# Sanskrit Compound Paraphrase Generator

Anil Kumar[1], V.Sheebasudheer[2], Amba Kulkarni[3]

[1]Department of Sanskrit Studies, University of Hyderabad, Hyderabad,
anil.lalit22@gmail.com,
[2]Chinmaya International Foundation Shodhasansthan, Veliyanad, Kerala,
v.sheeba@gmail.com,
[3]Dept. of Sanskrit Studies, University of Hyderbad, Hyderabad,
ambapradeep@gmail.com

**Abstract**

Sanskrit is very rich in compound formation unlike modern Indian Languages. The compound formation being productive it forms an open-set and as such it is also not possible to list all the compounds in a dictionary. The compound formation involves a mandatory sandhi. But mere sandhi splitting does not help a reader in identifying the meaning of a compound, since typically a compound does not code the relation between its components explicitly. To understand the meaning of a compound, it is necessary to identify its components and discover the relation between them. An expression providing the meaning of a compound is called a paraphrase.

In this paper, we summarize our experience in building an automatic paraphrase generator for Sanskrit compounds. On the face of it, it simply turns out to be a generation problem and hence should be trivial. The actual implementation, however needed a morphological analyser that can handle the bound morphemes in compounds. The paraphrase handler could produce correct paraphrases for around 90% of the compounds.

## 1 Introduction

Sanskrit has more than 2500 years old almost exhaustive grammar in the form of Pānini's Aṣtādhyāyī which has the features of computationality. However, only recently Sanskrit Computional Linguistics[1] has gained a momentum. In this decade there have been many efforts to develop computational tools to access Sanskrit texts ( [3], [7], [12], [8], [2]). There have also been ongoing efforts to compare the performances of various morph analysers and also to discuss the interoperability issues, both at the national as well as international levels.

Most of the Sanskrit Computational tools handle morphological analysis and sandhi splitting. Some of them( [3]) also do the sentential parsing. However, there have been almost negligible efforts in handling Sanskrit compounds. Sanskrit is very rich in compound formation unlike modern Indian Languages. The compound formation being productive it forms an open-set and as such it is also not possible to list all the compounds in a dictionary. The compound formation involves a mandatory sandhi. But mere sandhi splitting does not help a reader in identifying the meaning of a compound, since typically a compound does not code the relation between its components explicitly. To understand the meaning of a compound, it is necessary to identify its components and discover the relation between them. An expression providing the meaning of a compound is called a paraphrase (or vigrahavākya).

In this paper we summarize our experience in building an automatic paraphrase generator for Sanskrit compounds. The organisation of the paper is as follows: We give a brief outline of salient features of Sanskrit compounds and its semantic classification in the next section. In the third section, we describe the actual algorithms. The results of evaluation are presented in the fourth section.

[1]International Symposium on Sanskrit Computational Linguistics held in 2007, 2008 and 2009.

## 2 Sanskrit Compounds

Typical Western Linguistics definition of compound is a "lexeme with more than one stems". 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. A Sanskrit compound thus has one or more of the following features ( [9], [5]):

1. It is a single word (ekapadam).
2. It has a single case suffix (ekavibhaktikam) with an exception of (aluk) compounds such as (yudhiṣṭiraḥ), where there is no deletion of case suffix of the first component.
3. It has a single accent(ekasvaraḥ).
4. The order of components in a compound is fixed.
5. No words can be inserted in between the compounds.
6. The compound formation is binary with an exception of dvandva and bahupada bahuvrīhi.
7. Euphonic change (sandhi) is a must in a compound formation./
8. Constituents of a compound may require special gender or number different from thier default gender and number. e.g. pāṇipādam, pācikābhāryaḥ, etc.

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.

1. वेदवेदाङ्गतत्त्वज्ञः[2]=वेद - वेदाङ्ग - तत्त्व - ज्ञः(a person knowledgable in the philosophies/essence of the veda and vedānga)

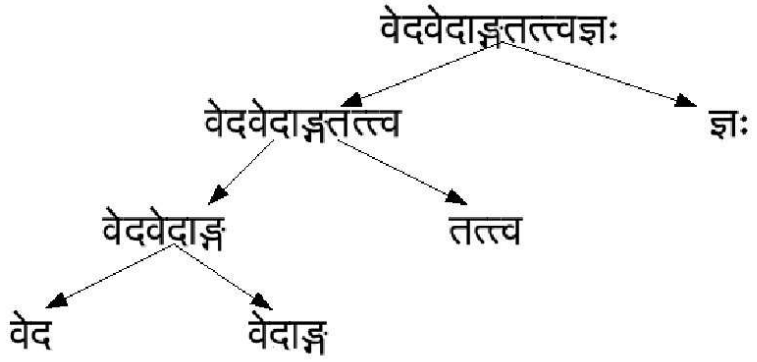


Figure 1: Constituency representation of वेदवेदाङ्गतत्त्वज्ञः

2. प्रवरमुकुटमणिमरीचिमञ्जरीचयचर्चितचरणयुगल[3]=प्रवर - मुकुट - मणि - मरीचि - मञ्जरी - चय - चर्चित - चरण - युगल
3. जलादिव्यापकपृथिवीत्वाभावप्रतियोगिपृथिवीत्ववती[4]=जल - आदि - व्यापक - पृथिवीत्व - अभाव - प्रतियोगि - पृथिवीत्ववती

The Sanskrit compounds are classified semantically into four major types:

- Tatpuruṣaḥ: (Endocentric with head typically to the right),

[2]Rāmāyaṇam 1-1-14
[3]Pañcatantram in kathāmukham
[4]Kevalavyatireki-prakaraṇam : Maṇikaṇa

- Bahuvrīhiḥ: (Exocentric),
- Dvandvaḥ: (Copulative), and
- Avyayībhāvaḥ: (Endocentric with head typically to the left and behaves as an indeclinable).

The compounds are formed with two words at a time with an exception of Dvandva [5] and bahupada bahuvrīhi. Hence compounds can be represented faithfully as a binary tree, as shown in figure 1.

# 3 Compound Processor

There are four tasks involved in identifying the meaning of a compound.

1. **Segmentation**: This involves splitting the sandhi and identifying individual components. eg. splitting तपस्स्वाध्यायनिरतेन as तपस् - स्वाध्याय - निरतेन.

2. **Syntactic Parsing**: This involves getting the parse of a compound depicting its composition, in terms of a tree.
   A Syntactic parser will produce the parse of an input string तपस्स्वाध्यायनिरतेन as <<तपस् - स्वाध्याय>-निरतेन>, or equivalantly as shown in figure 2.

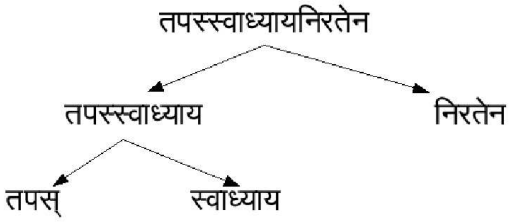


Figure 2: Syntactic Parser

3. **Semantic analysis**: To decide the type of a compound at each node of composition.
   A Semantic analyser would assign labels to each of these nodes as
   <<तपस् - स्वाध्याय>Di-निरतेन>T7
   or as shown in figure 3,

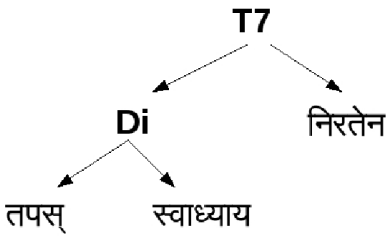


Figure 3: Semantic analysis

   where **T7** and **Di** for the compound labels viz. saptamī-tatpuruṣa and itaretara dvandva.

4. **Paraphrase Generation**: To generate a paraphrase of a compound.
   The Paraphrase generator would paraphrase the given compound as

[5] An example of dvandva compound involving more than two constituents is रामलक्ष्मणभरतेभ्यः (rāmalakṣmaṇabharatebhyaḥ) which means for or from rāma, lakṣmaṇa and bharata.

तपः च स्वाध्यायः च = तपस्स्वाध्यायः (= तत्1)[6]
**gloss:** penance and self-study
(तत्1{7}[7]=) तस्मिन् निरतः = तपस्स्वाध्यायनिरतः(=तत्2)
**gloss:** who is constantly engaged in penance and self-study
(तत्2{3}=) तेन = तपस्स्वाध्यायनिरतेन
**gloss:** By that person who is constantly engaged in penance and self-study

Segmentation needs a good sandhi splitter and morphological analyser.The syntactic parser needs in addition the knowledge about mutual expectancy. The semantic analyser requires a lexical resource rich with semantic information. Pāṇini has given a set of rules metion the range of सूत्रs which help in deciphering the compounds. These rules are being explored further. A tagging scheme to assign appropriate tags to the nodes is evolved(See Appendix 1). Two important aspects for deciphering a compound are: to identify the head and to identify the relation between components. The proposed tags mark both of them. E.g. **T7** stands for compound of type tatpuruṣa and 7 indicates the vibhakti of the first component. Assuming that a manually parsed compound is available, we describe below a paraphrase generator.

## 3.1 Paraphrase generator

A semantically analysed compound has the following syntax.

```
compound: < component - component > tag
component: word | compound
tag: A[1-6]
    | Bs[2-7] | Bs[dgpsu] | Bsm[gn] | Bv[sSU] | B[bv]
    | D[is]   | K[1-5]    | Km      | T[1-7]  | T[bgmnp]
    | Tds     | [ESUd]
word: [a-zA-Z]+
```

Sanskrit compounds tagging syntax

We define a compound formed with the leaf nodes of a binary tree as of level one. Compounds involving at least one component as a compound (i.e. a non-leaf node) are termed as of leaf 2.

### 3.1.1 Paraphrase Generation at level 1

In Sanskrit compounds, as mentioned above, only the last component contains the case suffix[8]. Hence the major task of the paraphrase generator is to decide the gender, number and case suffix of the last component of compound. The paraphrase of a compound varies with the type of a compound. We illustrate below the rules and algorithms for two compounds viz. tatpuruṣa and bahuvrīhi. Appendix I contains rules for generating paraphrases for different compound types.

1. **Algorithm for tatpuruṣaḥ:**
   Input: <x-y>Tn
   where **x** is nominal stem,
   **y** is a noun,
   **T** stands for the tatpuruṣa compound and **n** indicates the case.
   Output: The paraphrase of <x-y>Tn
   Steps:

   (a) Get default gender of x from the lexicon. Let it be $\mathbf{g_1}$.

[6]The pronoun तत् is used as a place holder or a variable. The indices 1,2,.. indicate different place holders.
[7]{7} indicates $7^{th}$ case
[8]with an exception of an aluk samāsaḥ

(b) Get the morphological analysis for y. In case of multiple answers,
use the following heuristics to select the most appropriate analysis.
-In case of different stems and different genders, keep all answers.
-In case of same stems with different genders, select the default gender.
Let the stem and the gender be **y'** and $\mathbf{g_2}$ respectively. Let the case suffix be $\mathbf{c_1}$.

(c) Generate the appropriate forms of individual components, by invoking the generator. Let
ẋ= x<gen=$g_1$> <case=n> <num=1>,
and
ÿ=y' <gen=$g_2$> <case=1> <num=1>

(d) Call the Sentence Generator to generater ẋÿ as the paraphrase of the compound <x-y>$T_n$.

(e) If **y** is not in nominative form, generate the singular form with case $\mathbf{c_1}$ of the $\mathbf{3^{rd}}$ person pronoun in gender $\mathbf{g_2}$.
For example
input=<तपस्स्वाध्याय-निरतेन>T7
x=तपस्स्वाध्याय
y=निरतेन
n=7

**1.** default gender of x is neuter
ẋ=तपस्स्वाध्याय<gender=mesculine><case=7><number=2> = तपस्स्वाध्याययोः

**2.** निरतेन=निरत<gender=neuter><case=3><number=1>
ý=निरत,
$g_2$=neuter,
g=3,
ÿ=निरत<gender=neuter><case=1><number=1> = निरतः

**3.** तेन=तत्<gender=neuter><case=3><number=1>
तपस्स्वाध्याययोः निरतः, तेन=तपस्स्वाध्यायनिरतेन

2. **Algorithm for bahuvrīhi :**
The bahuvrīhi is an exocentric compound. As such the morphological analysis of such compounds pose a special problem. The head of such compounds assume the gender of an object it denotes. For example consider a bahuvrīhi 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 compounds in these cases are tagged as <pīta-ambaraḥ>Bs6 and <pīta-ambarā>Bs6. Now to generate the paraphrase, we require the nominal base of *ambaraḥ* and *ambarā*. But since the dictionary does not have the base form with these genders, an analyser may fail to analyse the head. In order to handle such cases, the morphological analyser has been modified to get the default nominal base in case the word is a post component (*uttarapada*) of a compound. We describe below the algorithm for generation of paraphrase of bahuvrīhi compounds.

**Input:** <x-y>Bsn
where **x** is the nominal stem,
**y** is a noun,
**Bs** stands for the samānādhikaraṇa bahuvrīhi compound and **n** indicates the case.

**Output:** Paraphrase of <x-y>Bsn.

**Steps:**

a) Get the morphological analysis of y.
Let it be stem=y', gender=$g_2$, case=$c_1$.
Get the default gender of y. Let it be $g_1$.

b) Generate
x'=x <gen=$g_1$><num=1><case=1>

y"=y'<gen=$g_2$><num=1><case=1>

c) Let **z** be the $3^{rd}$ person pronoun with gender $g_1$.
Generate

z'=z<gen=$g_1$><num=1><case=$c_1$>

and

z"=z<gen=$g_1$><num=1><case=1>

The paraphrase is **x' y" z' z"**

e.g. <पीत-अम्बरः>Bs6=पीतम् अम्बरं यस्य सः,
<पीत-अम्बरा>Bs6=पीतम् अम्बरं यस्याः सा.

### 3.1.2 Paraphrase Generation for higher level compounds

While generating a paraphrase of a compound involving another compound as a component, decision has to be made as to whether to use the compound or its paraphrase in the generation. To make the point clear consider a 2 level compound

तपस्स्वाध्यायनिरतेन=<तपस्स्वाध्याय-निरतेन>T
=<<तपस्-स्वाध्याय> $Di$–निरतेन>T7

The paraphrase of <तपस्-स्वाध्याय>Di is तपः च स्वाध्यायः च. The paraphrase of the complete compound then can be expressed as

a) तपः च स्वाध्यायः च=तपस्स्वाध्यायौ,
तपसि च स्वाध्याये च निरतः=तपस्स्वाध्यायनिरतः
तेन तपस्स्वाध्यायनिरतेन

or as

b) तपः च स्वाध्यायः च=तपस्स्वाध्यायौ,
तपस्स्वाध्याययोः निरतः=तपस्स्वाध्यायनिरतः
तेन तपस्स्वाध्यायनिरतेन

or as

c) तपः च स्वाध्यायः च=तपस्स्वाध्यायौ(=तत्1),
तस्मिन् निरतः=तपस्स्वाध्यायनिरतः(=तत्2),
तेन तपस्स्वाध्यायनिरतेन

From generation point of view **b** is easier than **a** and **c** is more easier then **b**. We have decided to follow **c**, not only becuase of its simplicity but because of its wide usage in Sanskrit literature as well. The only way we differ from the tradition is in indexing the pronouns if more than one components are present.

Following this approach, we give below the paraphrase of
<<<<<<<प्रवर-मुकुट>Tm-मणि>T6-मरीचि>T6-मञ्जरी>K3-चय>T6-चर्चित>T3-<चरण-युगलः>T6>Bs6
as

प्रवरराजानां मुकुटः = प्रवरमुकुटः (=तत्1),
(तत्1{6}=) तस्य मणिः = प्रवरमुकुटमणिः (=तत्2),
(तत्2{6}=) तस्य मरीचिः = प्रवरमुकुटमणिमरीचिः (=तत्3),
(तत्3{1}=) ते एव मञ्जर्यः = प्रवरमुकुटमणिमरीचिमञ्जर्यः (=तत्4),
(तत्4{6}=) तेषां चयः = प्रवरमुकुटमणिमरीचिमञ्जरीचयः (=तत्5),
(तत्5{3}=) तेन चर्चितम् = प्रवरमुकुटमणिमरीचिमञ्जरीचर्यचर्चितम् (=तत्6),
चरणस्य युगलम् = चरणयुगलम् (=तत्7),
(तत्6{1} तत्7{1}=) तत्6 तत्7 यस्य सः = प्रवरमुकुटमणिमरीचिमञ्जरीचयचर्चितचरणयुगलः

### 3.1.3 Problem Cases and their solutions

- **अलुक् समासः-**
  only the last component of a compound has a case suffix. However as noted earlier there are exceptions typically with certain compounds whose first and intermediate components also have

Vibhaktis. Such compounds are called aluk samāsa. The current tagset of the Sanskrit Consortium does not mark aluk samāsas. In case of paraphrase generation, therefore it is necessary to check whether the pre-component is in the base form or not.

- **मध्यमपदलोपिः-**
  This is a special type of compound in which some of the words in the paraphrase do not occur in its compound form. e.g. Devabrāhmaṇaḥ is a compound whose paraphrase is Devapujakaḥ brāhmaṇaḥ(a brāhmin who worships god). So to get the paraphrase of such compounds mere components are not sufficient. One should also know the context to supply the missing words. We do not handle such compounds in the present implementation. These compounds being very rare not handling them does not affect the overall coverage of the paraphrase generator much.

- **Special cases from Gaṇapāṭa** etc.:-
  Compounds with special paraphrases have been listed by Pāṇini separately in a list. Examples of such compounds are Mayuravaymsakaḥ, Kambojamuṇḍaḥ, Yavanamuṇḍaḥ etc. Each one of them have a special paraphrase. Readymade paraphrases of such compounds are provided.

- **उपपद-समासः-**
  An upapada tatpuruṣa samāsaḥ has a verbal noun (kṛdanta) as a post component(e.g. jnaḥ and kāraḥ in Tattvajnaḥ and Kumbhakāraḥ respectively). These forms are special and occur only as bound forms in a compound. Hence, a special morphological analyser to handle these forms is necessary. The current implementation does not handle these forms as well.

# 4 Evaluation

We have tested 160 examples from Saṅkṣepa Rāmāyaṇam. The performance of tested examples is given below.

| | No. of examples | % |
|---|---|---|
| Correct Paraphrase | 145 | 90.60 |
| Wrong/Incomplete Paraphrase | 15 | 9.40 |

The wrong or incomplete paraphrase is mainly because of the dictionary incompleteness or failure of morph analyser to provide the analysis. A morph analyser that guesses the stem can improve the system further. Another challenge we faced in the generation of paraphrase is related to gender.

Bahuvrihī compounds which are exocentric, typically are adjectives. Hence their paraphrases will change as per the gender of the nouns they qualify. The compound tagging naturally does not code this information, and hence the paraphrase (from the Sanskrit grammar point of view) may be defective. e.g. consider the word Nīlavarṇam as used in

1. idam nīlavarṇaṃ phalam
   (This is a blue coloured fruit.)

2. Nīlavarṇaṃ phalaṃ ānaya
   (Bring a blue coloured fruit.)

3. Nīlavarṇaṃ Krishṇaṃ paśya
   (See the blue coloured Krishna.)

In all these examples Nīlavarṇam is a ṣaṣṭī bahuvrīhi (Bs6) but is in nominative case in (1), and accusative case in (2) and (3), further it is neuter in (1) and (2) and masculine in (3). Hence the paraphrase will be different in all the three cases, viz.

1. नीलं वर्णं यस्य **तत्**, **तत्**[9] फलम्
   (नीलं वर्णं=नीलवर्णम्)

[9]Nominative case

2. नीलं वर्णं यस्य तत्, तत्[10] आनय
   (नीलं वर्णं=नीलवर्णम्)

3. नीलं वर्णं यस्य सः, तं पश्य
   (नीलं वर्णं=नीलवर्णम्)

Thus to generate a correct paraphrase, we need the context and sentence level analysis viz. the adj-noun relation (to get gender of adjective) and also the kāraka role of the noun (to know the vibhakti) in case the word form is ambiguous.

# 5 Future Work

The next task is to develop a syntactic parser and semantic analyser. A syntactic parser requires a sandhi splitter and the semantic analyser needs a lexicon rich with semantic features. It will be very interesting to look at the Pāṇini's rules that govern the semantic aspects of the compounds. There are around 200 rules in the Aṣṭādhyāyī dealing with the semantic aspect of the Sanskrit compounds. It may provide some insight into various kinds of features the lexicon should be enriched with in order to be able to carry out the semantic analysis mechanically.

**Acknowlegement**

We would like to thank K.V. Ramakrishnamacharyulu, K. Narayana Murty, J.S.R.A Prasad, Devanand Shukl and Sheetal Pokar for their valuable guidance and suggestions to improve the system.

This work is being carried out as a part of consortium project entitled "Development of Computational Toolkit and Sanskrit-Hindi Machine Translation System" sponsored by MCIT(DIT).

[10] Accusative case

# References


**[ 1 ]** Bhat G. Mahabaleshvara, *Samāsaḥ*. Samskrita Bharati, Bangalore, Karnataka, 2006.

**[ 2 ]** Oliver Hellwig, *Sanskrit Tagger: A Stochastic Lexical and POS Tagger for Sanskrit*. Proceeding of the International Symposium on Sanskrit Compoutational Linguistics(SCLS-2007), Paris, France, 2007.

**[ 3 ]** Gérard Huet, *Formal Structure of Sanskrit Text: Requirement Analysis for a Mechanical Sanskrit Processor*. Proceeding of the International Symposium on Sanskrit Computational Linguistics(SCLS-2007), Paris, France, 2007.

**[ 4 ]** Gillon Brendan S. , *Exocentric Compounds in Classical Sanskrit*. Proceeding of the First International Symposium on Sanskrit Compoutational Linguistics(SCLS-2007), Paris, France, 2007.

**[ 5 ]** Gillon Brendan S., *Tagging Classical Sanskrit Compounds*. Proceeding of the Third International Symposium on Sanskrit Compoutational Linguistics(SCLS-2009), Hyderabad, India, 2009.

**[ 6 ]** Pandit Ishvarachandra, *Aṣṭādhyāyī*. Chaukhamba Sanskrit Pratisthan, Delhi, 2004.

**[ 7 ]** Girish Nath Jha and Sudhir K. Mishra, *Semantic Processing in Pāṇni's Kāraka System*. Proceeding of the International Symposium on Sanskrit Computational Linguistics(SCLS-2007), Paris, France, 2007.

**[ 8 ]** Amba 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/)

**[ 9 ]** Mahavira, *Pāṇini as Grammarian (With special reference to compound formation)*. Bharatiya Vidya Prakashan [Delhi - Varanasi], India, June 1978.

**[ 10 ]** M. Srimannarayana Murty, *Sanskrit Compounds-A Philosophical Study*. Chowkhamba Sanskrit Series Office, Varanasai(India), 1974.

**[ 11 ]** Vasudeva Lakshman Shastri Paniskar, *Sidhānta Kaumudi*. Meharchand Lachamandas Publications, New-Delhi, 1985.

**[ 12 ]** Peter M. Scharf, *Levels in Pāṇini's Aṣtādhyāyī*. Proceeding of the International Symposium on Sanskrit Compoutational Linguistics(SCLS-2009), University of Hyderabad, Hyderabad, India, 2009.

**[ 13 ]** V. Sheebasudheer, Amba Kulkarni, K.V.Ramakrishnamacharulu , *Guidelines for tagging of Sanskrit Compounds*. prepared for the MCIT(DIT) sponsored project on "Development of Computational Toolkit and Sanskrit-Hindi Machine Translation System".

# 6 Appendix:

**The following table gives the paraphrase of compounds with various tags**

| | |
|---|---|
| | अव्ययीभावः |
| 1 | <x-y>A1 => y{6} f{x}<br>where f maps x to the noun with same semantic content.<br>A function f needs to be defined. |
| 2 | <x-y>A2 => x{3} विपरीतम् वृत्तम् |
| 3 | <x-y>A4 => x{6} y{6} समाहारः |
| 4 | <x-y>A5 => x'{1} y{1} यस्मिन् देशे<br>x' has same gender as that of y'. |
| 5 | <x-y>A6 => x{6} y{6} समाहारः<br>if x = द्वि, both x and y will be in द्विवचनम् |
| 6 | <x-y>A7 => x{6} y |
| | तत्पुरुषः |
| 7 | <x-y>$T_n$ => x{n} y $2 \leq n \leq 7$ |
| 8 | <x-y>Tn => x{n} y |
| 9 | <x-y>Tds => x{6;ba} y{6;ba} समाहारः |
| 10 | <x-y-z>Tb = x{1} y{1} z{1} |
| | कर्मधारयः |
| 11 | <x-y>K1 => x{1} तत् y{1} च |
| 12 | <x-y>K2 => x{1} च y{1} च |
| 13 | <x-y>K3 => x{1} च असौ y{1} च |
| 14 | <x-y>K4 => x{1} इव y{1} |
| 15 | <x-y>K5 => x{1} y{1} इव |
| 16 | <x-y>K6 => x{1} एव y{1} |
| 17 | <x-y>K7 => x{1} इति y{1} |
| | बहुव्रीहिः |
| 18 | <x-y>$BS_n$ => x{1} y{1} यत्{g}{n} where g is the gender of y $2 \leq n \leq 7$ |
| 19 | <x-y>Bsd => x{6} च y{6} च यदन्तरालम् |
| 20 | <x-y>Bsp => x{3} च y{3} च प्रहृत्य इदं युद्धं प्रवृत्तम् |
| 21 | <x-y>Bsg => x{7}-y{7}+ गृहीत्वा इदं युद्धं प्रवृत्तम् |
| 22 | <x-y>Bsmn => x'-y{1} यस्य |
| 23 | <x-y>Bss = > x{1} वा y{1} यस्य |
| 24 | <x-y>Bsu => x{1} इव y{1} यस्याः |
| 25 | <x-y>Bv => x y{1} यस्य |
| 26 | <x-y>Bvs => y{6} x' ये सन्ति ते |
| 27 | <x-y>BvS => y{3} सह |
| 28 | <x-y>BvU => x{6} इव y यस्य |
| | द्वन्द्वः |
| 29 | <x-y>Di => x{1} च y{1} च |
| 30 | <x-y>Ds => x{1} च y{1} च |
| 31 | <x-y>S => y{1} x{1} |
| 32 | <x-y>d => x y |