Sanskrit Morphological Analyser: Some Issues

Amba Kulkarni and Devanand Shukl

Department of Sanskrit Studies,
University of Hyderabad,
Hyderabad
apksh@uohyd.ernet.in,dev.shukl@gmail.com

1 Introduction

Sanskrit has rich inflectional as well as derivational morphology. At the same time it enjoys the privilege of having a formal (to a large extent) grammar in the form of Aṣṭādhyāyī built over 25 centuries ago. One may think, therefore, it would be a trivial task to build a morphological analyser based on this grammar. But it is not so. Though well defined rules for Sanskrit morphology exist in Aṣṭādhyāyī, for a typical computational linguist without any knowledge of Sanskrit, it is difficult to build a system incorporating these rules. At the same time, it is rare to find a person who understands the Aṣṭādhyāyī well and has also a good knowledge of computers for its implementation. Further there are controversies over the use of meta rules (paribhāṣā) used for application of the rules of Aṣṭādhyāyī. Therefore, incorporating the rules of Aṣṭādhyāyī as they are, is a difficult task. As a result, in spite of the existence of a formally defined and well described grammar, construction of a set of computational tools for the analysis of Sanskrit texts could not take a momentum for a long time.

If we look at the Sanskrit literature, we see that many attempts have been made to render the learning of Sanskrit word formation easier. The Sanskrit grammatical tradition provides evidence for the implementation of the two well known models of morphological descriptions viz. Item and Process model and the Word and Paradigm model. While the rules in Aṣṭādhyāyī may be best described as an example of Item and Process model, the later Sanskrit texts such as Siddhānta-Kaumudī[15], Rūpacandrikā (a compilation of representative word forms), Dhāturatnākara[19] (a lexicon of verb forms), Kṛdanta-Rūpamālā[17] (a book listing frequently used nominal stems derived from the verbal roots) use the Word and Paradigm model and are best suited either for pedagogical purpose or for quick reference. Both these traditions are equally popular among the Sanskrit scholars and have their own advantages.

The issues related to the development of Sanskrit morphological analyser ranging from the representation of phonemes for computation, the suitability of Unicode versus various transliteration schemes for internal representation, handling sandhi and the complexity involved in building the analyser are well described

by Huet in [7],[8]. In this paper, we would like to model the morphology described by Pāṇini from a computational perspective, and show how one can take advantage of the various readily available databases described above to develop a morphological analyser quickly. Such an analyser then can be used to evaluate the performance of various other morphological analysers. The issues involved in comparing the analysers are discussed, and finally we suggest evaluation criteria to evaluate the morphological analysers.

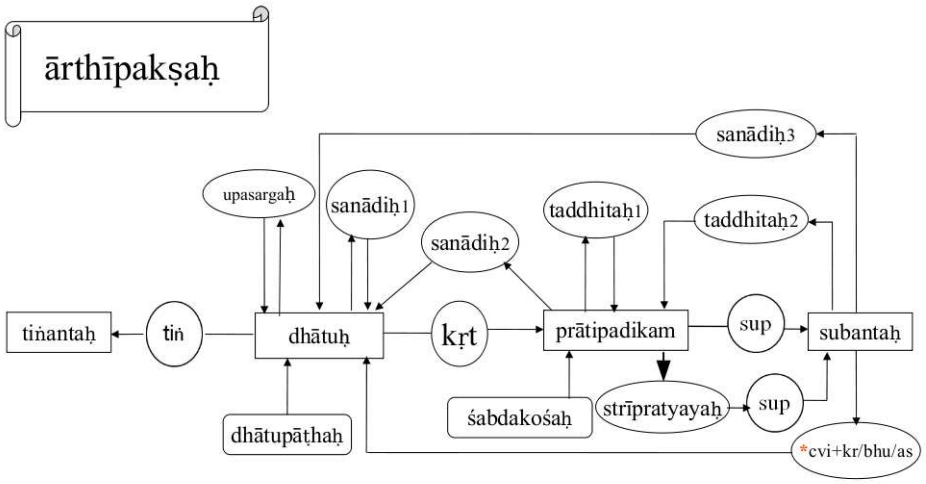
2 Word Formation in Sanskrit

Figure given below gives a computer scientist's perspective of the Sanskrit word formation as described by Pāṇini. We deviate from Pāṇini at certain places and interpret the whole process from the meaning point of view rather than just looking at the process involved therein[1]. The basic inflectional morphology is described by the sūtra *suptiṅantam padam* (1.4.14). Thus in the figure below, the formation of verbal forms (tiṅantas) and the nominal forms (subantas) from the verbal roots (dhātus) and the nominal bases (prātipadikas) respectively, represents the inflectional morphology in Sanskrit. The list of non-derived verbal roots is more or less a closed list[2]. The list of nominal bases on the other hand is open[3]. *Sup* and *tiṅ* are the list of inflectional suffixes a nominal base and a verbal root take respectively.

The *kṛt* and the *taddhita* are the derivational suffixes added to the verbal roots[4] and nominal bases[5] respectively, producing new nominal bases. *Sanādi* suffixes are of 3 types: some operate on nominal bases[6], some on nouns[7] and some operate on verbal roots[8]. Prefixes when attached to the verbal roots change the meaning of verbal roots and inflect further like a verb[9]. Finally there are certain nominal forms which take a special suffix *cvi* followed by one of the verbal roots *kṛ*, *bhū* or *as* to produce new verbal bases[10].

It is obvious from the given figure that the recursion in this word formation diagram leads to over-generation. But as is evident from the Pāṇinian sūtras, there

[1] From processing point of view first the *tiṅ* is attached and then the upasarga gets attached to the verb form, whereas from the meaning point of view, verbal root together with the upasarga give a special meaning to the verb.
[2] We find some instances of verbal roots which are not listed in the dhātupāṭha such as *ḍamb* from which the word *viḍambanā* is derived.
[3] arthavat adhātuḥ apratyayaḥ prātipadikam (1.2.45)
[4] (dhātoḥ) kṛt atiṅ (3.1.93)
[5] kṛt taddhita samāsaśca (1.2.46)
[6] lohitādiḍājabhaḥ kyaṣ 3.1.13
[7] sup ātmanaḥ kyac (3.1.8), etc.
[8] sanādyanta dhātava.h (3.1.32)
[9] prādayaḥ upasargaḥ kriyāyoge (1.4.58)
[10] kṛbhvastiyoge saṃpadyakartari cviḥ (5.4.50)



	Legends:
tiṅantaḥ: verbal form	**tiṅ**: verbal suffix
prātipadikam: nominal base	**sup**: nominal suffix
kṛt: derivational suffix	**sanādiḥ**: derivational suffixes
upasargaḥ: verbal prefix	**śabdakoṣaḥ**: list of nominal bases
strīpratyayaḥ: feminine suffixes	**taddhitaḥ**: nominal derivational suffixes
subantaḥ: nominal form	**dhātupāṭaḥ**: list of verbal roots

is no mention of any limit over the recursion. Here are some examples showing recursion and productive nature of Sanskrit morphology.

1. pāṭhayitā = paṭh(verbal root) + ṇic(sanādiḥ1) + tṛc(kṛt) + feminine suffix + sup .
2. pipāṭhayiṣati = paṭh(verbal root) + ṇic(sanādiḥ1) + san(sanādiḥ1) + tiṅ .
3. vyāvahārī = vi(upasarga) + āṅ(upasarga) + ava(upasarga) + hṛ(verbal root) + ṇic(sanādih1) + añ + feminine suffix + sup .
4. samabhivyāhāra = sam(upasarga) + abhi(upasarga) + vi(upasarga) + āṅ(upasarga) + hṛ(verbal root) + ghañ(kṛt) + sup .
5. susammodayantikā = su(upasarga) + sam(upasarga) + mud(verbal root) + ṇic(sanādiḥ1) + śatṛ(kṛt) + feminine suffix + kan(taddhita) + feminine suffix + sup .

However, the ability of human mind to process a complex string puts a limit on these potentially infinite productions to a finite number and is supported by the actual data. Words with more than three suffixes are typically rare compared to the words with single suffix or double suffixes. Further, for all practical purposes, the derived word forms can always be kept in the lexicon as headwords. These may be analyzed if there is a need.

There is another important phenomenon which needs to be discussed viz. the compound formation in Sanskrit. The Sanskrit word *samāsaḥ* for a compound means *samasanam* which means "combination of more than one words into one word which conveys the same meaning as that of the collection of the component words together". While combining the components together, a compound undergoes certain operations such as loss of case suffixes, loss of accent etc. ([12], [3]). The Sanskrit compounds may be classified in two different ways – semantic and syntactic. Sanskrit literature mentions 6 types of syntactic classification[11] viz:

Subanta (noun) + Subanta (noun) (*rājapuruṣaḥ*)
Subanta (noun) + Tinanta (verb) (*paryyabhūṣayat*)
Subanta (noun) + nāma (nominal base) (*kumbhakāraḥ*)
Subanta (noun) + Dhātu (verbal root) (*kataprū*)
Tinanta (verb) + Subanta (noun) (*kṛntavicakṣaṇā*)
Tinanta (verb) + Tinanta (verb) (*pibata-khādata*)

Of these, only three types of compounds viz. noun-noun, noun-nonimal base and noun-verb(essentially the pre-verb verb sequence) are very frequent, and other 3 are rare. The rare compounds can always go in the lexicon. It is the productive compounds that need to be handled by the morphological analyser. Though compounds of 2 or 3 words are more frequent, compounds involving more than 3 constituent words with some compounds even running through pages is very common in Sanskrit literature. Here are some examples of Sanskrit compounds involving more than 3 words.

- veda-vedāṅga-tatva-jñaḥ
- pravara-mukuṭa-maṇi-marīci-mañjarī-caya-carcita-caraṇa-yugala
- jala-ādi-vyāpaka-pṛthivītva-abhāva-pratiyogi-pṛthivītva-vatī

Only the last component of the compound has the case suffix, and all the previous components assume a 'compounding form' – distinct from the nominal base. Thus a Sanskrit morphological analyser needs to analyse the compounding forms as well. Another phenomenon of Sanskrit compounds that needs to be handled by Sanskrit morpological analyser is the exo-centric(bahuvrīhi) compounds with the head or the final component exhibiting a gender of an object it denotes which may be different from its default gender. For example consider an exo-centric compound formed by the words *pīta* (yellow) and *ambaram* (cloth). The word *ambaram* is in neuter gender. When it is used to refer to a man wearing yellow clothes its form is *pītāmbaraḥ* and when it refers to a woman wearing yellow clothes its form is *pītāmbarā*. The head words *ambaraḥ* and *ambarā* are the nominative singular forms in masculine and feminine gender respectively. The default morphological analyser will fail to analyse these forms, *ambaram* being a neuter gender word.

[11] supāṁ supā tiṅā nāmnā dhātunātha tiṅāṁ tiṅā |
subanteti vijñeyaḥ samāsaḥ ṣaḍvidhoḥ budheḥ ||

2.1 Formal Grammar for Sanskrit Morphology

For the benefit of computational linguists, we translate the above figure into a more formal grammar by following the notation of context free grammar formalism. This exercise, we believe, will give us a better way to compare various implementations.[12]

1. verbal_form = verbal_root .[13] verbal_suffix (*(sup)tiṅantam padam* 1.4.14)
2. nominal form = stem . sup (*sup(tiṅ)antam padam* 1.4.14)
3. stem = nominal_stem + compound_stem
4. verbal_root = root (from Dhātupāṭha) +
   verbal_root . sanādiḥ1 (*sanādyanta dhātavaḥ* 3.1.32) +
   nominal_stem . sanādiḥ2 (*lohitādiḍājabhyaḥ kyaṣ* 3.1.13) +
   nominal_form . sanādiḥ3 (*sup ātmanaḥ kyac* 3.1.8) +
   upasargaḥ_seq . root (*upasargāḥ dhātuyoge* 1.4.58; Also covers the compound of type *supāṁ tiṅā*) +
   base . cvi . bhū,kṛ,as (*kṛbhvastiyoge sampadyakartari cviḥ 5.4.50)*
5. nominal_stem = nominal_base + nominal_base . feminine_suffix
6. nominal_base = base (*arthavat adhātuḥ apratyayaḥ prātipadikam* 1.2.45) +
   base . taddhita (*(kṛt) taddhita (samāsaḥ) ca* 1.2.46) +
   verbal_root . kṛt (*kṛt (taddhita samāsaḥ) ca* 1.2.46, and *dhātoḥ kṛt atiṅ* 3.1.96)
7. comp_stem = comp_base + compound_base . feminine_suffix
8. compound_base = pre_compound_base . nominal_base (*supāṁ supā*)
   pre_compound_base . {bhū,prū,..} (*supāṁ dhātu*) +
   pre_compound_base . {kāra/da/ja/...} (*supāṁ nāmnā*)
9. pre_compound_base = nominal_base . compound_suff +
   pre_compound_base . nominal_base . compound_suff

3 Various efforts

Attempts have been made in the past to develop morphological analysers for Sanskrit by the Center for the Development of Advanced Computing, India and Academy of Sanskrit Research, Melkote, India. However, they had either limited coverage or were almost unusable for any serious NLP applications. There have been notable efforts in the last few years in the area of Sanskrit computing. The efforts by Mishra[13], Scharf[16], Goyal[4], and Sridhar[18] are towards modelling Aṣṭādhyāyī using computers. They try to simulate the process of word generation following the rules of Aṣṭādhyāyī as closely as possible. One of the goals in their efforts is to understand 'how' much formal the system is and also to provide an environment so that various interpretations of Pāṇini's rules may be tested objectively. These implementations are still in evolving stage. Huet[7],[8]

[12] We thank G. Huet for the valuable discussions on the Sanskrit Word Formation. These discussions not only helped us to improve the Sanskrit Word Formation figure substantially but also motivated us to express the grammar in a more formal way.

[13] . is used to represent a joining operation and + indicates alternative possibilities

is more focussed on building a practical system rather than on the philosophical issues related to the structure of Aṣṭādhyāyī etc. Naturally he uses the modular finite state transducers to model the morphological phenomenon including the internal sandhi as well. The various inflected and derived forms are generated following the Word and Paradigm approach, and though is not in the strict sense Pāṇinian, captures the generalisation found in Pāṇini. Jha et al[10] have implemented the Sanskrit morphological analyser using database models wherein they store the suffixes and stems in the databases and report the possible stem-suffix search pairs from the databases. The suffix database can also be used standalone to handle out-of-lexicon words by 'guessing' their suffixes. The Akshar Bharati group has been working on the development of morphological analysers for various Indian languages using the Word and Paradigm model, which is now extended to Sanskrit as well. The complexity of Sanksrit morphological analyser being more when compared to other Indian languages, and at the same time availability of various databases of readymade word forms, the group followed a different approach taking leverage of available resources and data bases for Sanskrit. We describe this approach here.

4 Building a wide coverage morphological analyser quickly

Developing a morphological analysers by designing a generator and then using a Finite State Automata to analyse is a natural choice. But in case of Sanskrit, writing a morphological generator with a reasonable coverage itself is a few man years effort. At the same time, Sanskrit is rich in various databases. For example, Dhāturatnākara is a list of finite verb forms for various verbs for different lakāras (tense-mood-aspect). In a strict sense it does not follow the Pāṇinian Dhātupāṭha, and at some places also deviates a bit from Pāṇinian grammar, but is still a good starting point. Kṛdanta-rūpamālā gives the list of frequently occuring non-finite verbal forms for various verbs. Rūpacandrikā lists all the noun paradigms. These books are being used by people with some exposure of Sanskrit Grammar as an aid to understand and interpret various Sanskrit texts. As such these time-tested books serve as a rich source of readymade databases for developing a reasonable coverage Sanskrit morphological analyser quickly.

Morphological analysers being an important module needed in any NLP application, we decided to take leverage of these existing resources and built a system quickly. The existing resources provide all the analysed forms, from which a Finite State Transducer is built. There are two advantages of this approach. The first one is – in a short span of time, a system with reasonable coverage is available for use which can be plugged in into any other NLP application. The other advantage is, since this system is built from the actual forms listed in the books, and not generated by any software, the data used may be treated as a GOLD standard data for testing the generators built following various approches.

We used the lt-toolbox[14] – a tool developed by the Apertium group for developing the FST for Sanskrit Analysis. Separate FSTs are built for analysing the basic inflectional forms, the primary derivatives (i.e. kṛdantas) along with the inflected forms that can be generated from them, the secondary derivatives (i.e. taddhitas) along with their inflected forms, compounding forms and finally a FST to recognise the heads of the exo-centric compounds that have gender different from their default gender. These separate FSTs can always be combined together to get a single FST. The trade-off is between the compile time and the run time. The total forms - inflectional, derivational and compounding - our analyser could recognise are 140 million. It is now necessary to evaluate the morphological analyser and also to compare it with other existing morphological analysers to verify the claim that the data used to build the analyser forms a GOLD data.

5 Issues

Initial level of comparison of the noun morphological analysers developed by Scharf-Hyman, Huet and ours brought out some issues.

1. Treatement of feminine adjectives:
   According to the Pāṇinian system at morphological level, nouns and adjectives behave alike. In case of adjectives before forming their feminine forms a feminine suffix is added to the nominal base. Thus *pūrva* is the nominal base in masculine and neuter, and pūrvā in feminine. However, Pāṇini does not treat the nominal base with feminine suffix as a *prātipadika*[15]. Now the question is should the analysis of feminine forms produce the root as pūrvā or pūrva. From semantics point of view, it is reasonable to agree that pūrva is the base in all the three genders. It brings in brevity, and also the organisation of the dictionary is simple.
2. Treatment of homonymy:
   Another issue is related to homonymy. The question is should the morphologically indistinguishable forms with the same stem but different meanings be analysed seperately? This in fact is difficult to handle as it is very difficult to have consensus on whether a word is homonymous or polysemous. It would be desirable, then to distinguish the roots at morphological level, provided they have different behaviour at this level. For example, there are two different seventh case singular forms of the word *sva* viz. *sve* and *svasmin*. These two forms in fact relate to two different roots. In one case *sva* means the 'self', whereas in other case it means 'one's own property'.
3. Level of analysis:
   There is always an issue in the representation of analysis in case of derivational morphology. For example, for the word *gacchati*, should the machine produce the output as seventh case singular form of the nominal base *gacchat*

[14] http://www.apertium.org
[15] ṅyāp-prātipadikāt(4.1.2)

or should it further add the information that the word *gacchat* is derived from the verbal root *gam* by adding a *kṛt* suffix? With the use of computers, however, this is not an issue, since computers provide several ways of presenting the information in layers, thereby hiding the unnecessary information. While designing a tool for comparison of analysis generated by various analysers, one needs to take this factor into account.

Many more such issues will surface, once we extend our comparison for verbal inflections and other forms.

5.1 Evaluation Parameters

Typically the evaluation metrics of any tool involves two parameters – Precision and Recall. Precision tells you the confidence which you can have on the performance of the tool, and the Recall gives you the coverage.

Since the morphological analyser gives more than one answers, the evaluation of Precision and Recall should range over all possible analysis and not over the words. Further only these two parameters for evaluation of morphological analyser are not enough. We need a more sophisticated measure which tells us which of the answers produced by the morphological analyser are wrong, and also how many answers are missing.

5.2 Precision and Recall

Let the total number of words be N.
Let A_i denote number of possible analysis for the word W_i.
Thus total number of possible answers for all the words together is
T $= \sum_{i=1}^{n} A_i$.
Let B_i denote total answers produced by the morphological analyser for a word W_i.
Let C_i denote the number of correct answers (true positives) and
D_i denote the number of wrong answers (false positives).
Thus $B_i = C_i + D_i$.

Sum of all the correct answers is C $= \sum_{i=1}^{n} C_i$.
Sum of all the wrong answers is D $= \sum_{i=1}^{n} D_i$.
Precision $P = C/(C + D)$
Recall $R = C/T$

5.3 6 point evaluation

Since the morphological analyser produces more than one answers, it will be appropriate to carry out more detailed evaluation of the morpholoical analyser than just evaluating the precision and recall values. Precision-Recall gives general

overall impression about the performance of a system. A more detailed evaluation is necessary to know what kind of words are over analysed, which are under analysed, etc. There are 3 parameters viz. an analysis produced by the machine is either correct (true positive) or wrong (false positive) or machine has missed (false negative) an analysis. So when a machine produces an output, it may have zero or more correct analysis, zero or more wrong analysis and zero or more missing analysis. Accordingly, we have the following possibilities:

sr no	Correct (True positives)	Missing (False Negatives)	Wrong (False Positives)	remark
1	1	1	1	ativyāpti as well as avyāpti
2	1	1	0	avyāpti
3	1	0	1	ativyāpti
4	1	0	0	Ideal Case
5	0	1	1	sp case of 1
6	0	1	0	Unrecognised words (sp case of 2)
The other two possibilities will never arise				

The frequency count of each of these 6 cases, and also the morphological phenomena related to these cases will help the developers of morphological analysers to decide which aspect of morphology needs further attention for improvement.

5.4 Evaluation Methodology

Several Sanskrit texts tagged maually at various levels – at the level of sandhi splits (*padaccheda*), morphological analysis (*pada viśleṣaṇa*), sentential analysis (*anvaya*, *śābdabodha*), etc. are available[16]. These manually tagged data may be used for evaluating the system performance. However, comparing the machine produced morphological analysis with these manually tagged texts will give the Precision Recall values only for the 'words in a given context'. These values will be biased because it is possible that the word has some other analysis in some other context which goes un-noticed or un-evaluated. Better evaluation strategy would be to prepare a GOLD standard data for evaluation of morphological analysers. This data contains all possible analysis of the words selected carefully to represent various morph phenomenon. The evaluation – either Precision Recall or 6 point may be done against this GOLD standard data. The possible drawback of this method is, it is possible that missing out some rare analysis lowers down the performance of a system. For all practical purposes, it is also necesary to know the performance on more frequent analysis. For this purpose, one may use manually analysed texts.

[16] E.g. Saṁkṣeparāmāyaṇam by Rashtriya Sanskrit Sansthan, Rāmopākhyāna by Peter Scharf, Bhagvad-geeta by Geeta Press, Gorakhpur, to name a few.

6 Conclusion

In this paper we have presented a pictorial representation of the Sanskrit morphology as described by Pāṇini. Further we discussed in brief how we can take the leverege of various available resources and build the Sanskrit morphological analyser quickly, that can serve as a model to compare and evaluate other morphological analysers. We also suggested two different criteria for the evaluation of morphological analysers. An indicative list of issues involved in the comparison of various analysers based on our experience is given. It is now a high time to carry out such comparison rigourously to smoothen out the differences in the representation if any.

References

1. B. S. Gillon: *Autonomy of word formation: evidence from Classical Sanskrit.* Indian Linguistics, 56 (1-4), pages 1552, 1995.
2. B. S. Gillon: *Exocentric Compounds in Classical Sanskrit.* In G. Huet, A. Kulkarni, and P. Scharf, editors, Sanskrit Computational Linguistics 1 and 2. Springer-Verlag LNAI 5402, 2009.
3. 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.
4. P. Goyal, A. Kulkarni and L. Behra: *Computer Simulation of Aṣṭādhyāyī: Some Insights* In A. Kulkarni, and G. Huet, editors, Sanskrit Computational Linguistics 1 and 2, pages 139-161, Springer-Verlag LNAI 5402, 2009.
5. O. Hellwig: *Sanskrit Tagger: A Stochastic Lexical and POS Tagger for Sanskrit.* In G. Huet, A. Kulkarni, and P. Scharf, editors, Sanskrit Computational Linguistics 1 and 2, pages 266277. Springer-Verlag LNAI 5402, 2009.
6. O. Hellwig: *Extracting dependency trees form from Sanskrit texts*, In A. Kulkarni and G. Huet, editors, Sanskrit Computational Linguistics 3, pages 106115. Springer-Verlag LNAI 5406, 2009.
7. G. Huet: *A functional toolkit for morphological and phonological processing, application to a Sanskrit tagger.* J. Functional Programming, 15,4:573614, 2005.
8. G. Huet: *Formal Structure of Sanskrit Text: Requirement Analysis for a Mechanical Sanskrit Processor.* In G. Huet, A. Kulkarni, and P. Scharf, editors, Sanskrit Computational Linguistics 1 and 2. Springer-Verlag LNAI 5402, 2009.
9. Pandit Ishvarachandra: *Aṣṭādhyāyī.* Chaukhamba Sanskrit Pratisthan, Delhi, 2004.
10. G. N. Jha, M. Agrawal, Subash, S. K. Mishra, D. Mani, D. Mishra, M. Bhadra and S. K. Singh: *Inflectional Morphology Analyzer for Sanskrit* In A. Kulkarni, and G. Huet, editors, Sanskrit Computational Linguistics 1 and 2, pages 219-238, Springer-Verlag LNAI 5402, 2009.
11. A. Kulkarni and V. Sheeba: *Building a wide coverage Morphological Analyser for Sanskrit: A practical approach.* Invited speech at 'First National Symposium on Modeling and Shallow Parsing of Indian Languages', 31st March - 4th April 2006, IIT Mumbai.(http://sanskrit.uohyd.ernet.in/faculty/amba/)
12. Mahavira: *Pāṇini as Grammarian (With special reference to compound formation).* Bharatiya Vidya Prakashan [Delhi - Varanasi], India, June 1978.

13. A. Mishra: *Modelling the Grammatical Circle of the Pinian System of Sanskrit Grammar* In A. Kulkarni, and G. Huet, editors, Sanskrit Computational Linguistics 3, pages 40-55, Springer-Verlag LNAI 5406, 2009.
14. A. Mishra: *Simulating the Pinian System of Sanskrit Grammar* In A. Kulkarni, and G. Huet, editors, Sanskrit Computational Linguistics 1 and 2, pages 127-138, Springer-Verlag LNAI 5402, 2009.
15. Vasudeva Lakshman Shastri Panasikar: *Siddhānta Kaumudi*. Meharchand Lachamandas Publications, New-Delhi, 1985.
16. Peter M. Scharf: *Levels in Pāṇini's Aṣṭādhyāyī*. In A. Kulkarni, and G. Huet, editors, Sanskrit Computational Linguistics 3, pages 66-77, Springer-Verlag LNAI 5406, 2009.
17. S. Ramasubba Sastri: *Kṛdanta-Rūpamālā*, The Sanskrit Education Society, Madras, 1989
18. S Sridhar and S Varkhedi: *Computational Structure of the Aṣṭādhyāyī and Conflict Resolution Techniques* In A. Kulkarni, and G. Huet, editors, Sanskrit Computational Linguistics 3, pages 56-65, Springer-Verlag LNAI 5406, 2009.
19. Muni Lavannya Vijaya Suri: *Dhātu-Ratnākara* Bharatia Book Corporation, Delhi,2005.