as an annotated trie, amenable to optimum sharing as well. Furthermore, automata and transducers are seen as specific instantiations of very general relational abstract machines, called effective Eilenberg machines, admitting uniform simulators [14].

The first real-scale application of this Zen technology was to show how to solve efficiently the inverse problem of external sandhi [10], represented as a *juncture rewrite system*. Let us recall briefly the key notions.

We model external sandhi with rewrite rules of the form $u|v \to w$, where u, v and w are words (standing for strings of phonemes). Such a rule represents the rational relation that holds between all pairs of strings (from now on we use strings and words interchangeably) $\lambda u|v\rho$ and $\lambda w\rho$, for λ and ρ any strings. The symbol | stands for word juncture. The sandhi problem may then be posed as a regular expression problem, namely the correspondance between $(L \cdot |)^*$ and Σ^* by relation $\mathcal{R}$, where Σ is the word alphabet (our extended phoneme alphabet), L is the set of inflected forms generated from the lexicon, and $\mathcal{R}$ is the rational relation that is the concatenation closure of the union of the rational relations corresponding to the sandhi rules. This presentation is a standard one since the classic work of Kaplan and Kay [17].

Note that the sandhi problem is expressed in a symmetric way. Going from $z_1|z_2|...z_n| \in (L \cdot |)^*$ to $s \in \Sigma^*$ is generating a correct enunciation s with word forms $z_1, z_2, ...z_n$, using the sandhi transformations. Whereas going the other way means analysing the sentence s as a possible phonemic stream using words from the lexicon as amalgamated by sandhi. It is this second problem (*sandhi viccheda*) we are interested in solving, since sandhi, while basically deterministic in generation, is strongly ambiguous in analysis.

The article [10] explains how to use the Zen toolkit to perform this analysis automatically. It then goes on to give a formal justification of the methodology, under two assumptions, called *strictness* and *non-overlapping*. In a nutshell, the algorithm is sound (all outputs are plausible solutions in the sense of yielding the input sentence by sandhi), complete (all such solutions are enumerated) and terminating (the set of solutions is finite, and the enumeration algorithm always terminates). Let us now examine the correctness assumptions. The strictness condition is simply that in every sandhi rule $u|v \to w$ the right hand side word w is not empty, and this is trivially satisfied. The non-overlapping condition involves both the lexicon L component and the rewrite rule R component. It expresses that there can be overlap of juncture rules of R within one word of L. We say that there is overlapping of rules $u|v \to w$ and $u'|v' \to w'$ within word $z \in L$ whenever the right hand side v of the first rule, initial substring of z, overlaps with the terminal substring of z identical with the left hand side u' of the second rule: $|z| < |v| + |u'|$. In other words, the first rule potentially *bleeds* the second one (in the sense of blocking its application). Since rules of external sandhi are very local ($|v| = 1$, and $|u| = 1$ or u=$a\underline{h}$ or u=$\bar{a}\underline{h}$), such overlapping can occur only for very short words in L, of length at most 2. Furthermore, since neither $a\underline{h}$ nor $\bar{a}\underline{h}$ are legal autonomous forms, this leaves us to consider only the rare words of length 1. There are basically two such words to consider in Classical Sanskrit, besides the Vedic exclamation u which is not used in an autonomous manner in Classical Sanskrit, but only in combination with other particles, marking a hiatus situation not amenable to sandhi (*pragṛhya*). These

two words are the preverb/preposition ā, and the negation prefix a. We shall deal with these two mono-phonemic words in the following two sections.

4.2 Phantom phonemes

We are dealing in this section with sandhi interference in the case of the ā preverb. Let us start with an example, the simple imperative sentence "come here". The sentence is built up from the word *iha* (here) and the imperative form *ihi* of root *i* (to go), prefixed by preverb ā. If we build the imperative form *ehi* (come), and consider the sequence *iha ehi*, we would obtain by sandhi the wrong **ihaihi*. The mistake comes from a too hasty glueing of the preverb to the root form, in a situation where sandhi is indeed not associative, precisely because of overlapping. We should first glue *iha* and ā, getting *ihā*. Now glueing *ihā* and *ihi* by sandhi leads to the correct sentence *ihehi*. This example shows that the terminology "preverb" is actually misleading. Preverbs are just particles which have shifted from the Vedic language to the Classic one from a role of postposition to a preferred position of prefix to root and primary derivatives (*kṛdantas*). But they have kept enough autonomy as individual words to interact with their predecessor, in left-to-right phonetic interaction, before the resulting utterance interacts with the root form. This vindicates our choice not to generate pre-cooked forms obtained by glueing preverbs to root forms, but rather to consider (sequences of) preverbs as an autonomous phase in lexical analysis. This has the added interest to keep only one root form, instead of all the verb forms with preverbs attached.

Still, some special mechanism is required, in order to circumvent the violation of the Zen segmenter non-overlapping requirement in case of preverb ā. This mechanism uses so-called *phantom phonemes*, and was first described in [9, 11]. The idea is to pre-cook special forms, encoding the ā-prefixed root forms. Such forms are needed only for root forms starting with a vowel, say *v*. We add to such a form *v*-*w* a special form **v*-*w*, with **v* a new symbol, called the *phantom phoneme* of the *varṇa* of vowel *v*. Phantom phonemes obey new sandhi rules, which are coined to simulate the left-to-right application of sandhi, should preverb ā have occurred before the standard form *v*-*w*. We have 4 such symbols, noted respectively **a*, **e*, **o* and **r*, for the 4 vowel *varṇas*. Thus **a* replaces *a* or ā, **e* replaces *i* or ī, **o* replaces *u* or ū, **r* replaces *ṛ* or *ṝ*. The new rules of sandhi are all instances of $u|*v \rightarrow w$, for u any vowel, $*v$ the phantom of vowel v, w the result of sandhi of z and v, for z the result of sandhi of u and ā. For the above example, we add to the bank of root forms the special form **ehi*, and we analyse *ihehi* as the glueing of *iha* with **ehi*, using special sandhi rule $a|*e \rightarrow e$.

Correspondence with *Aṣṭādhyāyī* Treatment

In terms of *Pāṇinian* derivation, the concept of phantom phonemes can be shown to be in correspondence with the following rules:

6.1.85 *antādivat ca*: A replacement in the place of the preceding and following sound segments in *saṃhitā* is treated as the final of what precedes and initial of what follows.

6.1.95 *omāṅoḥ ca*: When a vowel '*a*' is followed by the '*ā*' of '*āṅ*' or '*o*' of '*om*', the second is a replacement in place of both.

So, when we reach at the stage *iha + ehi* in the *Pāṇinian* derivation, because of the metarule 6.1.85, the second word is 'as if starting with '*ā*' of '*āṅ*' and therefore, 6.1.95 finds its scope and replaces the '*a*' of '*iha*' and '*e*' of '*ehi*' by the second (which is '*e*'), giving the desired form *ihehi*. If we try to make a correspondence between the *Pāṇinian* derivation and phantom phonemes, we can think of these phantom phonemes as an indication that these have been obtained because of sandhi between $\bar{a}$ and another vowel and therefore, will follow the sandhi rule 6.1.95. In general, if we have a sandhi rule such that:
[$\bar{a}$ + Vowel x = Vowel y],
we can replace the Vowel y with a phantom phoneme *y which follows the following sandhi rule after the vowel *a*:
[*a* + *y = y].
Replacing y by *y corresponds to the metarule 6.1.85, where the phonetic form is preserved but an indication is provided that it comes from the element $\bar{a}$, while the sandhi rule '*a* + *y = y' corresponds to 6.1.95, where the two phonemes are replaced by the second.

4.3 Privative compounds

Treatment of privative compounds in the Heritage platform

In this section we shall examine possible overlappings with the privative particle *a*. First of all, we emphasize that the morpheme *a* used for augment forms of the past tenses such as imperfect or aorist is not to be considered here, since it applies only in the generative morphology of root forms, which is assumed to be done in a previous phase (the construction of the Verb database of forms).[7]

It turns out that the privative particle does not lead to an interference situation, due to its dual nature; indeed, it is marked by the *pada* '*a*' in front of consonants, but by the *pada* '*an*' in front of vowels. Since '*a*' combines with a preceding phoneme either to form a syllable ended in '*a*' in case of a consonant, and to a vowel otherwise, it is clear that no sandhi overlap of the result may occur with the following *prātipadika*. Note that the alliteration nasal '*n*' forbids the analysis of phoneme '*ā*' as an arbitrary succession of negations, should '*a*' alone be cumulative, leading to hopelessly ambiguous statements.

Thus it is enough to give the analysis of privative compounds by splitting the Noun database into two subsets Nounc (starting with a consonant) and Nounv

[7] Indeed, if Sanskrit analysis was attempted without this initial generative phase, the inverting of augment would be a major hurdle, as well as retroflexion analysis; we do not believe such analysis, inverting Pāṇini's grammar rules fully down to root stems, to be computationally tractable.

(starting with a vowel), and by having distinct phases A or An, corresponding to unit databases representing the two respective strings '*a*' or '*an*'. The state space of the segmenter links phases A to Nounc and An to Nounv. A similar duplication is needed for the Iic phase, as well as for the Voc phase of vocative forms. This leads to a significant increase of the number of phases, as witnessed by Figure 2 below, but no essential complication.

It should be emphasized, though, that the duplication of some phase does not lead to a duplication of the corresponding database. It is straightforward to share Noun, Nounc and Nounv by only duplicating the first layer of the corresponding dags, and profiting of the fact that vowels are listed before consonants, leading to more sharing. There is thus very little increase of the space requirements of these databases, in good adequacy with the regime of economy (*lāghava*) of the Zen machinery.

Treatment of privative compounds in the *Aṣṭādhyāyī*

The *nañ* compounds correspond to the following rules in Panini's grammar:

6.3.73 *nalopaḥ nañaḥ*: In a compound, when *nañ* is followed by another word, the '*n*' of *nañ* is deleted.

6.3.74 *tasmāt nuṭ aci*: In a compound, if *nañ* is followed by a word starting with a vowel, the augment *nuṭ* is inserted after the *n* has been deleted.

Together, these two rules imply that *nañ* is replaced by *a* and *an* if the following word in the compound starts in a consonant and vowel respectively. In the Heritage system, *a* and *an* are denoted by two distinct phases A and An and the noun database is split into Nounc and Nounv for the nouns starting with a consonant and the nouns starting with a vowel, respectively. The phase A is linked to Nounc (6.3.73) and An is linked to Nounv (6.3.74).

4.4 Analysis of retroflexion across word and compound boundaries

The retroflexion rules in Pāṇini's grammar appear in the *tripādī* section and as the other rules in this section, are applied as phonetic smoothing of the final forms. The retroflexion rules can be divided into two categories[8]:

- Retroflexion rules for *n*→*ṇ*
- Retroflexion rules for *s*→*ṣ*

For a system that only uses word generation mechanism for handling sentences, handling retroflexion rules is not trivial. A parser based on an underlying word generating device works on the principle that the word forms are stored and a compound as well as a pada with external sandhi is analyzed by glueing together the constituent word elements using external sandhi. These external sandhi rules are assumed by the transducer *a priori* and do not include retroflexion rules. So,

[8] Please note that we will not discuss the sandhi rules such as 8.4.41 *ṣṭunā ṣṭuḥ*, which can also cause the changes from *s* →*ṣ* and *n* →*ṇ*.

while the constituent words themselves would have gone through retroflexion if the conditions are present within that word, it would not be the case if the condition for retroflexion is present in another word. Let us go back to the example *rāmāyaṇa*. The two constituent words forming this compound are *rāma* and *ayaṇa* (*ayana*, the independent entry). A parser based on an underlying word generating device would have problems recognizing the term *ayaṇa* because the independent word *ayana* does not have such retroflexion and the condition for retroflexion in *ayana* is triggered by the preceding segment *rāma*.

We shall look into this issue by analyzing the two cases of retroflexion separately:

Retroflexion rules for $n \rightarrow ṇ$

The main rules for this retroflexion in *Pāṇini*'s grammar are the following:

8.4.1 *raṣābhyāṃ naḥ ṇaḥ samānapade*: A replacement in '*ṇ*' comes in place of '*n*' when '*n*' occurs in close proximity, preceded by '*r*' or '*ṣ*' in the same word.

8.4.2 *aṭkupvāṅnumvyavāye api*: The above replacement takes place even if intervened by any phoneme in the set $\{aṭ^{9}, ku^{10}, pu^{11}, āṅ, num\}$.

The main rule 8.4.1 uses the term '*samānapade*' (in the same word), which implies that this retroflexion can be applied in a single pada only. It clearly overrides the possibility of retroflexion in the case of external sandhi across word boundaries. Thus '*agnir + nayati = agnirnayati*'[12]. This settles the problem of retroflexion in the case of external sandhi. However, *Pāṇini* lists many rules for

[9] $aṭ$ = Vowels $\cup$ {*h, y, v, r*}

[10] ku = {*k, kh, g, gh, ṅ*}

[11] pu = {*p, ph, b, bh, m*}

[12] This example is interesting because of the fact that the generative model of *Aṣṭādhyāyī* uses the context from the complete sentence, in contrast with a generative model that first generates the separate final forms and then, glues these together to generate the sentence. For this specific case of *agnirnayati*, the separate final forms in a word generating device would be *agniḥ* and *nayati*. However, there is no rule in *tripādī* which can give the following desired form *agnirnayati*, starting with the input *agniḥ + nayati*, i.e.

agniḥ + nayati $\not\rightarrow$ *agnirnayati*

In this example, the final form stored by a word generating device will not give the correct form in external sandhi. However, it is to be noted that the only instances, which can lead to this problem are the rules which have a specific condition of *avasāna*. A pada in context of other words does not get this condition unless it is the final pada of a sentence. There are two rules in *tripādī* with the condition of *avasāna*:

8.3.15 *kharavasānayoḥ visarjanīyaḥ*: The phoneme '*r*' is changed to '*ḥ*' if followed by a phoneme in *pratyāhāra khar* or in *avasāna*.

8.4.56 *vā avasāne* : A phoneme belonging to the set '*jhal*' is changed to a phoneme in the set '*car*' optionally if the right context is *avasāna*.

Thus, the rule 8.4.56 changes *vāg* to the optional forms *vāk, vāg* in *avasāna*, which would be stored as the final forms in a word generating device. It might have caused problem if the external sandhi between *vāk + hari* gives a different form that *vāg*

retroflexion across constituent words in a compound formation, specifically rules 8.4.3 to 8.4.13. Let us consider the first of these rules, 8.4.3, here:

8.4.3 *pūrvapadāt sañjñāyām agaḥ*: A replacement '*ṇ*' comes in place of '*n*', when '*n*' occurs in a pada, combined after another pada, containing '*r*' or '*ṣ*' as the condition of replacement, but not containing '*g*', provided the derivate denotes a name.

This rule clearly gives the cases for retroflexion across constituent words (*pada*) in a compound (*samāsa*). Now, questions have been raised in the tradition whether 8.4.3 is an operational rule or a restrictive rule [25], that is, whether the retroflexion across pada was available from the rules 8.4.1 and 8.4.2. Let us describe these two views in brief below:

Operational view: A compound cannot be accepted as a single pada. Therefore, in *dru–nasa*, the retroflexion is not available by the previous rules and 8.4.3 prescribes this operation, making 8.4.3 to be an operational rule.

Restrictive view: Since nominal endings are introduced after a compound nominal stem to derive a single pada, the compound as a whole constitutes a single pada. Therefore retroflexion in compounds was available from the previous rules. The rule 8.4.3 restricts this availability only to the compounds, denoting a name.

Whether we accept the operational view or the restrictive view, it leads to the same interpretation, that is, in the case of compounds, retroflexion across pada occurs only in the case of *sañjñā* (names) such as '*rāmāyaṇa*', '*śūrpaṇakhā*' etc. It suggests that these compounds can actually be lexicalized as frozen expressions.

Apart from 8.4.3, other rules dealing with retroflexion across pada in a compound have a limited scope since the conditions for these rules are the specific lexical entries. For instance:

8.4.4 *vanaṃ puragāmiśrakāsidhrakāsārikākoṭarāgrebhyaḥ*: When '*vana*' occurs in combination after '*r*' or '*ṣ*' contained within a previous pada constituted by '*puragā*', '*sidhrakā*', '*sārikā*', '*koṭarā*' or '*agre*', the '*n*' of *vana* undergoes retroflexion.

$+ hari \xrightarrow{8.4.62} vāgghari$. Thankfully, there is another rule 8.2.39 (*jhalāṃ jaśaḥ ante*), which can convert this final *k* back to *g*. Thus,

$vāk + hari \xrightarrow{8.2.39} vāg + hari \xrightarrow{8.4.62} vāgghari$

gives the same form as $vāg + hari$ and the sandhi rules $\{k + h = ggh,\ g + h = ggh\}$ will give the correct sandhi, in correspondence with *tripādī* rules.

We will now discuss the methodology adopted by the system to handle this situation created by the condition of *avasāna* in 8.3.15. The system stores the final form as to what it would be before applying that particular rule. So, the final form is stored as ending in '*r*' and the following external sandhi rules are applied:

$r + khar \xrightarrow{8.3.15,8.3.34} s + khar$

r + All phonemes $-khar \rightarrow$ no change

r (end of pada) $\xrightarrow{8.3.15}$ *ḥ* (end of pada)

8.4.8 *vāhanam āhitāt*: When '*vāhana*' occurs in combination after '*r*' or '*ṣ*' contained within a previous pada constituted by a word signifying 'that which is carried', the '*n*' of '*vāhana*' undergoes retroflexion.

Similarly, other rules in this section also have a limited scope and these cases can be lexicalized. However, there is another rule in this section, which is general enough:

8.4.11 *prātipadikāntanumvibhaktiṣu ca*: A replacement in '*ṇ*' comes optionally in place of '*n*', which either occurs at the end of a nominal stem, or occurs as a part of '*num*' or '*vibhakti*', if preceded by a pada containing '*r*' or '*ṣ*' as the condition of replacement. For example,

māṣa–vāpin + Nominative dual → *māṣavāpiṇau* ('*n*' occurs at the end of the nominal stem *vāpin*)

māṣa–vāpa + Nominative plural → *māṣavāpāṇi* ('*n*' of *num*)

māṣa–vāpa + Instrumental singular → *māṣavāpeṇa* ('*n*' of *vibhakti*)

Rule 8.4.11 is a general rule, not specific to the proper names only and therefore, it is not clear whether lexicalizing such occurrences would suffice. This rule, however is an optional rule and it needs to be verified that there are sufficient corpus occurrences of this specific case of retroflexion. To access the number of compounds which may fall in this category, that is, satisfy the conditions of 8.4.11, we gathered some statistics from the attested compounds in the Monier Williams (MW) [21] dictionary. The compounds satisfying the conditions for 8.4.11 can fall in the following two categories:

- The compounds ending in '*n*', where the left constituent of the compound may trigger the retroflexion in the right constituent: There are 57 such compounds attested in MW. Example, *agra–gāmin* might decline as {*agra–gāminau*, *agra–gāmiṇau*}.
- The compounds where there is no '*n*' in the right constituent but the left constituent may trigger the retroflexion when the compound word declines: There are 509 such compounds, attested in MW. Example, *śāra–bhūmi* might decline as {*śāra–bhūminā*, *śāra–bhūmiṇā*}.

We are now left with the question whether the optional retroflexed forms allowed by 8.4.11 actually appear in the corpus and if they do, how many such forms are there? An analysis into corpus statistics is needed for all these forms[13]. As of now, we do not have any statistics regarding the occurrences of such forms with/without retroflexion in a corpus.

Retroflexion rules dealing with preverbs for the cases of *prādi* compounds as well as in a single pada are listed from 8.4.15 to 8.4.35. The main *sūtra* 8.4.14 states:

8.4.14 *upasargād asamāse api ṇopadeśasya*: If a preverb (*upasarga*) contains the '*r*' or '*ṣ*' as the condition of retroflexion, it can cause retroflexion in the following verb if the verb contains *ṇ* in the initial citation (*upadeśa*). Thus *praṇāyakaḥ*.

[13] It is not possible to get these statistics from MW because this retroflexion occurs for the inflected forms.

Retroflexion rules for *s*→*ṣ*

The main rules in *Pāṇini*'s grammar corresponding to this retroflexion case are the following:

8.3.55 *apadāntasya mūrdhanyaḥ*: A replacement in *mūrdhanya* (retroflex) comes in place of that '*s*', which does not occur at the end of a pada.

8.3.57 *iṇkoḥ*: A non-final '*s*' is replaced with '*ṣ*', when the same occurs after *iṇ*[14] or *ku*.

8.3.58 *numvisarjanīyaśarvyavāye api*: The replacement takes place even if intervened by *num*, *visarjanīya* or *śar*[15].

For this case of retroflexion, *Pāṇini* does not use the term '*samānapade*'. Therefore, we need to consider both the word and compound boundary cases in our analysis. However, this retroflexion has a different property. The number of intervening phonemes can at most be 1 and therefore, it limits the cases where this retroflexion can occur across words. We will enumerate all the possibilities where the condition for this retroflexion may be obtained across the word boundaries. Before going into that, we define the following sets:

C_s: The set of phonemes that trigger this retroflexion: (8.3.57: *iṇ* ∪ *ku*)

I_s: The set of phonemes that can intervene between any phoneme $p \in C_s$ and 's', which is to be retroflexed: (8.3.58: *n, ḥ, ś, ṣ, s*)

Let us denote the left and right constituents under consideration by P_1 and P_2 respectively. We are looking for the cases where a phoneme in P_1 can trigger retroflexion in P_2. We will have the following cases:

Case 1: $p \in C_s$, $q \in I_s$ and P_1 ends in 'pq', P_2 starts with an '*s*'.

8.3.111 *sātpadādyoḥ* states that if the '*s*' occurs in the beginning of a word, it will not undergo retroflexion. In this case, '*s*' occurs in the beginning of P_2 and therefore, this case is overruled by the grammar.

Case 2: $p \in C_s$, P_1 ends in *p*.

As per 8.3.111, if P_2 starts with an '*s*', it will not undergo retroflexion. So, for this case to apply, '*s*' has to be the 2^{nd} phoneme of the word P_2 and P_2 should start with a phoneme $q \in I_s$. Clearly, it requires that P_2 should start with a conjunct consonant from the set {*ns, ḥs, śs, ṣs, ss*}. The language does not allow such words, starting with this pattern. Internal sandhi rules explicitly eliminate the last 4, and general sonority hierarchy rules out the first in syllable structure.

Case 3: So, we are left only with the case where the first phoneme *p* of P_2 is not in the set C_s and by external sandhi with P_1, changes to the set C_s. Also, the second phoneme of P_2 should be '*s*' after external sandhi of P_1 and P_2, so that the first phoneme $p \in C_s$ of P_2 will provide a condition for retroflexion in the second phoneme '*s*'. Let us now look into this case in further details, taking the two exhaustive cases, where the first phoneme of P_2 did not belong to C_s and external sandhi changed it into a phoneme in C_s, either consonant or vowel.

Let us consider the first possibility. Due to the language constraints, the consonant $p \in C_s$ has to be '*k*' and only '*k*', because other consonants in set C_s

[14] *iṇ* = {Vowels - {*a*, *ā*}} ∪ {*h, y, v, r, l*}

[15] *śar* = {*ś, ṣ, s*}

do not occur before 's' or 'ṣ', while starting a word. Let us consider the rules which can change a consonant $q \notin C_s$ to '*k*'.

8.2.30. *coḥ kuḥ*: A phoneme in the set *cu*={*c,ch,j,jh,ñ*} will be changed to the corresponding phoneme in the set *ku*={*k,kh,g,gh,ṅ*}, if a phoneme in *pratyāhāra jhal* follows. Though '*s*' belongs to the *pratyāhāra jhal*, this rule would not qualify for our analysis since this rule does not look at the phoneme preceding '*cu*' and therefore, would have been applied on the original word itself and the original word would have the retroflexion.

The same reasoning can be applied to the following rule as well:

8.2.41. *ṣaḍhoḥ kaḥ si*: A phoneme in the set {*ṣ, ḍh*} will be changed to '*k*' if '*s*' follows.

So, we will never have a case where external sandhi changes a consonant $q \notin C_s$ to a consonant $p \in C_s$, such that p is the first phoneme and '*s*' is the second phoneme of P_2 after external sandhi.

Let us now consider the second possibility, which is common enough and the language does not constrain the vowels to be used before '*s*'. However, the only possible cases here are if any vowel q in the set A={*a*, *ā*} is changed to a vowel $p \in C_s$ by external sandhi. Such a case arises. Consider the following example:

ka.h + *asicat* → *ka u* + *asicat* (6.1.109 *ato roraplutādaplute*)

ka u + *asicat* → *ko* + *asicat* (6.1.84 *ādguṇaḥ*)

ko + *asicat* → *kosicat* (6.1.105 *eṅaḥ padāntādati*)

At this stage, the '*o*' could have triggered the retroflexion in '*s*'. However, this application is prevented because of the following rule:

6.1.86 (*ṣatvatukorasiddhaḥ*): A single replacement of two sound elements is suspended when a replacement in '*ṣ*' or insertion of '*tuk*' is to take place. Therefore, this case is prevented by the grammar itself.

We have now exhausted all the possibilities and from this discussion, we can safely say that the general case for *s*→*ṣ* retroflexion is that the retroflexion does not happen across pada, even though the term *samānapade* has not been used by *Pāṇini* for this case. There are certain rules, allowing retroflexion across the constituent words in a compound. These rules are treated as exceptions of the rule 8.3.111. Some of the examples are:

aṅguli – *saṇga* → *aṅguliṣaṅga* (8.3.80 *samāse aṅguleḥ saṅgaḥ*)

agni – *stoma* → *agniṣṭoma* (8.3.82 *agneḥ stutstomasomāḥ*)

mātṛ – *svasṛ* → *mātṛṣvasṛ* (8.3.84 *mātṛpitṛbhyāṃ svasā*)

These examples provide a finite list, which can easily be lexicalized. Retroflexion rules corresponding to the compounds are also discussed in 8.3.65 to 8.3.77. In tradition, these rules are also treated as exceptions of 8.3.111. In this section, there are specific rules for various verbs and preverbs and each pair of preverb and verb is to be handled separately.

While we are discussing retroflexion, we will also like to draw the attention of the readers to a specific case of non-retroflexion in the case of *kurvanti*.

The case of '*kurvanti*'

A Sanskrit student who does not know *Pāṇini*'s grammar by heart may find it difficult to give a quick answer as to why there is no retroflexion in '*kurvanti*'? As stated before, the standard rules for retroflexion *n*→*ṇ* in *Pāṇini* are 8.4.1 and 8.4.2. The word kurvanti has an '*n*' preceded by '*r*' in a single word (8.4.1) and the intervening phonemes '*a*' and '*v*' fall in the set '*aṃ*', matching the criteria of 8.4.2. Therefore, it appears as if the form '*kurvanti*' should undergo retroflexion. But it does not, the reason being two other rules stated in *Pāṇini*'s grammar.

8.3.24 *naḥ ca apadāntasya jhali*: If '*n*' is followed by a consonant in *pratyāhāra jhal*, it is replaced by *ṃ*.

8.4.58 *anusvārasya yayi parasavarṇaḥ*: If *ṃ* is followed by a consonant in *pratyāhāra yay*, it is replaced by the homogeneous nasal.

Consider the set T = Consonants - $\{\{\dot{n}, \tilde{n}, \dot{n}, n, m\} \cup \{y, v, r\}\}$, which denotes the phonemes in the *pratyāhāra* set *jhal*. If '*n*' is followed by a phoneme from the set T, the rule 8.3.24 gets the application, bleeding the retroflexion rules. For the set $H = T - \{h, ś, ṣ, s\}$, the rule 8.3.24 feeds the rule 8.4.58. Specifically, in the case under consideration,

kurva(n)ti $\xrightarrow{8.3.24}$ *kurva(ṃ)ti* $\xrightarrow{8.4.58}$ *kurva(n)ti*

So, effectively no change took place. On the other hand, if it has to undergo retroflexion, the process would be:

kurvanti $\xrightarrow{8.4.2}$ *kurvaṇti*

The second process does not correspond to the desired form *kurvanti*. There is no rule in *Pāṇini*'s grammar that allows converting the form *kurvaṇti* to *kurvanti*.

What prevents one from following this wrong process is the control structure of *tripādī*. The rules in *tripādī* are to be applied sequentially. The rule 8.3.24 appears before 8.4.2 and therefore, should be applied before 8.4.2. Once the form is *kurvaṃti* by the application of 8.3.24, 8.4.2 does not get the application (bleeding by 8.3.24) and the process directly goes to 8.4.58 (feeding by 8.3.24), which converts *kurvaṃti* back to *kurvanti* by changing '*ṃ*' to the homonasal of '*d*', which is '*n*' in this case.[16]

Now, let us have a look at how this process is implemented in the Heritage site. In addition to coding the rules 8.4.1 and 8.4.2, an additional criteria is employed for retroflexion. As per this criteria, the right hand side of '*n*' is checked and only if it does not contain any consonant in *pratyāhāra jhal*, it can undergo retroflexion. Clearly, the retroflexion process is not exactly Paninian and if asked to enumerate the rules leading to the form '*kurvanti*' in the Heritage system, it will definitely not list the rules 8.3.24 and 8.4.58, which actually cancel the effect of each other. However, it might be looked as a much more complete rule for the retroflexion condition.

[16] We thank Malhar Kulkarni for his explanation of this intricate Paninian process.

5 Idiosyncracies of phonology

5.1 Phonemic variations

The Heritage engine computes on words and sentences represented as phonemic lists, with phonemes represented as integers, in the simplistic but efficient style of the Zen library. The user interface interprets user input in four common transliteration styles, and immediately converts it to normalized words, where normalization, explained in footnote 2, replaces (non genuine) *anusvāra* by the equivalent nasal. All computed forms and morphemes are stored in this normalized form, avoiding useless non determinism at segmenting time.

One important remark is that these stored forms are generally in terminal sandhi form, *except* that final '*r*' is preserved, and *not* turned into *visarga*. This is necessary to parse phrases such as *punarapi*, analysable as the sandhi of *punar* and *api*. Storing *punaḥ* instead would prevent this analysis, and allow the wrong **punopi*.

Gemination, allowing e.g. *karmma* for *karma*, is not allowed, although allowed (optionally) by 8.4.46[17]. In that we follow the opinion of Śākalya[18]. Degemination, for instance of *vārtā* for *vārttā*, is allowed for a few lexical items[19].

The data structure of *pratyāhāra* is not used in the Heritage machinery. The corresponding sets of phonemes are represented by unions of integer intervals.

5.2 Finer phonemic distinctions

The Heritage machinery makes finer phonemic distinctions than the tradition, in order to simplify some morpho-phonemic treatment. Specifically, we consider 3 versions of phoneme *h* and 2 versions of phoneme *j*.

Variants of phoneme *h*

The standard (cerebral) version of phoneme *h* combines with *t* in sandhi to yield *ḍh*. Thus e.g. root *gāh* admits as past participle *gāḍha*. But sometimes

[17] 8.4.46 *acaḥ rahābhyāṃ dve*: A phoneme in the *pratyāhāra yar*, when occurring in close proximity after a vowel followed by *r* or *h*, is optionally doubled.

[18] 8.4.51 *sarvatra śākalyasya*: In the opinion of Śākalya, there is no doubling.

[19] In *tripādī*, there are two optional rules regarding degemination:

8.4.64 *halaḥ yamāṃ yami lopaḥ*: A phoneme in the *pratyāhāra yam* is deleted optionally if it is preceded by any consonant (*hal*) and is followed by another phoneme in the *pratyāhāra yam*.

8.4.65 *jharaḥ jhari savarṇe*: A phoneme in the *pratyāhāra jhar* is deleted optionally if it is preceded by any consonant and is followed by a homogeneous phoneme in the *pratyāhāra jhar*.

As an interesting observation, the two *pratyāhāra* sets, *yam* and *jhar* are exclusive and exhaustive to cover all the consonants. Thus, after any consonant, if two homogeneous consonants follow in the close proximity, the first of the two can optionally be deleted by these two rules.

gdh is obtained, like in *dugdha* for root *duh*, whose final *h* has kept a guttural character. And rarely *ddh* is obtained, like in *naddha* for root *nah*, whose final *h* has kept a dental character.

These sandhi rules are handled in the Heritage site by a finer phoneme distinction than in the traditional view, with extra phonemes *h'* and *h"*, subject to the following sandhi rules:
h'+*t* → *gdh* and *h"*+*t* → *ddh* (and other similar rules for consonants other than '*t*').

As per Pāṇini's grammar, the general rule for sandhi between '*h*' and any phoneme in *jhal* is

8.2.31 *haḥ ḍhaḥ* : '*h*' is replaced by '*ḍh*' if any phoneme in *jhal* follows. For the specific case of *h* followed by *t*, this rule feeds some other rules, yielding the following derivation sequence:

$h + t \xrightarrow{8.2.31} ḍh + t \xrightarrow{8.2.40}$[20] $ḍh + dh \xrightarrow{8.4.41} ḍh + ḍh \xrightarrow{8.3.13}$[21] $ḍh$

However, there are rules, exception to 8.2.31, which in addition to looking at the phonemes, also look for the lexical criteria such as the actual root word and the sandhi rule for the '*h*' appearing in such roots is different. For example:

8.2.32 *dādeḥ dhātoḥ ghaḥ*: For the roots starting with '*d*', '*h*' is replaced by '*gh*' if any phoneme in *jhal* follows. Roots such as {*dah, dih, duh*} fall in this category.

8.2.33 *vā druhamuhaṣṇuhaṣṇihām*: For the roots {*druh, muh, ṣṇih, ṣṇuh*}, '*h*' is optionally replaced by '*gh*' if any phoneme in *jhal* follows.

For the specific case of *h* followed by *t* in these roots, these rules can yield the following derivation sequence:

$h + t \xrightarrow{8.2.32|8.2.33} gh + t \xrightarrow{8.2.40} gh + dh \xrightarrow{8.4.53}$[22] $g + dh$

8.2.34 *nahaḥ dhaḥ*: '*h*' of the root *nah* is replaced by '*dh*' if any phoneme in *jhal* follows. For the specific case of *h* followed by *t* in *nah*, this rule can yield the following derivation sequence:

$h + t \xrightarrow{8.2.34} dh + t \xrightarrow{8.2.40} dh + dh \xrightarrow{8.4.53} d + dh$

In the Heritage system, the phoneme *h* corresponding to the rules 8.2.32 and 8.2.33 is denoted by *h'*, while that corresponding to 8.2.34 is denoted by *h"*.

Variants of phoneme *j*

Similarly, phoneme *j* in its usual guttural version combines with *t* in sandhi

[20] 8.2.40 *jhaṣaḥ tathoḥ dhaḥ adhaḥ*: The phonemes *t* and *th* are replaced by *dh*, if they follow after a phoneme in the set *jhaṣ*, with the exception of the root *dhā*.

[21] 8.3.13 *ḍhaḥ ḍhe lopaḥ*: The phoneme *ḍh* is deleted, when followed by *ḍh*.
It is to be noted that the rule 8.3.13 has been applied here after 8.4.41 and therefore, is an exception of the *asiddha* principle. These cases of *āśrayāt siddham* have been discussed in detail by Cardona in [2].

[22] 8.4.53 *jhalāṃ jaś jhaśi*: A phoneme in the set *jhal* is replaced by a phoneme in the set *jaś*, if followed by a phoneme in the set *jhaś*. In effect, this rule replaces a phoneme in the set *jhal* by the third letters, if followed by fourth letters of a group.

to yield *kt*. Thus *yukta* for past participle of root *yuj*. But sometimes it acts similarly to the sibilant *ś*, and yields rather *ṣṭ*. Thus *mṛṣṭa* for past participle of root *mṛj*. The Heritage site recognizes this distinction with a variant *j'* such that *j'*+*t* → *ṣṭ* in order to accommodate forms of roots such as *mṛj*. The roots concerned are *bhrāj*, *mṛj*, *yaj*, *rāj*, *vraj* and *sṛj*.

As per Pāṇini's grammar, the general rule for sandhi between '*j*' and any phoneme in *jhal* is

8.2.30 *coḥ kuḥ*: A phoneme belonging to the set '*cu*' (palatals) is replaced by the corresponding phoneme in the set '*ku*' if any phoneme in *jhal* follows or at the end of a word.

Under specific conditions, this rule can feed another rule in the grammar:

8.4.55 *khari ca*: A phoneme belonging to the set *jhal* is replaced by the corresponding phoneme in set *car*, if followed by a phoneme in set *khar*.

khar = {*kh, ph, ch, ṭh, th, c, ṭ, t, k, p, ś, ṣ, s*}

car = {*c, ṭ, t, k, p, ś, ṣ, s*}

In effect, the rule 8.4.55 changes the second, third and fourth phonemes of the sound set to its first phoneme, if followed by a phoneme in set *khar*. So, in general, when *j* combines with *t*, both the rules 8.2.30 and 8.4.55 apply, yielding the following:

$j + t \xrightarrow{8.2.30} g + t \xrightarrow{8.4.55} k + t$

However, there is an exception to rule 8.2.30:

8.2.36 *vraścabhrasjasṛjamṛjayajarājabhrājachaśāṃ ṣaḥ*: The last phoneme of the roots '*vraśc*', '*bhrasj*', '*sṛj*', '*mṛj*', '*yaj*', '*rāj*', '*bhrāj*' and roots ending in *ch* or *ś* is substituted by *ṣ* if any phoneme in *jhal* follows or at the end of a word.

Focusing on the combination of *j* and *t*, this rule gives an exception for the roots {*bhrasj, sṛj, mṛj, yaj, rāj, bhrāj*} and changes the *j* to *ṣ* instead of *g*, as specified by the general rule 8.2.30. The phoneme *ṣ* before *t* then feeds the following rule:

8.4.41 *ṣṭunā ṣṭuḥ*: A phoneme in the set *stu* ({*s, t, th, d, dh, n*}) is substituted by the corresponding phoneme in the set *ṣṭu* ({*ṣ, t, th, d, dh, n*}), if it occurs before or after a phoneme in the set *ṣṭu*.

Together, these rules yield the following sandhi rule for these specific roots:

$j + t \xrightarrow{8.2.36} ṣ + t \xrightarrow{8.4.41} ṣ + ṭ$

In the Heritage system, the phoneme *j* corresponding to the rule 8.2.36 is denoted by *j'*.

The variant phonemes which are dealt with in the Heritage engine may be justified by comparative linguistics, as remnants from earlier language substratum. They yield a more unified sandhi processing. The Paninian tradition recognizes them indirectly, by giving specific rules for specific roots, but the end result is equivalent.

6 Soundness and completeness

Evaluation of a computational linguistics platform is usually discussed in terms of the parameters of precision and recall, measuring the quality of response

to a query. For a query admitting rv relevant answers, let rt be the number of retrieved answers returned by the platform, and $right$ be the number of retrieved relevant answers. The precision is defined as $right/rt$, the recall as $right/rv$. The recall measures the completeness of the system. A low recall means that the system performs badly, in not recognizing correct sentences. Increasing the recall usually decreases the precision; we say that the system overgenerates, its correct answers being drowned in irrelevant ones.

We shall not give here precise figures of precision and recall for the Heritage platform, for lack of a statistically significant tagged corpus consistent with our system of tags and our assignment of homonymy indexes. Furthermore, many evaluation criteria may be envisioned, for the generational aspect as well as for the recognition aspect.

We shall however discuss the known incompletenesses of the system with respect to the Paninian framework, and try to assess their statistical relevance.

Before discussing recall, i.e. completeness, we discuss the mechanisms which limit overgeneration.

6.1 Curbing overgeneration

Direct inversion of sandhi on a non-trivial sentence would result in literally millions of potential segmentations, most of them being sterile, with segments un-recognizable as padas. We curb this overgeneration by having our segmenter lexicon-directed - non-compound padas are stored in pre-generated databases of forms arranged in lexical categories, compounds are analysed recursively as sequences of lexicalised forms, and the lexical analyser search is constrained by the phase transitions, which reflect the morphological geometry. Still, certain situations lead to important branching factors of the non-deterministic search. For instance, the form *parā* leads to a potential 164 legal choices, as explained in [10].

In order to curb overgeneration, a semantic filter is applied to the potential segmentations, using a simplified semantic roles (*kāraka*) analysis. This mechanism has been described in [12]. Every segmentation candidate is expanded into the set of all its tagged interpretations. Each tagged interpretation is given a penalty, reflecting the lack of match between the semantic roles of the substantive items and the regime of the verbal items (*sakarmaka/akarmaka*), determining its expectations (*ākāṅkṣā*). In this simplified analysis, only the roles of agent (*kartṛ*) and goal/patient (*karma*) are taken into account. Every segmentation is then assigned as penalty the minimum penalty of its interpretations, and the system truncates all solutions below a given threshold. This semantic filtering reduces drastically the number of returned solutions, with a clear increase in precision, without too much loss in recall.

Two remarks are in order. The first one is that the inner structure of compounds is not relevant to this simplified semantic analysis, since no inner component of a compound may feed direct roles in Sanskrit. Note that the examples of *asamartha* compounds exhibited by Gillon [4, 5] concern only cases where the inner component of a compound has an exocentric argument – as Bhartṛhari

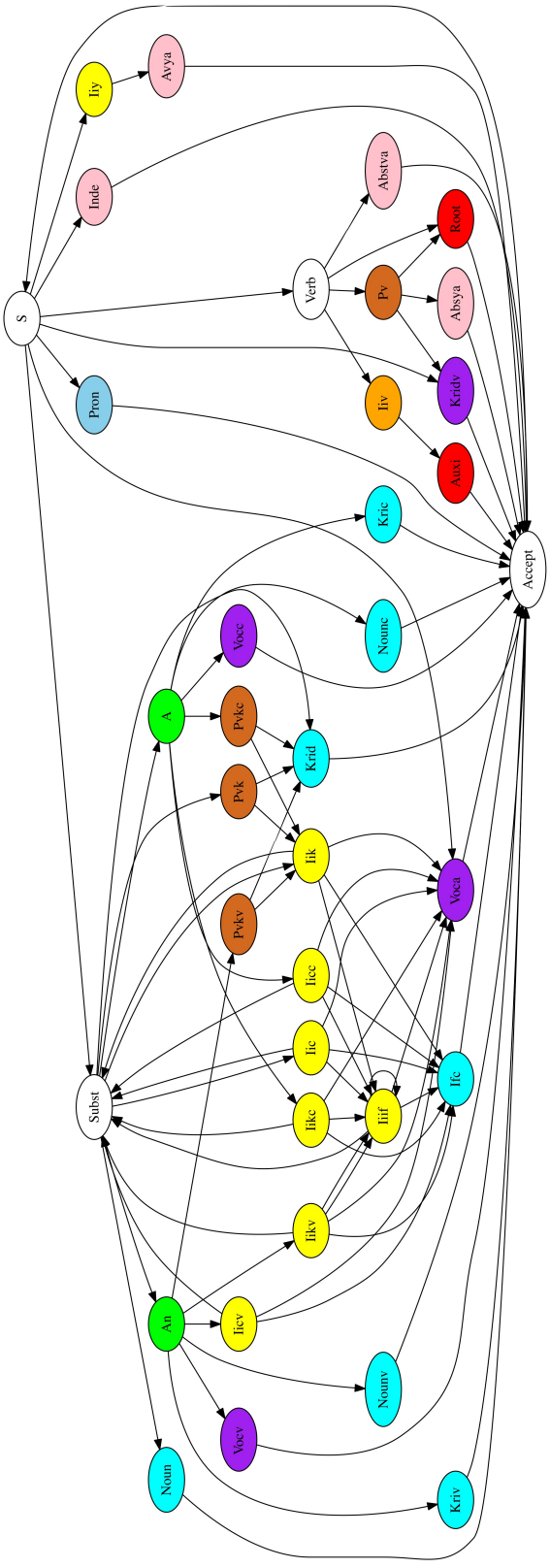

Fig. 2. A more complete lexical analyser.

puts it, this component expresses a relation. This vindicates our flat analysis of pre-compounds, since a pre-compound with $n+1$ components may correspond to C_n possible binary compounds, where the Catalan number C_n is exponential in n, as noted above.

The second remark is that, nonetheless, certain good segmentations may be discarded because of ellipses (typically, the agent may be implicit from the context as the current topic). This means that the *kāraka* analysis ought to be done at the level of discourse, rather than individual sentences.

Another device that curbs overgeneration in the Heritage platform is the regular nature of the segmenting automaton, the phases of the lexical analyser being constrained by a regular expression limiting the recursion. Thus small lexical items such as root substantives (*-pa*, *-ga*, *-nī*, etc.) can generally occur only in Ifc position (right component of a compound), limiting drastically their combinatorics.

Still another important device is that vocatives are allowed only as terminal strings of the various segments. This avoids frequent ambiguities of stems in *-a* as bare stems usable as left components of compounds and as vocative interjections. While Pāṇini does not exclude the vocatives from undergoing sandhi, it is a rare phenomena to obtain the vocatives as a non-final segment and it can be justified by the prosody constraints. It still adds to the incompletenesses of the system, as the systems cannot recognize the vocatives which appear as a non-final segment[23].

Finally, two distinct modes of the Sanskrit Reader are offered to the user. The so-called Simplified mode uses the simple state diagram shown in Figure 1. It forbids vocatives, and does not use the productive devices of participle generation and privative compounds - such stems must be lexicalized in order to be recognized. The simplified mode is adequate for simple sentences, and has very good precision (often, a unique segmentation is proposed). For more complex sentences, the Complete mode, using the full databank of participles and the complete analysis of privative compounds, is necessary. Let us now indicate in which ways this "Complete" mode is still incomplete.

6.2 Known incompletenesses

One basic limitation of the Sanskrit Heritage engine is that it relies on lexicalisation of atomic items. At the time of writing, its lexicon lists 588 roots and 9200 non-compound stems. 1560 combinations of preverbs and roots provide as many

[23] In the *Bhagavad Gītā* (Bh.G.), for instance, there are at least 3 verses where the vocatives appear as a non-final segment:

- Bh.G. 3.22: *na me* ***pārthāsti*** *kartavyaṃ triṣu lokeṣu kiñcana*
- Bh.G. 3.28: *tattvavit tu* ***mahābāho*** *guṇakarmavibhāgayoḥ*
- Bh.G. 5.3 : *nirdvandvo hi* ***mahābāho*** *sukhaṃ bandhāt pramucyate*

The segments containing the non-final vocatives are shown in bold. The first verse also gives an example '*pārthāsti*', where the vocatives can undergo sandhi.

verbs, each allowed in all finite conjugation paradigms available in Classical Sanskrit (present, imperfect, optative, imperative, future, perfect, aorist/injunctive), plus infinitive and absolutive (with transitions allowing absolutives in *-tvā* only from roots, and those in *-ya* only when a preverb is present). 10 forms of participles are generated. Secondary conjugation stems (causative, desiderative and intensive) are lexicalized on demand. Other primary derivatives (*kṛdantas*) are available only when lexicalized. The same is true of secondary derivatives, which is certainly a strong limitation, that will have to be lifted at some future point, since many *taddhita* suffixes are very productive (*-tā*, *-tva*, *-tṛ*, etc.)

A few lexical items have been omitted from the lexicon because they caused too much overgeneration. For instance, the substantive *vi* (bird), for obvious clash with the homophonic particle, or the rare substantive *atra* (food) for similar confusion with the adverb. Also the substantive *āya* for its conflict with dative forms, as well as the form *nā* (nominative of *nṛ*). It is hoped that such difficulties will be overcome at some stage, when the system will be tuned by statistical training.

The analysis given in section 3 above ought to convince the reader that most productive schemes are correctly dealt with. Non productive schemes (*aluk*, retroflexion) assume lexicalisation of the corresponding compound stem.

The strongest limitation concerns substantival verbs, such as *yantrayati*. At present, such verbal stems must be lexicalized. Dealing with substantival verbs in all generality, including for past forms with augment, seems out of reach with the present technology.

Sanskrit is a complex language, whose literature covers many styles over 3 millenia. It has an extremely productive morphology, allowing the generation of extravagant items such as *cicandrīyīṣakāyamāṇena* [3]. Current technology does not permit the automatic analysis of such complex forms. However, compounds with any number of components, such as *pravaranṛpamukuṭamaṇimarīcimañjarīcayacarcitacaraṇayugalaḥ* (attested in Pañcatantra), do not pose particular problems.

7 Conclusion

The Sanskrit Heritage reader recognizes simple sentences from Classical Sanskrit. In Simplified mode (Figure 1 above) it is precise but incomplete. In Complete mode (Figure 2 below) it exhibits a good recall of sentences whose vocabulary is covered by the lexicon. The system is fast and robust. This paper made an attempt at relating its operations to those of the Paninian tradition. Although the details of derivation vary, there seems to be no major discrepancy. The most difficult problems left to solve are the analysis of denominative verbs, and the proper treatment of ellipsed agents and double accusative verbs (*dvikarmaka*).

References

1. G. Cardona. *Pāṇini: his work and its traditions.* Motilal Barnasidass, 1988.
2. G. Cardona. *Pūrvatrāsiddham and āśrayāt siddham.* D. K. Printworld, Delhi, 2011.
3. B. Dīkṣita, S. Vasu, and B. Das Basu. *The Siddhānta kaumudi of Bhattoji Dikshita.* Number v. 3 in The Siddhanta Kaumudi of Bhattoji Dikshita. The Panini office, 1905.
4. B. S. Gillon. Bartṛhari's solution to the problem of asamartha compounds. *Études Asiatiques/Asiatiche Studien*, 47,1:117–133, 1993.
5. B. S. Gillon. *Indian linguistic studies: Festschrift in honour of George Cardona; Eds. Madhav M. Deshpande et Peter E. Hook*, chapter Bartṛhari's rule for unexpressed kārakas: The problem of control in Classical Sanskrit. Motilal Banarsidass, Delhi, 2002.
6. B. S. Gillon. Exocentric (bahuvrīhi) compounds in classical Sanskrit. In G. Huet and A. Kulkarni, editors, *Proceedings, First International Symposium on Sanskrit Computational Linguistics*, pages 1–12, 2007.
7. B. S. Gillon. Tagging classical Sanskrit compounds. In A. Kulkarni and G. Huet, editors, *Sanskrit Computational Linguistics 3*, pages 98–105. Springer-Verlag LNAI 5406, 2009.
8. G. Huet. The Zen computational linguistics toolkit. Technical report, ESSLLI Course Notes, 2002.
9. G. Huet. Towards computational processing of Sanskrit. In *International Conference on Natural Language Processing (ICON)*, 2003.
10. G. Huet. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *J. Functional Programming*, 15,4:573–614, 2005.
11. G. Huet. *Themes and Tasks in Old and Middle Indo-Aryan Linguistics, Eds. Bertil Tikkanen and Heinrich Hettrich*, chapter Lexicon-directed Segmentation and Tagging of Sanskrit, pages 307–325. Motilal Banarsidass, Delhi, 2006.
12. 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*, New York, NY, USA, 2007. ACM.
13. G. Huet. Sanskrit segmentation. XXVIIIth South Asian Languages Analysis Roundatable, University of Denton, Texas, 2009.
14. 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.
15. S. Joshi and J. Roodbergen. *Patañjali's Vyākaraṇa-Mahābhāṣya Sthānivadbhāvāhnika: introduction, text, translation and notes.* Number v. 1 in Research Unit series. Bhandarkar Oriental Research Institute, 1990.
16. S. Joshi and J. Roodbergen. *The Aṣṭādhyāyī of Pāṇini with Translation and Explanatory Notes.* Number v. 11 in The Aṣṭādhyāyī of Pāṇini. Sāhitya Akādemī, 2004.
17. R. M. Kaplan and M. Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20,3:331–378, 1994.
18. F. Kielhorn. *The Paribhashendusekhara of Nagojibhatta.* Parimal Publications, Delhi (reprint), 1871.
19. P. Kiparsky. On the architecture of Pāṇini's grammar. In G. Huet, A. Kulkarni, and P. Scharf, editors, *Sanskrit Computational Linguistics 1 & 2.* Springer-Verlag LNAI 5402, 2009.

20. A. Kulkarni and D. Shukl. Sanskrit morphological analyser: Some issues. *Indian Linguistics*, 70(1-4):169–177, 2009.
21. M. Monier-Williams, E. Leumann, and C. Cappeller. *A Sanskrit-English Dictionary: Etymological And Philologically Arranged With Special Reference To Cognate Indo-European Languages*. Asian Educational Services, 1999.
22. O. Munśi, E. Unithiri, and N. Unithiri. *Dhāturūpaprapañca: (A Dictionary of All the Forms of All the Roots in Sanskrit)*. Number pt. 2 in Calicut University Sanskrit series. Publication Division, University of Calicut, 2004.
23. P. Scharf. Levels in *Pāṇini*'s *Aṣṭādyāyī*. In A. Kulkarni and G. Huet, editors, *Proceedings, Third International Symposium on Sanskrit Computational Linguistics*, volume LNAI 5406, pages 66–77. Springer, 2009.
24. A. Sharma, K. Deshpande, and D. Padhye. *Kāśikā: A Commentary on Pāṇini's Grammar*. Sanskrit Academy series. Sanskrit Academy, Osmania Universtiy, 2008.
25. R. Sharma. *The Aṣṭādhyāyī of Pāṇini: English translation of adhyāyas seven and eight with Sanskrit text, transliteration, word-boundary, anuvṛtti, vṛtti, explanatory notes, derivational history of examples, and indices*. The Aṣṭādhyāyī of Pāṇini. Munshiram Manoharlal Publishers, 2003.
26. S. Vasu. *The Aṣṭādhyāyī of Pāṇini*. Motilal Banarsidass, 1980.
27. O. von Böhtlingk. *Pāṇini's Grammatik*. Language and Linguistics Series. Motilal Banarsidass, 1998.

Appendix A

Let us consider the derivation of future passive participles (pfp) ending in *ya* and we will draw parallel between the Heritage system and the rules in *Aṣṭādhyāyī*.

Aṣṭādhyāyī Treatment

In *Aṣṭādhyāyī*, there are three different *kṛtya* suffixes, which are used for the pfp ending in *ya*: {*yat, kyap, ṇyat*}. The treatment of this pfp formation can be considered a two-stage process of firstly identifying the proper suffix out of the three, and secondly stem changes according to the suffix.

Identifying the proper suffix

Main rules in Pāṇini's grammar corresponding to identifying the pfp suffixes are:

3.1.97 *acaḥ yat*: The suffix *yat* is added to the roots ending in a vowel.

3.1.98 *poḥ adupadhāt*: The suffix *yat* is added to the roots ending in a vowel and having *a* as the penultimate phoneme.

3.1.109 *etistuśāsvṛdṛjuṣaḥ kyap*: The suffix *kyap* is applied to the roots {*i, stu, śās, vṛ, dṛ, juṣ*}.

3.1.110 *ṛdupadhāt ca aklpicṛteḥ*: The suffix *kyap* is applied to the roots having short *ṛ* in the penultimate, with the exception of *klp* and *cṛt*.

3.1.124 *ṛhalor ṇyat*: The suffix ṇyat is applied to the roots ending in *ṛ* or a consonant.

Together, these rules along with some specific rules in 3.1.97 to 3.1.132 determine which of the three suffixes should be applied to a root.

Stem changes

Once the suffix is decided, the rules in the grammar corresponding to the stem changes are applied. For example, **for suffix *yat***:

6.1.45 *āt ecaḥ upadeśe aśiti*: For a root ending with one of the {*e, ai, o, au*} in the *upadeśa*, the last phoneme is substitued by *ā*, if a suffix not having an indicatory *ś* follows.

6.4.65 *īt yati*: If suffix *yat* follows an *aṅga* ending in *ā*, the last *ā* is replaced by *ī*.

Because *yat* is an *ārdhadhātuka* suffix, the *aṅga* gets the *guṇa* by 7.3.84 (*sārvadhātukārdhadhātukayoḥ*).

For suffix *kyap*, the main stem changes occur because of the following rule:

6.1.71 *hrasvasya piti kṛti tuk*: When a short vowel is followed by a *kṛt* suffix having an indicatory *p*, it gets an insertion of *tuk* (*t*).

Because *kyap* has an indicatory *k*, it does not cause *guṇa* because of 1.1.5 *kkṅiti ca*.

For suffix *ṇyat*, the main stem changes occur because of the following rules:

7.2.115 *acaḥ ñṇiti*: An *aṅga* ending in a vowel gets *vṛddhi* before a suffix having an indicatory *ṇ* or an indicatory *ñ*.

7.2.116 *ataḥ upadhāyāḥ*: The penultimate *a* of an *aṅga* gets *vṛddhi* before a suffix having an indicatory *ṇ* or an indicatory *ñ*.

Because *ṇyat* is an *ārdhadhātuka* suffix, when the *aṅga* does not satisfy the above two rules, it gets the *guṇa* by 7.3.86 (*pugantalaghūpadhasya ca*).

After the stem gets *vṛddhi*, there are some phonetic changes because of the following rule:

7.3.52 *cajoḥ ku ghiṇyatoḥ*: The final *c* or *j* of an *aṅga* are replaced by *k* and *g* respectively, if a suffix having an indicatory *gh* or the suffix *ṇyat* follows.

There are some exceptions to 7.3.52, such as

7.3.59 *na kvādeḥ*: The change from *c* or *j* to *k* and *g* does not take place if the root starts with a velar.

7.3.66 *yajayācarucapravacarcaḥ ca*: The change from *c* or *j* to *k* and *g* does not take place for the roots {*yaj, yāc, ruc, pravac, ṛc, tyaj*}. The inclusion of the root *tyaj* is because of a *vārtikā* on this rule.

Heritage Treatment

The Heritage system does not implement pfp as a two-stage process as described above but the rules for identifying suffix and stem changes are implemented together. Given a root, the function **pfp-ya** builds the stem pfp-stem. Once the pfp-stem is built, the condition of phonemic change due to 7.3.52 is checked by a function **fix-pfp-ya** and the specific exceptions of 7.3.52 are encoded in the function **palatal-exceptions**. We will give a brief pseudocode of these three functions along with the reference to Pāṇini's rules below.

pfp-ya (root)

pfp-stem =

*(** if root ends in a vowel **)*

match last phoneme of root with

a →root *(3.1.97)*

ā | *ai* | *o* | *au* →replace last with *e (3.1.97, 6.1.45, 6.4.65, 7.3.84)*

i | *ī* | *u* | *ū* →*guṇa* of root *(3.1.97, 7.3.84)*

ṛ →*vṛddhi* of root *(3.1.124, 7.2.115)*

*(** if root ends in a consonant **)*

match last and penultimate phoneme with

{x,a} →if x ∈ {*p,ph,b,bh,m*} then root *(3.1.98)*

else replace *a* by *ā (3.1.124, 7.2.116)*

{x,ṛ} →root *(3.1.110)*

else →*guṇa* of root *(3.1.124, 7.3.86)*

fix-pfp-ya (root, pfp-stem)

pfp-stem-new =

if palatal-exception (root) →pfp-stem

else if pfp-stem starts with a velar →pfp-stem *(7.3.59)*

else match the last phoneme of pfp-stem with

c →replace *c* with *k (7.3.52)*

j →replace *j* with *g (7.3.52)*

else →pfp-stem

palatal-exception (root)

match root with

aj | *vraj* →True *(7.3.60)*

yaj | *yāc* | *ruc* | *ṛc* | *tyaj* →True *(7.3.66)*

sṛj | *vṛj* | *pṛc* →True *(** because these roots take kyap (3.1.110) **)*

else →False

The function pfp-ya takes care of all the roots, which take *yat* and *ṇyat* suffixes and the roots ending in a consonant, which take *kyap* suffix. The roots which take *kyap* suffix and end in a vowel, are dealt with specifically in a separate function, which also generates the supplementary pfp forms.

It is clear from the pseudocode that every operation in the Heritage system emulates the corresponding operations in Pāṇini's grammar. A single line of code (phonetic rewrite rule) corresponds to one or many rules (*sūtra*) in *Aṣṭādhyāyī*; as discussed in section 1, these rewrite rules could be used as a guide to the generation of the Paninian derivation.