वागर्थः

(An International Journal of Sanskrit Research)

# *Kāraka* analysis of complicated Sanskrit sentences

Sudhir K Mishra
Pune
sudhirkumarmishra@gmail.com

***Abstract:*** *The present paper is case studies of kāraka analysis of problematic kāraka situations of the following types –*

- *clipped sentences like 'gṛham' (in 'bhavān kutra gacchati ? gṛham)*
- *bigger noun phrases like sequences of adj1 adj2… adjn N with same vibhaktis (as in sundaraḥ suśīlaḥ nipuṇaḥ ca bālakaḥ) → identifying the kāraka in bālakaḥ only and not in other adjectives is a problem because we do not store nouns in the lexicon*
- *identifying lexical semantics and kārakas based on them (for example, the rule 'gati buddhipratyayavasānārthaśabdakarmākarmakāṇāmaṇikartā sa ṇau'[P 1.4.52] says that a particular kāraka will apply in the sense of these words)*
- *identification of abhihita and anabhihita (expressed and un-expressed)*
- *identification of the locus of the verb*
- *identification of the sense of tādarthya*

*kārakas play an important role in formation and analysis of sentences. Without complete analysis of kāraka, a sentence can not be analyzed. Analysis of Sanskrit sentences at both syntactic and semantic levels together through a computational model is challenging. By evolving a mechanism for kāraka interpretation of complicated Sanskrit sentences, the authors here present a case for using such systems for Sanskrit to Indian languages Machine Translation (MT). The overall aim is to test the algorithm on potentially problematic sentences to see if there is a need to further tuning the algorithm. This system is based on Pāṇini and Kātyāyana kāraka formulations*

***Key Words: Kārakas, Aṣṭādhyāyī.***

## I. INTRODUCTION

Sanskrit is a highly inflected and relatively free word order language. Therefore, identifying the constituents from the place cues (as in western languages) is not possible. In Sanskrit, the case endings of *padas* assign syntactic-semantic relations to the constituents of sentence with verb. In this work, the *vibhakti* endings and associated *kāraka* are analyzed for sentence comprehension. This approach is comparable with the broad class of *vibhakti* and *kāraka* based grammars such as Pāṇini and later grammarians. For *kāraka* analysis, first priority is identification of verb in sentence. Pāṇini discusses *kāraka* [P 1.4.23- P 1.4.54] and *vibhakti* [P 2.3.1-P 2.3.73.) in different chapters of Aṣṭādhyāyī. *Kāraka* is the underlying sense of the *vibhaktis* and *vibhaktis* are the markers of *kāraka. Kārakas* are not compulsory for each *pada* in a sentence, but *vibhaktis* are. So, the basic problem is correct identification of *kāraka* and *vibhakti* in a sentence. The task becomes slightly easier, if the verb is correctly identified and analyzed because many *kāraka* rules in Pāṇini assume the verb in the center. Secondly, except *kartṛ kāraka* all other *kārakas* are expressed only if they are un-expressed (*unabhihita)* by any other means (like *tiṅ*, *kṛt*, *taddhita*, *samāsa* or *nipāta*). If they are expressed (*abhihita*), they show as *kartṛ kāraka.* The present work assumes *sandhi* and *samāsa* free input text. The work on *kṛt*, *taddhita*, *samāsa* identification is in initial stage at this point.

*Kāraka* processing is done at three levels – structural, syntax and semantic. On the surface level, the identification of verb, *subanta*, *upasarga* and *avyaya* etc. will be done first, and then the verb and *kāraka* semantics is analyzed.

## II. KAS MODULES

The present research is actually being implemented as an online java servlet engine with relational database as backend. The system called *Kāraka* Analyzer for Sanskrit (KAS) has the following modules –

- the KAS engine.
- *tiṅanta* identification
- *subanta* identification
- *kāraka* identification and analysis

Pāṇini *kāraka* formulations are very complex and involve balanced interplay of morphological information, verb semantics, and sentence level syntax and semantics. The input text (according to the assumed specifications) will be checked for consistency by the KAS engine. If the consistency check succeeds, the *tiṅanta* identification is done with the help of a database of verb forms of commonly found verbs. This module will tag the verb for basic TAM, argument structure, *upasargas*, *nāma-dhatu*, derived forms, *vācya* etc. [Ref 11]. The *subanta* module will identify the case markers with the help of the *vibhakti* Knowledge Base (KBv). The KBv stores the primitive *vibhakti* morphemes and its allomorphs, and also possible exceptions. To make sure that KAS does not return wrong results for *upapada vibhaktis* like *paritaḥ kṛṣṇaṃ* (around *kṛṣṇa*) or other *avyaya-subanta* combination specially described by Pāṇini, this module will mark the constituent as suspect for special exception processing according to *kāraka* formulations. The *kāraka* module (KBk - a comprehensive database of *kāraka* formulations of Pāṇini, Patañjali and Kātyāyana) will search for *kāraka* rules for each *vibhakti* marked constituent and generate analysis for each *kāraka* - *vibhakti* situation in the sentence.

In case the KBv returns ambiguous results, the expectancy analysis of the verbs stored in the Verb Knowledge Base (KBV) as *sakarmaka*, *akarmaka*, *dvikarmaka, parasmai, ātmane, ubhaya, kartṛ-vācya, karma-vācya, bhāva-vācya* etc. will come in to disambiguate. The details for some of the components and problematic *kāraka* situations are as follows –

### A. Tiṅanta identification

Sanskrit verb forms are very complex. They carry tense, aspect, person, number information all in the inflection forms. Besides, they can also contain derivations containing semantic informations like causation, desire, repitition, negation etc. Therefore, it becomes very difficult to split out the verb and separate the verb root and complex information units encoded in it. Sanskrit has about 2000 verb roots classified in 10 morphological and semantic classes called *gaṇas*, and can also be further sub-classified as normal forms (without any of the 12 derivational affixes – 11 listed by Pāṇini [P 3.1.32], 1 more '*kvip*' added by Kātyāyana), and the derived forms with *ṇijanta* (causative – *ṇic*), *sannata* (expressing desire – *san, kyac, kāmyac, kvip, kyaṅ, kyaṣ,ṇiṅ, yak, āy and iyaṅ*) , *yaṅanta* (duplicated – *yaṅ* and *yaṅluṅant*). Then these can have *ātmane* and *parasmai* forms in 10 *lakāras* and 3 x 3 person and number combinations. Then these can also be potentially prefixed with 22 prefixes. Finally there could be in-numerable *nāmadhātus* (nominalized verbs).

We have stored all the verb roots from Pāṇini's *dhātu-pātha* (DP) with semantic class and other syntactic information. The backend structure is as follows –

| dhātu id | dhātu | artha | gaṇa | pada | s/ a/ v | ak./ sa./ dv |
|---|---|---|---|---|---|---|
| 1 | bhū | sattāyām | bhvādiḥ | p | s | ak |
| 2 | edha | vṛddhou | bhvādiḥ | p | s | ak |
| 3 | spardha | saṅgharṣe | bhvādiḥ | p | s | sa |
| 4 | gādhṛ | pratiṣṭhālipsayo-rgranthe | bhvādiḥ | p | s | sa |
| 5 | bādhṛ | vilodane | bhvādiḥ | p | s | sa |
| 6 | nāthṛ | yācñopatāpaiśva-ryāśīṣṣu | bhvādiḥ | p | s | dv |

**Table 1**

Since most of the DP *dhātus* are not found in literature, we have stored the forms for only 550 commonly occurring Sanskrit verb roots. The storage structure snippet in the backend is as follows –

| dhātu _id | 1.1.1 | 1.1.2 | 1.1.3 | 1.2.1 | 1.2.2 |
|---|---|---|---|---|---|
| 01 | bhavati | bhavtaḥ | bhavanti | bhavasi | bhavataḥ |
| 32 | yauti | yutaḥ | yuvanti | yauṣi | yuthaḥ |
| 39 | rauti | rutaḥ | ravanti | rauṣi | ruthḥ |
| 74 | nauti | nautaḥ | naunti | nautaasi | nautasthaḥ |
| 59 | kṣṇauti | kṣṇautaḥ | kṣṇonti | kṣṇauṣi | kṣṇauthaḥ |
| 76 | snauti | snautaḥ | snauvanti | snauṣi | snauthaḥ |
| 97 | uṇauti | uṇutaḥ | urṇuvanti | urṇaushi | urṇuthaḥ |

**Table 2**

### a. *tiṅanta* based *kāraka* complications

In this section, we are presenting some problems with respect to the *tiṅanta* identification –

- **In-complete sentences**
  sentences like *gṛham* which are answers to a question like *bhavāna kutra gacchati* ? or any other similar half or incomplete sentences will create problem in *Kāraka* analysis because the system will mark them as having no *kāraka* at all (as there is no verb). But such sentences do have verbs in the underlying representation. Therefore, the problem before us is to first complete these sentences with a suitable verb according to the context and then start *kāraka* analysis. Such single-word sentences could be verbs as well or ambiguous entities as in *āpaḍam kaḥ gacchati* ? *rāmaḥ*. In this instance, *rāmaḥ* may be a noun or a verb form of √*rā*. The KAS presently is not considering such sentences.

- ***Dvikarmaka* (di-transitive) verbs in certain senses**
  In such cases (as hinted in P 1.4.51 and later explained by *vṛttikāras*) the *dvikarmaka* verbs in 16 semantic categories mark *kārakas* optionally. For instance, in the sentences '*gām payaḥ dogdhi*' and '*go payaḥ dogdhi*', the *kārakas* are expressed differently in the same meaning. The optional use of *kāraka* in such cases depends on user *vivakṣā.*

### b. *Upasarga* based kāraka complication

[P 1.4.58] defines a class of 22 *nipātas* (*pra, prā, apa, sam, anu, ava, nis, nir, dus, dur, vi, āṅ, ni, adhi, api, ati, su, ut, abhi, prati, pari* and *upa*) listed in *prādigaṇa*. They are termed *upasarga* if they are used with a verb and play an important role in the identification of *kāraka*. [P 1.4.46] says if '*adhi*' *upasarga* is used before √*śīṅ,* √*sthā* and √*ās*, then the locus of verb gets *karma samjñā,* as in *adhiśete adhitiṣṭhati adhyāste vā vaikuṇṭham hariḥ* Some of these [P1.4.83 - P1.4.97] are discussed separately as *karmapravacaniya* with different *vibhakti* assignment rules. For instance, when '*upa*' implies inferiority it is termed *karma*, else if used in the sense of superiority then seventh *vibhakti* is used [P 1.4.87]. All such cases are stored separately as shown in the following table–

| *upasarga/ karmapravacaniya* | *dhātu* | *condition* | *kāraka/vibhakti* |
|---|---|---|---|
| adhi | śīṅ, sthā, ās | u + v = locus of verb | karman |
| upa, anu, adhi, āṅ | vas | u + v = locus of verb | karman |
| pari, apa, āṅ | | | fifth vibhakti |
| parā | ji | unbearable thing | fifth vibhakti |
| upa | | inferiority | second vibhakti |
| upa | | superiority | seventh vibhakti |

**Table 3**

### c. Vācya based kāraka complication

In Sanskrit there are three voices and in every voice sentence structure is different, for instance-

*kartṛ vācya* → subject in *prathamā vibhakti* + object in *dvitiyā vibhakti* + verb according to subject
*karma vācya* → subject in *tṛtīyā vibhakti* + object in *prathamā vibhakti* + verb according to object
*bhāva vācya* → subject in *tṛtīyā vibhakti* + no object
+ verb in third per, singular

this structure can help in solving the problem of ambiguity on surface level. The required information for this is stored in table 2 as shown above.

### d. Semantics based kāraka complication

- **the problem with √*spṛh***
  in the case of √*spṛh*, if the most desired object is marked karma by [P1.4.49], however, the other less desired objects are marked *sampradāna* [P1.4.36]. The KAS will provide both analyses. Such specific information is separately stored in the verb database.

- **the problem with √*nāthṛ***
  In the use of √*nāthṛ* , if the object of desire can optionally be marked by genitive marker [P 2.3.55] as in the sentence *sarpiṣo nāthate* (genitive) or *māṇavakam nāthate* (accusative). All such cases are stored separately as shown in the following table-

| Upa-sarga | Karmapra-vacaniya | dhātu | artha | condition | kāraka | vibhakti | rule |
|---|---|---|---|---|---|---|---|
| | | nāthṛ | āśīḥ | | | ṣaṣṭhī | 2.3.55 |
| | | gati | jānā | subject of ṇijanta | karman | | 1.4.52 |
| upa | | vas | not eating | locus of verb | adhikaraṇa | | vārtika |
| | anu | | tṛtīyā | | | dvitiyā | 1.4.85 |

**Table 4**

## B. Subanta identification

Correct *vibhakti* identification in nominal forms is a must for *kāraka* analysis. We are storing all possible allomorphs of the 21 (7x3) nominal *vibhaktis* in Sanskrit [P 4.1.2] as shown in the following table (for 'a' ending masculine nouns) -

| vibhakti | anta | liṅ | 1.1 | 1.2 | 1.3 |
|---|---|---|---|---|---|
| prathamā | a | P | aḥ | au | āḥ |
| prathamā | ā | P | āḥ | au | āḥ |
| prathamā | i | P | iḥ | ī | ayaḥ |
| prathamā | ī | P | īḥ | yau | yaḥ |
| prathamā | u | P | uḥ | ū | avaḥ |
| prathamā | ū | P | ūḥ | uvo | uvaḥ |
| prathamā | ṝ | P | ā | ārau/arau | āraḥ/araḥ |

**table 5**

There may be cases of ambiguity in some *vibhaktis* like *prathamā*, *dvitīyā* duals, *tṛtīyā*, *caturthī*, *pañcamī* plurals and also in *ṣaṣṭhī, saptamī* duals.

### a. Avyaya based kāraka complication

In case of indeclinable being used in conjunction with verbs, different *kārakas* are used as in *gurum namaskaroti* (*karma*), but if it not used otherwise, then the default *kāraka* will be used as in *gurave namaḥ* (*sampradāna*). This is discussed as *upa-pada-vibhakti* in Pāṇini. All such cases are stored separately as shown in the following table-

| avyaya | kāraka/vibhakti | exception | rule |
|---|---|---|---|
| namaḥ | caturthī | dvitiyā | 2.3.16 |
| nānā | dvitiyā, tṛtīyā, pañcamī | | 2.3.32 |
| ubhayatḥ | dvitiyā | | vārtika |
| abhitaḥ | dvitiyā | | vārtika |
| vinā | dvitiyā, tṛtīyā, pañcamī | | 2.3.32 |

**Table 6**

## C. Kāraka based complications

There are certain cases where the desire of the agent determines the *kāraka*. For example in case of more than one objects in a sentence, the most desired is *karma* according to

Pāṇini [P 1.4.50], however the other less desired are also termed *karma.* So, in sentences with such situations, the KAS should be able to differentiate between such *karmas*. For instance, in the sentence '*grāmam gacchan tṛṇam spṛśati*' ('while going to village (he) touches straw') agent's most desired goal is to go to village, and un-desired object is accidentally touching the straw (which he happens to trample on). Here both are marked object for different reasons. So, the KAS should be able to provide this analysis.

### a. **Mapping based Kāraka complication**

If any noun has n number of adjectives then the correct identification of the head noun becomes very challenging in Sanskrit as all of them will have the same *vibhaktis*. Since identifying the head noun may be important for *kāraka* analysis in cases of semantics bases assignments, this poses a big problem for any computer bases *kāraka* system. This becomes more challenging when the position of the head noun cannot be predicted due to relatively free word order within adj-n sequence in Sanskrit.

## III. SAMPLE ILLUSTRATION

The following examples illustrate the proposed kāraka processing of Sanskrit sentences by applying on Pāṇini and Kātyāyana kāraka formulations and data resources-

Input => makaradhvajena niśīthe prāyaśaḥ kāminaḥ balavaduttāpyante.

Module 1: uttāpyante →{([ut] Pre [tap] VR [yak] affix) laṭ_pra_bahu}

Module 2: karma vācya

Module 3: makaradhvajena (tri) niśīthe (sap) prāyaśaḥ (avy) kāminaḥ (pra) balavad
(pra)

Module 4: prāyaśaḥ (avyaya)

Module 5: makaradhvajena (2.3.18) niśīthe (2.3.7) prāyaśaḥ (avyaya) kāminaḥ (2.3.46)

## CONCLUSION

*Kāraka* analysis is complicated due to the complex nature of sentence structure in which several *kāraka* depends on other constituents of the sentence. It is only possible after integration of other modules like *subanta* analysis, *tiṅanta* analysis, *samāsa* analysis, *kṛdanta* analysis, *taddhita* analysis, *avyaya* analysis etc. The results and algorithm presented may need improvements based on the feedback.

## REFERENCES

[1]. A. Bharati, Sangal R., 1990, A karaka based approach to parsing of Indian languages, *proc of the 13 th COLING vol 3, pp 30-35*, Finland.

[2]. Rick Briggs, Knowledge representation in Sanskrit, *AI magazine,* 1985.

[3]. Sudhir K Mishra, Girish N Jha, Identifying Verb Inflections in Sanskrit Morphology, In *proc. of SIMPLE 05*, IIT Kharagpur, 2005, pp 79-81.

[4]. Sudhir K Mishra, Panini's Karaka System for Language Processing, Vidyanidhi Prakashan, New Delhi, 2016.

[5]. Sudhir K Mishra, *Aṣṭādhyāyīsūtrapāṭha* (Vārtika-Gaṇapāṭha-Dhātupāṭha-Liṅgānuśāsan-Uṇādi-Fiṭsūtrasahita), Vidyanidhi Prakashan, New Delhi, 2016.

[6]. Sudhir K Mishra, Computational Formulation and mapping of Pāṇini's Kāraka-Vibhakti for Machine Translation, International Journal of Linguistics & Computing Research, Vol. I, Issue. I, June-2017.