Shallow syntax analysis in Sanskrit guided by semantic nets constraints

Gerard Huet
INRIA
Rocquencourt, France
Gerard.Huet@inria.fr

ABSTRACT

We present the state of the art of a computational platform for the analysis of classical Sanskrit. The platform comprises modules for phonology, morphology, segmentation and shallow syntax analysis, organized around a structured lexical database. It relies on the Zen toolkit for finite state automata and transducers, which provides data structures and algorithms for the modular construction and execution of finite state machines, in a functional framework.

Some of the layers proceed in bottom-up synthesis mode - for instance, noun and verb morphological modules generate all inflected forms from stems and roots listed in the lexicon. Morphemes are assembled through internal sandhi, and the inflected forms are stored with morphological tags in dictionaries usable for lemmatizing. These dictionaries are then compiled into transducers, implementing the analysis of external sandhi, the phonological process which merges words together by euphony. This provides a tagging segmenter, which analyses a sentence presented as a stream of phonemes and produces a stream of tagged lexical entries, hyperlinked to the lexicon.

The next layer is a syntax analyser, guided by semantic nets constraints expressing dependencies between the word forms. Finite verb forms demand semantic roles, according to valency patterns depending on the voice (active, passive) of the form and the governance (transitive, etc) of the root. Conversely, noun/adjective forms provide actors which may fill those roles, provided agreement constraints are satisfied. Tool words are mapped to transducers operating on tagged streams, allowing the modeling of linguistic phenomena such as coordination by abstract interpretation of actor streams. The parser ranks the various interpretations (matching actors with roles) with penalties, and returns to the user the minimum penalty analyses, for final validation of ambiguities. The whole platform is organized as a Web service, allowing the piecewise tagging of a Sanskrit text.

1. INTRODUCTION

This paper explains the application to the Sanskrit language and litterature of computational linguistics technology. Computational linguistics is the branch of Informatics dealing with the automated processing by computer software of natural language. It involves the mathematical modelling of the structure of linguistic utterances and their precise representation. In that sense, it is a natural outgrowth of descriptive linguistics, and is in a strong sense a close cousin of theoretical linguistics. Thus, the work of the MIT linguist Noam Chomsky in the sixties, in collaboration with the Parisian algebraist Marcel-Paul Schützenberger, was a major influence in the shaping up of modern theories of syntax. This led to a major development of non-commutative algebra, supporting a new discipline "Theory of automata and formal languages", at the frontier between mathematical logic and the emerging field of informatics or theoretical computer science. The power of descriptive formalisms for formal languages (sets of strings over a finite alphabet) was investigated thoroughly and their computational complexity assessed with the help of mathematical formalisms such as finite-state automata (regular languages, corresponding to finitely representable rational series), push-down stack automata (context-free languages, corresponding to algebraic series), etc.

The immediate application of this theory to the problem of parsing and interpreting computer software is well known. Its application to the treatment of natural language is slower to unfold, the natural theoretical complexity of natural language leading to higher complexity classes (roughly mildly context-sensitive languages, parsable in polynomial time). Furthermore, the intrinsic ambiguities of natural speech preclude any simple-minded formal treatment, since the main artificial intelligence stumbing blocks, such as common sense understanding, are ultimately lurking in any mechanical endeavour to understand natural language.

Despite these difficulties, considerable progress has been done in the second half of the 20th century to the computational treatment of natural language in its various syntactic and semantic aspects. Besides the specific signal processing techniques used for the treatment of speech, the main tools for this technical development are algebraico-logical tools on one hand, evolved from the aforementioned theories, and statistical methods, learning statistical biases from computerized corpus. The corresponding formal models are in strong critical interaction with linguistic theories. Con-

crete applications (notably to text processing tools such as spell and syntax correction, to tools for search in textual databases such as Web search engines, and in a longer term perspective to inter-lingual document translation software) have been strong incentives to develop powerful computerized linguistic platforms, notably for the English language.

Other languages are not so advanced in terms of computerization. On the one hand, elaborate theories of syntax for English are of little help for the development of parsers for languages having radically different syntactic structure. Thus, a language with strong morphology such as Sanskrit will tend to give light constraints to the sequential order of words in a sentence, contrarily to languages with rigid word order such as English, whose syntax enforces a rather strict subject-verb-object ordering. On the other hand, English has benefited from its situation as the de facto lingua franca of the modern business world, with generous funding of research and development of its linguistic treatment, leading to early large coverage computerized lexicons, extensive treebanks of tagged corpus, and a wealth of statistical data from immense speech and text resources.

Thus until recently there was very little computational support for Sanskrit. Digitalized texts were entered manually, there was no mechanical way to tag or search reliably Sanskrit corpus, and consequently no computerized philology assistance, even for simple statistics computations. This paper presents an ongoing effort to fill this need.

2. THE VYĀKARAṆA TRADITION

The linguistic tradition of ancient India was developed very early as a part of valid Vedic knowledge. Among the six recognized Vedic branches of knowledge (*vedaṣaḍaṅga*), four are concerned with language: phonetics (*śikṣā*), grammar (*vyākaraṇa*), hermeneutics (*nirukta*) and prosody (*chandas*). In these ancient disciplines, linguistics is not the study of the vernacular language (*bhāṣā*), but concerns rather the *refined language* (*saṃskṛta*) - i.e. Sanskrit. In particular, the grammatical tradition of Sanskrit is extremely ancient. Many Sanskrit scholars must have studied the formation rules of phonetics and morphology, until an intellectual giant, Pāṇini, appeared in the 5th century before Christian era, with a grammar in eight chapters (*Aṣṭādhyāyī*) which superseded all previous work and set an unsurpassed standard of linguistic precision. Further linguists (notably Kātyāyana and Patañjali) will essentially limit themselves to commentaries aiming at the proper explanation of the very terse and formal presentation of the *Aṣṭādhyāyī*.

Pāṇini's grammar is an almost exhaustive descriptive presentation of linguistic facts concerning the refined language used by intellectuals of his time, which came to be labeled as Classical Sanskrit. It also describes a number of archaic forms, in order to accommodate the much earlier language of the Vedas. It had such authority that it was soon taken as prescriptive, and consequently Sanskrit was more or less frozen in its evolution, except that the existence of formal generative rules justified their use at arbitrary levels of recursion, unknown up to then in spontaneous speech. Thus the inductive definition of compound substantives was taken as incentive by poets to form arbitrarily long nominal phrases, composed of just one compound word.

Pāṇini's grammar is presented in the form of extremely terse aphorisms (*sūtra*), whose proper understanding needs specific technical training. The most striking characteristic of this work is its density. The complete listing of its rules fits easily in 100 pages of a small format book [28]. Most aphorisms are formal rules, governing rewriting operations over strings composed of Sanskrit phonemes, complemented with formal markers used as meta-linguistic tokens. A complex system of exceptions governs the application of one rule over another in case of ambiguities. The system is to a certain extent lexicalized, since it has to be understood with the help of a roots lexicon (*dhātupāṭha*) and a nouns lexicon (*gaṇapāṭha*) providing morphological information.

It would seem natural to profit of this early tradition in formal linguistics to attempt to use Pāṇini's grammar at the core of a computational linguistics platform for Sanskrit. This is *not* what we chose to do, and an explanation is in order.

The first problem is that Pāṇini's grammar is *generative*, in the sense that it explains how a Sanskrit locutor with a communicative intention is to proceed in order to construct a correct sentence with the intended meaning. It thus proceeds from semantics to syntax to morphology to phonology, applying rules until he obtains a terminal stream of phonemes representing the correct enunciation of a syntactically correct paraphrase of his intended message. At every step in the process the earlier decisions are available, and thus for instance morphological derivations of a word may depend of the precise sense in which it is used in the sentence, phonological rules may depend on the meaning and morphological features of the word giving rise to the current phonological realization, etc. It is thus not possible to just inverse the rules to get an analysis of a Sanskrit sentence presented as a stream of phonemes (which is basically the input representation of a *devanāgarī* text, as a syllabic representation unambiguously decomposable as a stream of phonemes, representing faithfully the euphonic combination of the words composing the enunciation). In other words, Pāṇini tells how to speak, given the meaning to be conveyed, not how to understand a speech, i.e. extract its intended meaning from the phonetic signal. Even if we could formally present a Pāṇinean generator, its inversion would lead to all the non-determinism intrinsic to the linguistic ambiguities, which in last resort must use a lot of contextual information for their resolution, including metaphorical knowledge, proper names recognition, and common sense reasoning, all very complex phenomena not presently successfully modelisable computationally. Not to speak of the inherent ambiguities of sandhi exploited by poets to convey parallel meanings.

Another problem is that actually Pāṇini's grammar has not been to this day analysed to a degree of precision warranting its computer implementation. The mutual scope of the various rules is not explicitly stated in the *sūtraṇi* - only the beginning of sections distributing the applicability of a given rule is indicated, the end of the section being implicit. A long tradition of interpretation has led to more or less standard explicit indications of scoping, like in the classical [28], but such conventions have not yet brought a consensus on this issue. Furthermore, the meta-linguistic rewriting

machinery of Pāṇini has not yet been presented with a level of detail allowing its precise algorithmic properties to be assessed - is there a complete mechanism for arbitration of conflicting rules, is the system determinate (confluence), is the rewriting always terminating, what is the computational power of the underlying formal system, etc. Such issues are currently a hot topic of research, and progress on these issues ought to follow, but at present there is no serious hope to build a "computerized Pāṇini" soon, and still less to use it in reverse as a parsing tool. Furthermore, as remarked recently on Internet by Michael Witzel from Harvard University, a critical edition of the *Aṣṭādhyāyī* manuscripts and of its various commentaries is still badly lacking.

A final consideration is of course that the need of linguistic tools for making sense of real Sanskrit corpus, even restricted to the Classical Sanskrit period, will demand a much wider coverage than the canonical Pāṇinean utterances. Not only Sanskrit new vocabulary was acquired in the 25 centuries time span since he wrote his work, but a lot of say Epic corpus contains important variations from his standard [27]. Specialized dialects exist, such as Buddhist Sanskrit, as well as regional variants, although admittedly the linguistic evolution phenomenon is much less active for Sanskrit, because of the *trimuni* tradition, than for vernacular languages. Even though, a linguistic analysis of Sanskrit theater plays will demand proper modeling of *prākṛita* (vernacular) dialects such as *ardhamāgadhī*, in order to analyse the discourse of women and servants.

For all these reasons, Pāṇini's grammar was not chosen as the core ingredient in the linguistic modeling of our platform. Nonetheless, Pāṇini's grammar is ultimately the golden standard against which the generative tools underlying any Sanskrit implementation must be measured.

3. LEXICON

It is our thesis that any platform of natural language mechanical treatment must put the lexicon at the heart of the model, since it is the basic repository of all linguistic facts through which the various layers must communicate. This is a general trend in computational linguistic research: algebraico-logical tools must be lexicalized, in the sense of driving the combinatorial process by rules and parameters read from the lexical database.

Actually our computer lexicon is obtained mechanically from a Sanskrit to French dictionary which we started back in 1994, originally as a human-readable Sanskrit lexicon of Indian culture. Its reverse engineering as a computer-readable lexical database was told in [12, 14, 20]. From the source of the dictionary is mechanically extracted a lexical database of root words informed with morphological features, which is itself further processed by morphological generative processes to databanks of inflected forms, as we shall see. It is to be noted that this process is not a one-time translation: the Sanskrit to French dictionary keeps to be updated asynchronously with the platform development.

A word about the computational environment is in order. We use as our platform programming language a version of the ML functional language developed at INRIA for the last 20 years, called Objective Caml [25]. The main advantage for this kind of computational linguistics software is its modular nature. Parameterized modules called functors, described in separately compilable source files, allow proper sharing of interfaces of independently developed processes. Thus, the translation of our dictionary source into lexical databases is an instance of a generic process which compiles our high-level dictionary presentation into a parametric backend generative process. One such process is the actual printing of the dictionary as a typographically neat high-resolution book form (through the devnag-TeX-pdf chain of treatment), and as a Web site interlinking a hypertext version of the dictionary with various linguistic tools (through XML-HTML-CSS-CGI-Unicode Web standards). The full production of a new version of the whole platform takes less than an hour of computer time on my laptop, and thus a high rate of versioning is easily achieved, with incremental compiling of the software and linguistic data.

The structure of our Sanskrit database is independent of French, which concerns only the semantic definition of the words, and not their combinatorial features. Thus the dictionary is designed as a multilingual dictionary, where semantic components for other natural languages (English, Hindi, other Indian languages) will be able to replace the current French interface, either by translation of the corresponding entry, or by alignment of existing digitalized Sanskrit dictionaries such as the Monier-Williams Sanskrit-English dictionary.

Our current lexical coverage, at the time of writing is the following. Our dictionay has 15486 entries, of which 544 are verbal roots, 8512 are substantival generative stems and elementary undeclinables, 3797 are secondary derived stems and compounds, 223 are composed compounds, 60 are morphological suffixes, 121 are frozen inflected forms, 400 are idiomatic forms, 1314 are idiomatic expressions, 203 are citations and 312 are cross-reference links. This vocabulary coverage is sufficient for the analysis of simple Classical Sanskrit sentences, given that our analysis takes care of preverbs recognition and compound analysis, as we shall see.

4. PHONOLOGY

The main stumbling block for the mechanical analysis of Sanskrit is *sandhi*, i.e. euphonic combination. Sandhi comes in two varieties, morphological sandhi and sentential/compound sandhi. The first kind is invoked when a particle is affixed to a stem, typicallly as a derivational morpheme, or as a flexional suffix. The second kind is used for glueing words together for forming compound words and sentences. In the Western grammatical terminology, the first one is called internal, the second one external. This terminology is to a certain extent misleading. External sandhi is more or less well defined, and consists in local rewriting of phonetic strings (although special cases make it context dependent, for instance there are special cases for dual forms and some pronominal forms). Internal sandhi is more complex, since it involves long-distance cascading retroflex transformations. Furthermore, it is not unique as a string merging operation, there are various forms of internal sandhi according to the morphological process in which it is used - this is where a closer Pāṇinean modeling would be useful.

We modeled external sandhi as a regular relation over phone-

mic streams, with contextual rewrite rules attached as attributes to the lexicalised forms. We showed that the regular relation is invertible by an appropriate finite-state transducer, in a situation where the number of analyses is actually finite. This modelisation involved the development of a finite-state library for Ocaml, the ZEN toolkit [15, 16, 18, 19], available as free software at the distribution site `http://sanskrit.inria.fr/ZEN`. This toolkit is actually independent of Sanskrit. The complete explanation of the segmentation process (*sandhi viccheda*) is available for computational linguists and theoretical computer scientists as [21], simplified accounts for natural language specialists have been presented in [13, 17].

We also wrote an internal sandhi computer, which operates on phonemic streams not just as a finite state transducer, but as a computational process involving context-dependent transformations for instance for retroflex transformation. This internal sandhi generator is the workhorse of flexional morphology, as we shall see. In order to avoid parasitic overgeneration of roots and stems ending in phonemes *j* and *h*, we have been led to split the corresponding phonemes into two tokens, with the addition of so-called "extra phonemes" *j'* and *h'*. These are indeed remnants from an earlier Indo-European stratum of language, still implicit from their distinct sandhi behaviour. Thus *j'* combines with *t* to form combination *ṣṭ* (whereas *j* obtains *ḍh*). Similarly *h'* combines with *t* to form combination *gdh* (whereas *h* obtains *kt*). The lexicon indicates that roots *bhrāj*, *mṛj*, *yaj*, *rāj*, *vraj* and *sṛj* are ended in *j'*, leading to forms such as participle *mṛṣṭa*. Similarly, roots *dah*, *dih*, *duh*, *druh*, *muh* and *snih* are ended in *h'*, leading to forms such as *dugdha*. These are typical context dependent internal sandhi rules, localized as uniform transformations with the help of additional lexicalized markers, fully in the spirit of Pāṇini.

5. MORPHOLOGY

We distinguish traditionally between derivational morphology and flexional morphology. We first implemented flexional morphology for nouns, using paradigm tables collected from various Sanskrit grammars, mostly Western (Whitney, Macdonnel, Gonda, Renou, Kale). A given substantival stem, with a given gender, leads to a paradigm table listing the permissible suffixes for each valid combination of number and case. The suffixes are glued to the stem with internal sandhi, and the tagged corresponding form is entered in the inflected form database of the corresponding lexical category. Three categories are recognized for such forms: inflected nouns, bare form as left compound component, and inflected forms of root stems, usable only as right compound components. At the time of writing, the number of paradigms for nouns/adjectives, pronouns and numbers is 101.

The morphological generation of verbal forms was more complex, and its unfolding spread over more than two years. It combines derivational morphology - how to generate the stems of the various finite forms systems, and flexional morphology, i.e. conjugation of root forms. The complete paradigms were developed for the present system (present, imperfect, optative/potential and imperative) in the three voices (active-*parasmaipade*, reflexive-*ātmanepade*, and passive), the future (both in finite form and periphrastic form), the reduplicating perfect and the seven forms of aorist in both *parasmaipade* and *ātmanepade*. Furthermore, the causative, intensive and desiderative conjugations were derived, for the roots which were explicitly marked in the lexicon as adopting such forms. This covers the most usual classical Sanskrit root verbal forms.

The lexicon indicates explicitly which verbs take which preverb. This is very useful for going back and forth, through hypertext links, between a root and a verbal form where preverb affixing may have cascaded. The transitive closure of preverb affixing generated mechanically the set of 81 most usual preverb sequences, kept in a specific database of verbal prefixes. Rather than generating explicitly all verbal forms, only root forms are stored in the inflected forms database of the corresponding category, on the assumption that verbal forms would be analysed through external sandhi analysis. This is a simplifying assumption, and currently a cause of incompleteness, for the preverbs which give rise to specific sandhi, sometimes involving internal retroflexion, like in stems *pratiṣṭhā*, *pariṣvañj*, *niṣūd*, *nirṇayati*, *viṣṭambh*, etc. Such constructions will have to be dealt specifically at some further stage. It is to be remarked that these irregular formations are not dealt in a generic way by Pāṇini, whose grammar lists such exceptions in all painful detail.

A special mechanism had to be coined in order to deal with preverb *ā*, since because of its brevity – it is the only mono-phonemic lexical item in classical Sanskrit, whereas the Vedic language has an exclamative particule *u* – it may combine with the previous word in ways which cascade with its own sandhi with a root form in non-associative ways. Thus the utterance *ihehi* (come here) results from the sandhi of *iha* with *ā*, yielding the intermediate *ihā* (towards here), which itself combines with imperative *ihi* (go). Storing the verbal form *ehi* (come) would lead to the incorrect **ihaihi*. The analysis of this phenomenon induces the introduction of two so-called *phantom phonemes *e* and **o*, which permit the generation of special forms for the roots starting respectively by short or long *i* and by short or long *u* in order to pre-compute special forms such as **ehi* in our example, with specific sandhi rules simulating the left-to-right character of sandhi. This mechanism is described in [17].

Finally, we generated the participial stems of roots, a specially generative process, since each participle in its turn yields substantivally declensed forms in all genders, numbers and cases. We identified a present participle, a future participle and a perfect participle in both *parasmaipade* and *ātmanepade*, a present passive participle, and a future passive participle with three possible paradigms (*-ya*, *-īya*, *-tavya*). Undeclinable forms are generated for infinitives, absolutives (in *-ya* and *-tvā*), and the rare periphrastic perfect forms.

6. SEGMENTATION AND LEXICAL ANALYSIS

All these inflected word forms are stored in 9 databases, corresponding to 9 different lexical categories, identified as *phases* or states of the lexical analyser. The lexical analyser is modeled as a *modular transducer* in the sense of [23]. The diagram in Figure 1 shows the superstructure of this finite-state process.

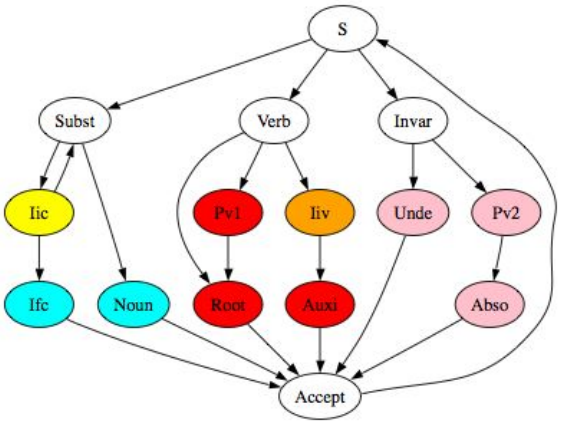

Figure 1: The 9-phases lexical analyser.

We can read on this figure the finite-state description of a Sanskrit sentence understood as a list of Words, where a Word is either a Substantive, a Verb, or an Invariable form. In turn a Substantive is an inflected Noun, possibly prefixed by an arbitrary list of compound-forming noun stems, or an inflected noun root form (Ifc), necessarily prefixed by a non-empty list of compound-forming noun stems. This last distinction is absolutely necessary in order to avoid bad over-generation of wrong segmentations, where the prefix-formations of $-pa$, $-ga$, $-da$ and the like would be attempted as prefixes of all kinds of words. Due to their shortness, these root forms must be restricted to right-components of compounds. Next, a Verb is a root form, possibly prefixed by a list of preverbs, or possibly a composite form of an auxiliary root finite form, with a special periphratic form of a substantive formed with an *-ī* suffix. We coined Iiv (*in initio verbum*) by analogy with Iic standing for *in initio compositi*, a common Western categorization. Such compound forms are rarely described in grammars, whereas they form common idiomatic expressions such as *ghanī bhū* (to harden), *dūrī bhū* (to go away), *dūrī kṛ* (to send away), *navī kṛ* (to renovate), *pavitrī kṛ* (to purify), *bhasmī kṛ* (to reduce to cinders), etc. In order for such idiomatic expressions to be recognized as legal Sanskrit, the lexical analyser must accommodate these *-ī* forms. Rather than trying to have an exhaustive list of such idiomatic usage in our lexicon, and an associated complex sub-automaton, we decided to make this lexical category generative, and to duplicate the finite root forms of auxiliaries *kṛ* and *bhū*. In a similar spirit of simplicity we decided not to limit preverb prefixes for a given root to the ones explicitly listed in the lexicon, although the corresponding data structure is indeed explicit in our implementation. Instead, we collect all preverb prefixes used for at least one root, and allow to regognize them as prefixes of any root. The slight overgeneration which this induces is compensated in our opinion, firstly in the simplified and thus more compact automaton, and secondly in the generative capacity with which it recognizes preverb formation. Our machine will recognize some verb forms as legal even when the corresponding entry does not exist in our limited lexicon.

Maybe this list of legal preverb sequences is worth listing explicitly, since to our knowledge this information is not available in standard grammars: *ati*, *adhi*, *adhyava*, *adhyā*, *anu*, *anuparā*, *anupra*, *anuvi*, *anta*, *apa*, *apā*, *abhi*, *abhini*, *abhipra*, *abhivi*, *abhisam*, *abhyanu*, *abhyava*, *abhyā*, *abhyut*, *abhyupa*, *abhyupā*, *ava*, *ā*, *ut*, *utpra*, *udā*, *upa*, *upani*, *upasam*, *upā*, *upādhi*, *tira*, *ni*, *ni*, *nirava*, *nirā*, *parā*, *pari*, *parini*, *parisam*, *paryupa*, *pura*, *pra*, *prati*, *pratini*, *prativi*, *pratisam*, *pratyapa*, *pratyava*, *pratyā*, *pratyut*, *prani*, *pravi*, *pravyā*, *prā*, *vi*, *vini*, *vini*, *viparā*, *vipari*, *vipra*, *vyati*, *vyapa*, *vyabhi*, *vyava*, *vyā*, *vyut*, *sa*, *saṃni*, *saṃpra*, *saṃprati*, *saṃpravi*, *saṃvi*, *sam*, *samava*, *samā*, *samut*, *samudā*, *samudvi*, *samupa*.

Finally, the Invariable words are classed into firstly Undeclinable forms, such as adverbs, prepositions, conjunctions and other tool words and particles listed in the lexicon, to which are added the root absolutives in *-tvā*, and secondly absolutive stems in *-ya*, necessarily prefixed by a non-empty preverb sequence. We see here how the distinction between the two forms of absolutives is dealt with correctly by the regular grammar formalization.

The state graph shown in Figure 1 is only the super-structure at the phase level, since each phase lexical analysis uses a finer-grain state machine compiled from the corresponding lexical database. The inter-linking of the various machines is implicit from the operation of the reactive engine implementing the lexical analyser as a modular transducer [23]. We remark that proper sharing is induced by this mechanism. Thus the two phases Pv1 and Pv2 of the diagram correspond to a unique preverb sequences recognizer.

Most importantly, the lexical analyser is an external sandhi analyser, which inverts the sandhi rational relation into a finite stream of segmentation solutions, along the *sandhi viccheda* algorithm explained in [21]. That is, a transition from one phase to the next in the above diagram predicts possible sandhi segmentation, and reports it as a stream of segmentation solutions, yielding an explicit sequence of segments with appropriate sandhi rules. Thus, for instance, on input *tacchrutvā* (having heard this), the lexer responds with one segmentation solution:

```
Input: tacchrutvaa
may be segmented as:

 Solution 1 :
[ tat <t|"s -> cch>]
[ "srutvaa <>]
```

This means that the segmentation *tat śrutvā* is recognized, with sandhi merging the final phoneme *t* of *tat* with the initial phoneme *ś* of *śrutvā* into the *liaison* phoneme string *cch*. The answer is thus a very precise *certificate* proving that the input string may indeed be derived from the stated words. Furthermore, since these words are generated from explicit morphological derivations, we may readily lemmatize each form by tagging the lexicon stem with its morphological features, by simple table look-up. In this lemmatizing mode, we effectively get a (non-deterministic) tagger. On the above example, this yields:

```
Input: tacchrutvaa
```

```
may be segmented as:

 Solution 1 :
[ tat
  { acc. sg. n. | nom. sg. n. }[tad] <t|"s -> cch>]
[ "srutvaa
  { abs. }["sru] <>]
```

Furthermore, in the Web interface, the stem *tat* and *śru* are hyperlinks to the corresponding dictionary entries, yielding an approximate interpretation, at least as much as lexical semantics allows.

Here is the treament of a more realistic example (the standing boy answers the master's questions):

```
Input: ti.s.thanbaalakasupaadhyaayasyapra"snaa-
       naamuttaraa.nikathayati
may be segmented as:

 Solution 1 :
[ ti.s.than
  { nom. sg. m. | voc. sg. m. }[ti.s.that] <>]
[ baalakas
  { nom. sg. m. }[baalaka] <>]
[ upaadhyaayasya
  { g. sg. m. }[upaadhyaaya] <>]
[ pra"snaanaam
  { g. pl. m. }[pra"sna] <>]
[ uttaraa.ni
  { acc. pl. n. | nom. pl. n. }[uttara] <>]
[ kathayati
  { pr. ac. sg. 3 }[kath] <>]

 Solution 2 :
...
```

Remark that the stream of input phonemes is continuous, there is no blank in the representation of a continuous *devanāgarī* text. Indeed, the user may provide a few blank spaces here and there, helping the system in guessing the segments, and this will in general decrease the number of returned solutions.

This state of affairs was reported at the XIIth World Sanskrit Conference in Helsinki in August 2003, and the broad lines of the implementation of this Sanskrit tagger were recently presented for an audience of linguists in [22].

Let us give a few figures concerning the automata sizes. Our lexicon currently generates 167163 full and initial iic noun forms, 127122 root verbal forms, 16571 ifc final noun forms, 841 undeclinable forms and 81 preverbs. It also generates the very large number of 224300 participial forms, which actually are not used at present in this tagging engine. The most frequent participles actually occur (usually as adjectives) in the nouns database. Therefore the participial forms are actually not used by the current version of the tagger. They shall be used at a latter stage, as part of a more robust stemmer which will attempt to analyse non-lexicalised forms, as part of a process of lexicon acquisition from the corpus. The actual transducers data-structures built by compiling the inflected forms databases with sandhi prediction points, is very compact due to the maximal sharing policy of the Zen toolkit. The state transition structure for roots fits in only 155Kb, the nouns one in 1M byte, the iic one in 119Kb, , the iiv one in 67Kb, the ifc one in 22Kb, the aux one in 23Kb. Overall, the full transducer representation needs less than 1.4Mb. Furthermore, its execution time is negligible with respect to the time needed to print the result, since all database accesses have been precompiled into the automaton structure.

The correctness, completeness and termination of this algorithm is proved in [21], assuming that all the forms composing the sentence are indeed generated by the morphological generator. Presently there are three difficulties. The first one corresponds to the exocentric or adjectival usage of compounds (*bahuvrīhi*), since such compounds may be used in a gender which the right component does not admit by itself. Thus *bīja* (seed), a neuter noun, forms the compound *raktabīja*, usable in the masculine gender as naming a monster "he whose blood is regenerated". Its nominative form *raktabījaḥ* is not obtainable as the iic form *rakta* followed by an inflected form of stem *bīja* - unless one generates all forms in all three genders for every noun, at the risk of important overgeneration. For the present version, we chose rather to record in the lexicon such *bahuvrīhi* usage of compounds, and to add the corresponding missing forms – such as *raktabījaḥ*.

Actually, a preprocessing phase on the lexicon identifies all compounds which are irregular, in the sense of not being obtainable by external sandhi of the iic. form of their left component with a regular stem of their right component. This analysis adds 180 'autonomous' compound forms. This list includes *dvandva* compounds with double dual forms, such as *mitrāvaruṇau*, compounds whose left component uses a feminine form, such as *durgāpūjā*, or an inflected form, such as *vasuṃdhara*, irregular external sandhi such as *pṛṣodara* or *viśvāmitra* or even internal sandhi such as *rāmāyaṇa*. Their inclusion as autonomous ready-made nouns allows the generation of their forms, since our automaton is not able to get their analysis (as compounds) from their components. Besides these exceptional cases, all compounds formed with stems from the lexicon are analysable, down to any nesting level, without the need to have them explicitly listed as lexicon entries.

A kind of reverse difficulty occurs for *avyayībhāva* compounds such as *yathāśakti*, which ought to be recognized as an invariable adverb, although its rightmost component admits inflexion. This is a minor cause of overgeneration. Finally, we shall have to add verbal forms in the cases where the affixing of a preverb provokes retroflexion or other sandhi transformation out of the realm of external sandhi, like *pratiṣṭhā*.

7. SHALLOW PARSING

The main problem with the tagger we just described is its horrendous over-generation. For the simple sentence given above, we get 60 candidate segmentations, most of which are completely nonsensical. For a slightly longer sentence, the number of candidate segmentations may well exceed 10000,

rendering the tool useless without extra analysis. This is why we set to build a filtering mechanism, based on semantic principles. In Pāṇinian terms, this amount to do its *kāraka* analysis, in Western terms this means computing dependency structures consistent with thematic roles of constituants.

The analysis consists in trying to match the valency or subcategorisation pattern of verbal forms - needed thematic roles for the verb to be meaningful for denoting a situation or action - with the possible roles assumed by noun phrases through their case classification. Thus a transitive verb will demand an Agent and a Patient. If it is in the active voice, these will be realized respectively by a substantive form in the nominative case (the Subject) and a substantive form in the accusative case (the Object). If it is in the passive case, we shall need an instrumental for the Agent; if the verb is transitive, we shall need also a nominative for the Patient. As in the previous case, the nominative must agree in number and person with the verbal finite form. For passive intransitive verbs, in the Impersonal aspect, there is no need of nominative, since there is no grammatical subject, the logical subject being the instrumental Agent. This leads to a graph-matching algorithm, for each possible interpretation of tags in any given candidate segmentation. Extra substantives in the nominative case will be tentatively merged with an already chosen one, provided they agree in number and gender, in order to account for adjectival use and appositions. Extra substantives in other cases are taken as semantically neutral adverbs (or unanalysed possessive noun phrases for genitives). Similarly vocatives are ignored. When finite verb forms are missing, the copula is postulated, and a substantive in the nominative case is taken as the predicate. Missing roles, as well as extra nominatives, are counted as penalities. For each branch fixing a choice of morphological tag to every segment, an integer penality is thus computed, and the minimum of all such penalities is the compound penality of a candidate segmentation. Then only segmentations with minimal penalty are presented to the user, in two lists. The first list is the preferred one, it gives the minimal penalty solutions with minimal number of segments. The second one gives second-choice potential solutions, with lowest penalty but non-minimal number of segments.

This algorithm, crude as it is, is very efficient in practice to reject most non-sensical solutions, while keeping the intended interpretation within a small list of candidate solutions proposed to the user for perusal. Let us give a few typical uses on non-trivial sentences. First of all, on our *tiṣṭhanbālakasupādhyāyasyapraśnānāmuttarāṇikathayati* example above, our parser returns only one segmentation among 60 possible ones (it is actually the first solution shown above), and thus the filtering efficiency is 100% in this example. Only one additional candidate is proposed along.

Let us now consider a sentence which exhibits interesting levels of ambiguity: *śvetodhāvati*. The lexical analyser finds 15 candidate segmentations. The parser filters out all of them but the first one, proposed as best interpretation:

```
Input: "svetodhaavati
may be segmented as:
```

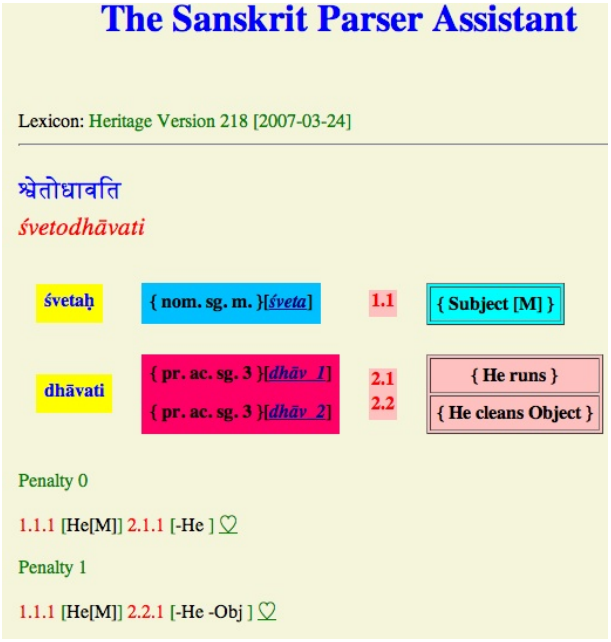

Figure 2: Semantic analyses of solution 1

```
  Solution 1 :
[ "svetas
  { nom. sg. m. }["sveta] <as|dh -> odh>]
[ dhaavati
  { pr. ac. sg. 3 }[dhaav#1]
| { pr. ac. sg. 3 }[dhaav#2] <>]
```

In this solution, there is an ambiguity between *dhāvati* as a form of root $dhāv_1$ (he runs) and as a homophonic form of root $dhāv_2$ (he cleans). The subject of the verb is the nominative form *śvetaḥ* - the white one (typically a white horse may be available from the context). But the cleaning interpretation demands an object to be cleaned, which is not available in the sentence, whence a penalty of 1. Our parser Web interface gives this information on demand, for every potential solution. The user may thus peruse the various interpretations, in order to finalize the morphological tags choice, and focus on one completely disambiguated interpretation. Figure 2 shows a screen image of the choices available to him if he chooses this solution.

Each semantic analysis is marked with a green heart symbol, which may be activated by the user in his final choice. If he clicks on the first one, with penalty 0, he will get the final unambiguous tagging corresponding to "the white (one) runs". He may then store this solution as an XML structure, and enter the next sentence of his text. Our machinery may thus be used as a little Sanskrit Tagging Web service.

Now it so happens that this sentence is actually ambiguous. It may also be parsed as "The dog runs here". This actually corresponds to solution 13, also with penalty 0, but relegated to the subsidiary list of candidate solutions, since it involves 3 segments instead of 2:

Figure 3: Semantic analyses of solution 13

```
 Solution 13 :
[ "svaa
  { nom. sg. m. }["svan#1] <aa|i -> e>]
[ itas
  { adv. }[itas] <as|dh -> odh>]
[ dhaavati
  { pr. ac. sg. 3 }[dhaav#1]
| { pr. ac. sg. 3 }[dhaav#2] <>]
```

If the user selects this solution, he will get the analysis in Figure 3.

Again, the two homophonic roots *dhāv* give rise to an ambiguity, but for the same reason as above the only acceptable one is "to run", leading to the second possible interpretation of this sentence.

We stress that our treatment of *kārakas*/thematic roles is minimal. We have so far only used the crudest subcategorisation patterns of verbs, namely whether they may be used as transitive or intransitive verbs. We could of course refine this analysis, demanding a dative argument for verbs of gift, etc. But the additional complexity of dealing with complex subcategorization patterns, with an added level of non-determinism, does not seem worth the trouble for the simple task of filtering out non-sensical interpretations.

We end this section with an example in the Impersonal voice. If we input the sentence *mayāsupyate* (I sleep), our parser returns correctly only one solution *mayā supyate* (out of 14 possible segmentations), with correct identification of the Agent (I) and process (sleeping), and with no penalty induced from the absence of a grammatical subject.

8. TOWARDS A MORE SOPHISTICATED SYNTACTIC ANALYSIS

The semantic analysis sketched in the preceding section is of course extremely naive. It amounts to deny any role to syntax as an autonomous structure, and assumes implicitly that the order of words in a Sanskrit sentence is irrelevant to its meaning. This is tenable only for very simple sentences, without subordinate verbal phrases (except absolutives, which may be modeled adequately to a certain extent, by accommodating the ellipsed Agent through appropriate subcategorization patterns for the absolutive forms). Also, common phenomena such as coordination are not accommodated properly in this simplistic commutative world. A more sophisticated treatment demands a proper consideration for a syntactic structural level, where the left-to-right precedence ordering between successive words is still available.

One problem in this quest is the scarcity of available material on descriptive Sanskrit syntax, in order to have rigorous arguments to choose one formal model of syntax or the other before even considering its mechanization. Besides Speijer's book on Sanskrit syntax [34] and Apte's "Guide to Sanskrit Composition as a treatise on Sanskrit syntax" [1], both 19th century compositions, there is very little litterature on the topic. Pāṇini is quite exhaustive on morphology, but discusses syntax very cursorily - it is not even clear if he explicitly authorizes a sentence with several finite verbal forms. He basically limits his analysis to the *kāraka* theory which we strive to emulate with the constraint machinery sketched above. Bhartṛhari, an Indian grammarian from the Seventh century, has produced an important treaty on semantics, the Vākyapadīya, which has not been at this date analysed by modern theoretical linguists in ways which could lead to a proper treatment of Sanskrit syntax. His work is actually a semiotics theory, comparable to the Western tradition of philosophy of language, but with the distinctive character that in the Indian tradition grammar is prior to logic, and thus the notions of semantics arise from natural language constructions, instead of relying on logical considerations built on mathematical principles - our proof theory of mathematical logic.

Of course there exist numerous case studies on aspects of Sanskrit Syntax [30, 36, 11], but few general theories. In [35], Staal discusses the crucial problem of word order in Sanskrit, proposing a "Calder mobile" model of dependency structures, which is a first approximation to a model of Sanskrit syntax, in which crossings by long-distance dependencies is ruled out. Generalizing the model to more dislocated sentences, a common phenomenon in Sanskrit litterature, involves an analysis of the constraints governing such dislocations, without degenerating into the fully free commutative model. Recent papers by B. Gillon [9, 8, 7] investigate such questions, by giving appropriate syntactic analysis to characteristic constructions taken from the examples in Apte's book. We intend to profit of these analyses in the design of our Sanskrit syntax analyser.

We decided to make experiments in the processing of a Sanskrit sentence by mediating the interface between our lexical analyser and the semantic constraint satisfaction module through a syntactic layer seen as the interpreter for linguis-

tic processes triggered by the particles and other tool words from Sanskrit. The relevant context is that of a *linguistic tool* modeled as a transducer over morphologically tagged word streams. A linguistic tool is triggered by a tool word in the stream of input words. What it does is to take control over the stream of the preceding words, to apply some transduction over it, and to return this as the input stream to the next item. This methodology is consistent with the postfix nature of most Sanskrit particles, typically *iti*, which closes a level of direct language as some sort of quotation ending, whose beginning is implicit - i.e. must be decided by the *iti* tool in light of the preceding context. In its case, the tool applies not just to the preceding stream of lexical items of the current sentence, but to the whole preceeding context.

We implemented as an experiment two such tools. The first one is the *saha* (with) tool, which just checks that its predecessor is a substantive in the instrumental case, and absorbs it so that it does not contribute to further dependencies. Here the resulting tag is no more a mere morphological tag, but rather a Saha clause having the instrumental item as an argument constituant. That is, our tool transducers are really operators on syntax tree streams.

The second tool concerns coordination, it is the *ca* (and) tool. This tool looks in its immediate left neighborhood, possibly grouping words in noun phrases in the case of agreement (typically an adjective qualifying a noun, and agreeing with it is number gender and case). It then applies addition of two or more such morphological chunks in order to return a compound chunk, making explicit the coordination structure. Here addition is an operation computing independently in the three dimensions of a noun morphological tag. Over their number, it is a non-standard commutative addition, where one plus one equals two, two plus one equals many, and many plus anything equals many. Over their gender, it is a max operation, where Masculine is higher than Feminine, itself higher than Neuter. Over their case, it must check that they agree. Finally, when the case is Nominative, it computes the addition of their person as a max operation, where the first person is higher than the second which is higher than the third (thus you and I makes us, and he and you makes the whole of you). Actually, the combinatorics of the *ca* particle is a little bit more complex, but I am simplifying in this exposition. With the help of the *ca* tool, our parser is able to find the right solution (among 120 possible segmentations) to the sentence *dvitīyakakṣyāyāṃdvebāliketrayobālakāścapaṭhante* (in the second class two girls and three boys are studying).

9. CONCLUSION

We have described in this paper a methodology for a computer-aided analysis of Sanskrit sentences, with three tiers. The first tier is a complete sandhi segmenter. The second tier is a dynamic shuffling of the morphologically tagged lexemes by stream combinators governed by the tool particles. It is very sketchy at this point. The third tier is the thematic roles constraint processing machinery, which we did not describe in detail, but which strives to be consistent with Pāṇini's view of syntax, as exposed for instance in Kiparsky's paper on the architecture of Pāṇini's grammar [24]. However, Pāṇini's *kāraka* assignment procedure does not give a explicit role to the polarities which underly our model, in the spirit of linear logic or categorial grammars - thematic roles in the verb subcategorization patterns are of negative polarity, and match the positive polarity of the actors, represented by substantive phrases of a given case. Furthermore, his treatment of Nominative as a default case is remote from our own treatment where the Subject is a prominent notion, except in the Impersonal voice. This may reflect an incorrect linguistic bias, and the role constraint management may require some adjustements to correct this discrepancy, which calls for further investigation.

We believe our notion of linguistic tool is a valuable abstraction. We have validated the general idea on two specific constructions, but many more tools need to be defined, and their mutual interaction will raise important problems in the elaboration of a proper semantical treatment. Furthermore, our current implementation is too naive to be viable in a long-term real-scale usage. Syntactic constructions are not shared during the non-deterministic search, leading to ultimate exponential behaviour. We believe a serious treatment of syntactic constraints must use the modern implementations of constraint programming to search for their satisfaction. Furthermore, appropriate notions of underspecified partial structures, in the spirit of Tree Adjoint Grammars or more recently of Interaction Grammars must be put to use to reify the dependency structure.

A proper notion of syntax will have of course to analyse compounds along their various modes of formation. The *bahuvrīhi* construction, or exocentric compound, promotes a substantive to an adjective (like in English long-nosed as the adjective qualifying a person with a long nose) itself often substantivized as a typical member of its class. These two coercions must be marked for their semantic interpretation, although they have no phonetic realisation. It seems unavoidable to depart from the tradition of pure dependency graphs, where only phonetically realised items support the structure.

Our Sanskrit engine, comprising the dictionary, parser and associated support tools, is accessible from Internet as a Web service, at URL `http://sanskrit.inria.fr`. It is easily available on any station running an HTTP server as a stand-alone offline application, and thus on any Linux box with Apache, or on Macintosh stations. Mirror sites of the Sanskrit engine have been installed at the University of Hyderabad and at the Rashtriya Sanskrit Vidyapeetha in Tirupati. Our morphological databases, as tagged XML documents, are available as free linguistic resources downloadable from the above URL. The Zen finite-state toolkit is available as free software from `http://sanskrit.inria.fr/ZEN/`.

Work is under way to adapt this machinery to the construction of a treebank of parsed sentences, issued from characteristic examples from Apte's Sanskrit Syntax manual. This work is in cooperation with Pr Brendan Gillon from McGill University. It is expected that this treebank will be used to learn statisticallly the parameters of the parser in order to increase its precision. Other modules will attempt statistical tagging, needed for bootstrapping this prototype into a more robust analyser, usable for lexicon acquisition from the corpus.

10. REFERENCES

[1] V. S. Apte. *The Student's Guide to Sanskrit Composition. A Treatise on Sanskrit Syntax for Use of Schools and Colleges*. Lokasamgraha Press, Poona, India, 1885.

[2] A. Bergaigne. *Manuel pour étudier la langue sanscrite*. F. Vieweg, Paris, 1884.

[3] R. S. Bucknell. *Sanskrit Manual*. Motilal Banarsidass, Delhi, 1994.

[4] M. Coulson. *Sanskrit - An Introduction to the Classical Language*. Hodder & Stoughton, 2nd ed., 1992.

[5] M. M. Deshpande. *A Sanskrit Primer*. University of Michigan, Ann Arbor, 2001.

[6] P.-S. Filliozat. *Grammaire sanskrite Pâninéenne*. Picard, Paris, 1988.

[7] B. S. Gillon. The autonomy of word formation: evidence form classical Sanskrit. Private communication, 1993.

[8] B. S. Gillon. Bartṛhari's solution to the problem of asamartha compounds. *Études Asiatiques/Asiatiche Studien*, 47,1:117–133, 1993.

[9] B. S. Gillon. Word order in classical Sanskrit. *Indian Linguistics*, 57,1:1–35, 1996.

[10] J. Gonda. *A Concise Elementary Grammar of the Sanskrit Language (Tr. Gordon B. Ford Jr)*. E. J. Brill, Leiden, 1966.

[11] H. H. Hock, editor. *Studies in Sanskrit Syntax*. Motilal Banarsidass, Delhi, 1991.

[12] G. Huet. Structure of a Sanskrit dictionary. Technical report, INRIA, 2000. `http://pauillac.inria.fr/~huet/PUBLIC/Dicostruct.ps`

[13] G. Huet. Computational tools for Sanskrit. XXIth South Asian Languages Analysis Roundatable, University of Constanz, 2001.

[14] 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.

[15] G. Huet. The Zen computational linguistics toolkit. Technical report, ESSLLI Course Notes, 2002. `http://pauillac.inria.fr/~huet/ZEN/esslli.pdf`

[16] G. Huet. The Zen computational linguistics toolkit: Lexicon structures and morphology computations using a modular functional programming language. In *Tutorial, Language Engineering Conference LEC'2002*, 2002.

[17] G. Huet. Towards computational processing of Sanskrit. In *International Conference on Natural Language Processing (ICON)*, 2003.

[18] 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. `http://pauillac.inria.fr/~huet/PUBLIC/padl.pdf`

[19] G. Huet. Automata mista. In N. Dershowitz, editor, *Verification: Theory and Practice: Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, pages 359–372. Springer-Verlag LNCS vol. 2772, 2004. `http://pauillac.inria.fr/~huet/PUBLIC/zohar.pdf`

[20] 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. `http://pauillac.inria.fr/~huet/PUBLIC/coling.pdf`

[21] G. Huet. A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger. *J. Functional Programming*, 15,4:573–614, 2005. `http://pauillac.inria.fr/~huet/PUBLIC/tagger.pdf`.

[22] 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.

[23] G. Huet and B. Razet. The reactive engine for modular transducers. In J.-P. J. Kokichi Futatsugi 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. `http://pauillac.inria.fr/~huet/PUBLIC/engine.pdf`

[24] P. Kiparsky. On the architecture of Pāṇini's grammar. In *International Conference on the Architecture of Grammar*, 2002.

[25] X. Leroy, D. Rémy, J. Vouillon, and D. Doligez. *The Objective Caml system, documentation and user's manual – release 3.00*. INRIA, 2000.

[26] M. Müller. *A Sanskrit Grammar for Beginners*. Munshiram Manoharlal, Delhi, repr. 2000.

[27] T. Oberlies. *A Grammar of Epic Sanskrit*. De Gruyter, Berlin, 2003.

[28] Pāṇini. *Aṣṭādhyāyī*. Chaukhamba Surbharati Prakashan, Vārāṇasī, 2004.

[29] E. D. Perry. *A Sanskrit Primer*. Columbia University Press, New York, 1936.

[30] L. Renou. *La valeur du parfait dans les hymnes védiques*. Edouard Champion, Paris, 1925.

[31] L. Renou. *Terminologie grammaticale du sanskrit*. Edouard Champion, Paris, 1942.

[32] L. Renou. *Grammaire sanscrite*. Adrien Maisonneuve, Paris, 1984.

[33] P. Scharf. *Rāmopākhyāna - the Story of Rāma in the Mahābhārata. An independent-study Reader in Sanskrit*. Routledge & Curzon, London, 2003.

[34] J. S. Speijer. *Sanskrit Syntax*. E. J. Brill, Leyden, 1886.

[35] J. F. Staal. *Word Order in Sanskrit and Universal Grammar*. Reidel, Dordrecht, 1967.

[36] B. Tikkanen. *The Sanskrit Gerund: a Synchronic, Diachronic and typological analysis*. Finnish Oriental Society, Helsinki, 1987.

[37] W. D. Whitney. *Sanskrit Grammar*. Leipzig, 1924. 5th edition.

[38] W. D. Whitney. *Roots, Verb-forms and Primary Derivatives of the Sanskrit Language*. Motilal Banarsidass, Delhi, 1997. (1st edition 1885).

[39] M. Williams. *A Practical Grammar of the Sanskrit Language*. Munshiram Manoharlal, Delhi, repr. 2000.