Sanskrit Compound Generation: With a Focus on the Order of Operations

A dissertation submitted to the University of Hyderabad for the award of the degree of

Doctor of Philosophy

in

Sanskrit Studies

Pavankumar Satuluri

11HSPH04

Department of Sanskrit Studies

School of Humanities

University of Hyderabad

Hyderabad

July 2015

Sanskrit Compound Generation: With a Focus on the Order of Operations

A dissertation submitted to the University of Hyderabad for the award of the degree of

Doctor of Philosophy

in

Sanskrit Studies

by

Pavankumar Satuluri

11HSPH04

under the guidance of

Dr. Amba P. Kulkarni

Department of Sanskrit Studies

School of Humanities

University of Hyderabad

Hyderabad

July 2015

Declaration

I, **Pavankumar Satuluri**, hereby declare that the work embodied in this dissertation entitled " **Sanskrit Compound Generation: With a Focus on the Order of Operations**" is carried out by me under the supervision of Dr. Amba P. Kulkarni, Associate Professor, Department of Sanskrit Studies, University of Hyderabad, Hyderabad and has not been submitted for any degree in part or in full to this university or any other university. I hereby agree that my thesis can be deposited in Shodhganga/INFLIBNET.

A report on plagiarism statistics from the University Librarian is enclosed.

Pavankumar Satuluri

11HSPH04

Date:

Place: Hyderabad

Signature of the Supervisor

Department of Sanskrit Studies

University of Hyderabad, Hyderabad

Certificate

This is to certify that **Pavankumar Satuluri (11HSPH04)** has carried out the research-work embodied in the present dissertation entitled "**Sanskrit Compound Generation: With a Focus on the Order of Operations**" at University of Hyderabad. The dissertation represents his independent work and has not been submitted for any research degree of this university or any other university.

Dr. Amba P. Kulkarni
Supervisor
Associate Professor
Department of Sanskrit Studies
School of Humanities

Dr. J.S.R.A Prasad
Head
Associate Professor
Department of Sanskrit Studies
School of Humanities

Prof. Panchanan Mohanty
Dean
School of Humanities
University of Hyderabad

Acknowledgements

My sincere gratitude to my supervisor *Dr. Amba P. Kulkarni* for her guidance, patience and continuous support throughout my research work. I got an opportunity to do my research under her valuable guidance.

I express my utmost gratitude to *Prof. K. V. Ramakrishnamacharyulu* for their valuable suggestions and guidance throughout my entire research work.

I express my sincere gratitude to *Prof. K. Subrahmanyam* for their continuous support in profound discussions on various research problems.

I express my heartfelt thanks to *Prof. K. N. Murthy* for their continuous support, guidance and encouragements.

I express my sincere gratitude to *Dr. J. S. R. A Prasad* for their constant support and encouragement in my research.

I also convey my regards to my teacher *Dr. B. V. Venkataramana* who gave me valuable insights on my work.

I am indebted to *Dr. Anil Kumar, Sivasenani* for their help during my research.

I thank to all my friends *Arjun, Monali, Dr. Vinaya, Dr. Vani, Dr. Siva, Surendra, Krishna Mohan, Sanjeev, Madhusudan, Krishnamurthy, Gowri* and *Karunakar* for their helping and support.

I am most grateful to my *parents* for their continuous support and encouragement.

I cannot forget office staff of our department, who provided me all kind of infrastructural help. I thank one and all from Department of Sanskrit

Studies office.

I would like to thank one and all, who have directly or indirectly been instrumental in the completion of my research work and thesis.

Contents

List of Figures

Thesis Related Papers Presented at International Conferences

Pavankumar Satuluri and Amba Kulkarni, “Generation of Sanskrit Compounds”. International Conference on Natural Language Processing December 19^{th}-20^{th} 2013. CDAC-Noida.

Pavankumar Satuluri, K. V. Ramakrishnamacharyulu and Amba Kulkarni, “Samāsaprakriyāyāṃ samāsāntānāṃ sthānaṃ, samāsanirdhāraṇe teṣāmupayogaśca” World Sanksrit Conference June 28^{th}-July 2^{nd} 2015. Bangkok, Thailand.

Chapter 1

Motivation

1.1 Importance of Aṣṭādhyāyī

Grammar plays an important role in the study of any language. It helps in the choice of the correct form of words. Pāṇini a great Indian grammarian compiled Aṣṭādhyāyī, a Sanskrit treatise on grammar, circa 4^{th} century BC. Several Grammars for Sanskrit were available prior to Pāṇini. Pāṇini built upon this knowledge of his precursors and compiled them in the form of Aṣṭādhyāyī. The simplicity, thoroughness and the sheer scope of this work, according to Kornai (2008), would place Pāṇini among the greatest landmarks of all intellectual history even if we disregarded the key methodological advances he made. Aṣṭādhyāyī is an extant and almost full fledged grammar covering the then prevalent Sanskrit. To our knowledge this is the only grammar of any natural language with such a wide coverage. The way Pāṇini analysed Sanskrit has attracted several linguists towards it and we see the influence of Pāṇini's grammar since the development stages of the Western linguistics. We mention below a few quotes or the works that motivated us to choose our current topic of research.

Staal (1994)[page:2917] says that

> "Pāṇini, then, was not an ancient and nebulous precursor of a science in which everything has since been done better, but a distant colleague of genius from whom linguists are still able to learn".

Leonard Bloomfield described Pāṇini's Aṣṭādhyāyī as "One of the greatest monuments of human intelligence".

Ferdinand de Saussure who is considered to be the father of modern structural linguistics was influenced by Pāṇini and Bhartṛhari. No wonder then that the western linguistics is influenced by Pāṇini's grammar during its development stage.

The organisation of sūtras, use of 'pratyāhārās', use of 'anubandhās', use of 'anuvṛtti', use of his meta-language and the ordering of rules have attracted the linguists as well as computer scientists.

Cardona (1969) observes that

> "Patañjali explains, the ordering of sounds and the placing of anubandhas are both intended for the formation of pratyāhāras to be used in rules. This point is repeated by various commentators but with one addition; they specifically mention brevity (lāghava)".

Kiparsky (2007)[page:1] states that

> "Completeness and Economy are rightly considered the two main goals of Pāṇini's grammar".

The computer scientists were attracted on the one hand by this algorithmic style, and various methodologies used in Pāṇini, and on the other hand looked at the Aṣṭādhyāyī as a tool for information processing.

Gérard Huet in his inaugural speech at 1^{st} International Sanskrit Computational Linguistics Symposium (2007) said that

> "Not only Pāṇini was by far the first linguist in recorded history, but we claim he was the first informaticien, 24 centuries before computers came into existence".

Bharati and Kulkarni (2010) discuss a set of sūtras from Aṣṭādhyāyī which deal with various aspects of the information coding in sanskrit.

Petersen (2004), a German scholar was intrigued by the arrangement of sounds in the Śiva sūtras and provided a mathematical tool to decide if a set of partial ordered sets can be arranged in a linear order, and if yes, provide the linear order as well. She also proved mathematically that the repetition of the sound *'h'* in the pratyāhāras is optimal, and further tried to provide Śivasūtras for the German sounds as well.

Joshi (2009) says that

> "Pāṇini has intutively used Calculus. Bhattojidīkshita composed the book Siddhantakaumudī based on the Aṣṭādhyāyī which is a generative grammar also known as prakriyā. When we deal with the prakriyā part, it involves various stages. At every stage some transformation, either on the stem of the suffix, takes place and this type of procedure is very similar to an algorithm".

Several researchers have looked at the algorithmic aspect of Pāṇini and tried to develop simulators that will follow Aṣṭādhyāyī sūtra order.

Goyal et al. (2009) developed an Aṣṭādhyāyī simulator. In that they discuss the sūtras *pūrvatrāsiddham* (A8.2.1), *asiddhavadatrābhāt* (A6.4.22) and *ṣatvatukorasiddhaḥ* (A6.1.86) to know how these play a major role in ordering of the sūtras and how Pāṇini dealt with the conflict resolution techniques especially utsarga and apavāda in Aṣṭādhyāyī. This simulator triggers the rules and resolve the conflicts between the sūtras automatically.

Subbanna and Varakhedi (2010) have developed a computational model of Aṣṭādhyāyī based on the principle of asiddhatva. A mathematical concept called 'filter' is newly introduced to deal with all the usages of asiddhatva. Further, they have discussed about the computational aspects and conflict resolution techniques in Aṣṭādhyāyī.

Hyman (2009) observes that the Pāṇinian sūtras for external sandhi can be modeled using a finite state grammar. Hyman designed an XML format in which Scharf coded all the sandhi sutras. This XML formalism has been chosen to express the rules in a machine readable form. Hyman wrote a rule compiler, which translates the rules that are encoded in the XML formalism and converts these into some rewrite rules and those are compiled automatically into a Finite State Transducer (FST) by using some algorithms.

Some other efforts towards computerisation of Aṣṭādhyāyī are by

Dhaval Patel[1] and Swami Taralabalu.[2]

The syntax of Pāṇini's sūtras was found to have striking similarity to the Backus Naur Form (BNF) that is used to describe the syntax of programming languages. BNF and chomsky's hierarchy of formal languages and grammar provided another dimension to look at Pāṇini's Aṣṭādhyāyī.

Bhate and Kak (1992) examined the context sensitive grammar for some rules in Aṣṭādhyāyī such as *iko'yaṇaci* (A6.1.74).

Staal (1965) observes that context free rules were clearly insufficient for Pāṇini's purpose. According to him, Pāṇini's Aṣṭādhyāyī is a mixture of context free and context sensitive rules.

Goal of research

Most of these works focussed on the sandhi formation and inflectional morphology. Sanskrit is also rich in derivational morphology and as such for any meaningful computational work, we need a generator that handles derivational morphology. Three important derivational processes that generate new nominal stems are kṛt, taddhita and samāsa operations. We decided to work on samāsa. We have a keen interest in understanding the concept of compounds at both prakriyā as well as ārthika level. For the usage, prakriyā part is very useful and ārthika level helps us to know the concepts like vṛtti and meaning compatibility (sāmarthya).

[1] http://www.sanskritworld.in/sanskrittool
[2] http://www.taralabalu.org/panini/

Compounds play a vital role in any language. Sanskrit is no exception to it. Compounds provide a compact way of expressing the thoughts. For the languages that are rich in compound formation, compound analysis and generation becomes an important and essential component of any Natural Language Processing system. The reason is, in Sanskrit every sentence consists of one or more compounds. For example: look at the following śloka from Kālidāsa's Raghuvaṃśa

vāgarthāviva sampṛktau vāgarthapratipattaye
jagataḥ pitarau vande pārvatīparameśvarau

It contains seven words out of which four are compounds. We should know the meaning of the compounds in order to understand the meaning of the śloka. This is not an isolated instance of a śloka with several compounds. The analysis of a sample corpus of 400 thousand words show that every fifth word in the corpus is a compound.[3]

In Indian grammatical tradition, there is an abundant discussion of compounds both at the level of *prakriyā* 'the compound formation involving morphology and phonology', and also at the level of *artha* 'meaning analysis'. In recent years, many scholars have worked on compounds on both of these aspects.

The *samarthāhnika* of Joshi (1968) gives a detailed account of the discussion involved in the Indian tradition on the semantic compatibility of

[3]This estimate is based on the corpus developed by the Sanskrit Consortium for the project on 'Development of tools for Analysis of Sanskrit texts and Sanskrit-Hindi Machine Translation System' funded by Deity, Government of India.

components and the compositionality of the meaning of a compound from its components. Further, he discusses the mutual compatibiity of meaning of components and how the principle of gamakatva helps in understanding the meaning of compounds that violate the mutual meaning compatibility (asamartha padavidhi).

Pataskar (1996) has discussed the use of the Dvandva compounds in Pāṇini sūtras in relation to their case endings.

Bhandare (1995) has discussed the structural and semantic aspects of dvandva compound. He discusses the complete process of dvandva compound formation and observes that unlike other compounds Dvandva compounds enjoy some free word order among its components.

Mahavir (1986) talks about the process of generating a compound from its paraphrase (vigrahavākya). He has discussed various transformations that take place on the vigrahavākya to get a compound form. Further, he discusses nityasamāsas and morphophonemic rules in great detail.

The semantic classification of compounds given by Pāṇini is not only restricted to Sanskrit language per say, but is universal. For example the Cambridge grammar of the English Language uses this classification to describe compounds in English.

Compounds are always binary with an exception of conjunctive (Dvandva) and some exocentric (Bahuvrīhi) compounds. Individual components of the compounds can themselves be compounds, thus allowing a recursion

in the formation of compounds. For understanding such compounds the underlying constituency structure is important.

Gillon (2009) proposes an extended phrase structure syntax to represent the underlying constituent structure of the compounds. He examined the analysis of the non constituent compounds and exocentric compounds with the help of constituency grammar. He suggested a tagging scheme for compounds using context free rules, wherein he specifies the vibhakti and identifies the head of the compound and also provides the category of the compound.

Kumar (2012) has described the computational analysis of Sanskrit compounds in his doctoral dissertation. While this system uses Pāṇini's rules for identification of the type and analysis of exceptional compounds, the frequent compound type identification and analysis is done using the statistical properties of the corpus.

Goyal and Huet (2013) describes various morphological phenomena involved in the generation and analysis of avyayībhāva compounds.

We were fascinated by all this literature and decided to look at the process of compound formation in detail. The main reasons for choosing the domain of 'compounds' are:

(a) There is a well laid down procedure for compound formation, with several examples worked out to all minute details.
(b) The sūtras related to the compound formation are of considerable and managable size (around 400 in number).

(c) While most of these sūtras are vidhi sūtras, among them there are niṣedha and adhikāra sūtras as well.

The main goal of our research is to generate Sanskrit compounds automatically from a given analytical paraphrase. This process starts with an alaukikavigraha (analytical paraphrase) and undergoes a series of transformations, resulting into a compound. Then the question rises: up to what extent can the compounds be generated automatically? For that we decided to build a computational model for compound generation, and study those cases where the machine needs some extra linguistic information for processing.

The organisation of thesis

In the first chapter we survey the literature of earlier works with focus on the compounds and computational treatment of Aṣṭādhyāyī, and state the goal of our research work.

As we are aware Pāṇini does not describe the compound formation procedure as such at one place. But later grammarians examined the rules and came up with an order of the operations to form a compound. In the second chapter we describe various steps involved in the entire compound formation following Pāṇinian tradition.

In the third chapter we discuss the issues related to the order of various steps involved in compound formation, especially with reference to the deletion of suffixes and addition of samāsānta sufffix. We discuss what the commentators like Patañjali, Nageśa and Dīkśita have to say on these issues, and explain with examples why Patañjali's treatment is better.

In the fourth chapter, we give the classification of all compound related sūtras based on the type of information each sūtra needs. We noted down the kind of information each sūtra needs for implementation. We observe that in some cases this information is just phonetic or morphological. But there are some sūtras, though less, which need semantic information of the lexical items such as the meaning, the class property, its relation to other word, and so on. In some small number of cases even pragmatic information such as whether the compound is being used to indicate a particular sense or not is needed. This study helps in developing the strategy for computational modeling of the sūtras, and the organisation of lexicon.

In the fifth chapter we study various semantic information needed for the compound formation, and the aim of this study is to arrive at a structured lexicon with all the needed semantic features associated with it. We look at the Amarakośa, and the Nyāya - Vaiśeṣika ontology, and the earlier work of Nair (2011) who enriched Amarakośa with the Nyāya - Vaiśeṣika ontology, and suggest some ammendments in the structure in order to make it useful for our task.

The sixth chapter gives a detail description of the implementation of the compound generation module wise. We have implemented the compound generator using the lexical analyser 'Lex' and a parser generator 'Yacc'. Some of the modules were written as rewrite rules in Yacc, and most of the others are implemented using a lexer. Each module implemented with a lexer is thus a finite state automaton. We have discussed the cases where we had to deviate from Pāṇini's representation in order to 'transcode' them in programming environments of Lex and Yacc. In some cases we had to split a rule into many, in some other cases the rules were better represented as a

function than just pattern matching.

The seventh chapter contains the conclusion part in which we have discussed the limitations and future directions for improving the Sanskrit compound generator.

This implementation has provided us clues for enriching the lexicon. The lexical analyser is sufficient to model the computational complexity of the rules. While implementing these rules we observed that most of the rules fall under the category of regular grammar. In our implementation we observed that most of the rules require a sequence of phonemes. We implemented those rules in Lex (Lexical Analyser). For other rules additional linguistic conditions such as morphological analysis, lexical meaning etc were needed and in some cases extra-linguistic conditions were required. Such rules are implemented using a C code to handle the additional linguistic conditions and an user interface to get the extra linguistic information from the user. There are around 90 rules which require additional information apart from the given input string. For the remaining rules pattern matching is enough to generate the compound.

The second part of the dissertation contains complete code of the implementation.

Chapter 2

Compound Formation : Pāṇinian Procedure

Compounds being an important part of the language, Pāṇini deals with them very thoroughly handling the exceptional cases in great detail. Out of around 4000 sūtras in Aṣṭādhyāyī a little more than 400 sūtras deal with the compound formation. In the first chapter of Aṣṭādhyāyī sūtras from A1.4.61 to A1.4.79 assign a label 'gati' to certain words during compound formation. Similarly sūtras in the 3^{rd} chapter are related to the formation of kṛt derivates. Based on some conditions that some of these terms occur as upapadas during the formation of an upapada compound. Since kṛt formation itself is vast topic. We have not included formation of upapada compounds in our scope of work. Sūtrās starting from A2.1.1 to A2.2.29 are related to the formation of a compound (*vidhāyaka sūtras*). Sūtras from A2.2.30 to A2.2.38 are related to the change of word order (*pūrvanipāta*) in the compound formation process. In the 5^{th} chapter sūtras from A5.4.68 to A5.4.160 introduce special suffixes related to the compounds called

samāsānta pratyayas. The 3^{rd} section of the 6^{th} chapter has several sūtras that deal with the various stages of compound formation. Sūtrās from A6.3.1 to A6.3.24 prohibit the deletion of the case suffix, The sūtras which transform the first constituent and second constituent of a compound start from 1^{st} and 3^{rd} section of the 6^{th} chapter.

The most relevant paper for our work is by Mahavir (1986). In this paper he discusses the treatment of compounds in Pāṇini. In the first section he discusses the mutual compatibility (*sāmarthya*) between the constituents of the compound. He gives the justification for the order in which Pāṇini deals with different compound types. He observes that if Avyayībhāva class were placed after Tatpuruṣa or Bahuvrīhi then it would cease to be an exception of either Tatpuruṣa or Bahuvrīhi and would result in the generation of non-existent irregular forms.

In the second section he talks about the various transformations a vigrahavākya undergoes resulting into a compound form.

These transformations correspond to various operations such as
(a) "Fixation of the order of the constituents.
(b) Assignment of the prātipadika (nominal stem) designation to this string.
(c) Morpho-phonemic changes in pūrvapada and uttarapada.
(d) Change of gender and number in certain types, and
(e) Change or retainment of original accent".

We describe below the steps involved in compound formation process followed in the Pāṇinian tradition.

1. 'Analytical Paraphrase' (*Alaukikavigraha*)

   Assume that a speaker wants to express the concept 'a person who is a servant of a king'. Then he has three basic conceptual elements viz. a person, a king and the servant-master relation between them. Now this servant-master relation (sevya-sevaka bhāva) in Sanskrit is expressed by ṣaṣṭhī vibhakti (6^{th} case suffix). So the concept is represented in the form of a linguistic expression as

   rājan + ṅas puruṣa + su

   where the words *rājan* and *puruṣa* denote the objects 'a king' and 'a person', and the suffix *ṅas* represents the relation between these two elements, and 'su' is the singular nominal case suffix. This representation is called an alaukikavigraha and is the starting point of the derivation of a compound. This alaukikavigraha is the paraphrase of the compound showing the linguistic analysis and thus is the analytic paraphrase of the compound.

2. (a) Samāsa designation: In the above example, since the components rājan and puruṣa have mutual meaning compatibility, if the speaker has an intention to form a compound, Pāṇinian sūtra *ṣaṣṭhī P2.2.8* sanctions the formation of such a compound. Such an alaukikavigraha then gets a **Samāsa** designation. The compound formation is purely under the dictum of the desire of a speaker (*vaktṛvivakṣādhīnam*). In the derivation process, corresponding to an alaukikavigraha the speaker has an option to produce either a sentential expression or a compound expression.[1] Of

[1]Strictly speaking, there are certain compounds which are termed as nityasamāsas

course, only if the language and hence the grammar allows such formation, a speaker can form a compound. The choice is governed by a condition that a compound may be formed only if the components involved are mutually compatible.[2] In other words, there should be some rule in the grammar, which guarantees the formation of a compound with given components and meanings.

(b) Introduction of *samāsānta pratyaya*s:

There are certain suffixes which are added to the compound at the end. These are called *samāsānta pratyaya*s. These are added at the end of an alaukikavigraha and are treated as a part of a compound. For example the sūtra *avyayībhāve śaratprabhṛtibhyaḥ* (A5.4.107) adds a *ṭac* suffix if the compound is of type Avyayībhāva and one of the word from the list starts with *śarad*. Thus

śarad + ṅas upa $\rightarrow$ *śarad + ṅas upa tac*.

3. Declaring the string to be a nominal stem (*prātipadikasaṃjñā*)

   At this point, now the stage is set to generate a new nominal stem (prātipadikam) from this expression and thus the whole expression is designated with a label prātipadikam[3].

4. Deciding the word order (*upasarjanasaṃjñā*)

   The head (*viśeṣya*) of a compound more-or-less depends on the type

(avigrahaḥ asvapadavigraho vā nityasamāsaḥ - PMa (I:568)) which do not have a sentential paraphrase or do not have a paraphrase in terms of its own constituents. Such compounds are governed by the sūtras falling in the range A2.1.5 to A2.1.10 and A2.2.17 to A2.2.20. Barring these, all the compounds can optionally be expressed as a sentential paraphrase as well.

[2] samarthaḥ padavidhiḥ (A2.1.1)

[3] *kṛttaddhitasamāsāśca* (A1.2.46)

of a compound. The Tatpuruṣa (endocentric) compound has second component as the head while the Avyayībhāva has the first component as the head. In case of Bahuvrīhi (exo-centric) the head is the one which this Bahuvrīhi compound modifies and in case of conjunctive compounds, both the words together are heads. However there are many exceptions to this general observation. So the semantic criterion for head does not help us in deciding the word order. Pāṇini uses a special mechanism to mark the order of words in a compound. In Pāṇini's grammar he labels the component which occupies the first place by the term *upasarjana*. The designator of a word in nominative case in the rule that decides the compound type is the one which gets the designation *upasarjana*. For example consider the input

śarad+ ṅas upa

The sūtra[4] which sanctions compound formation for this input consists of two words. Among these the word *avyayam* in the sūtra is in nominative. Hence that word in the alaukikavigraha which is an *avyaya* gets a designation upasarjana. In the above example *upa* is an avyaya, and therefore *upa* gets a designation upasarjana.

5. Change of word order (*pūrvanipātaḥ*)

   The consequence of assigning a label upasarjana is to place the word having been designated as upasarjana in the 1^{st} place. This action is termed as pūrvanipātaḥ. In the above example, '*upa*' which got upasarjanasaṃjñā occupies the first place. This changes the order of the input string to

   upa śarad + ṅas.

[4] *avyayam vibhaktisamīpasamṛddhivyṛddhyarthābhāvātyayāsampratiśabdaprādurbhāvapaścādyathānupūrvyayaugapadyasādṛśyasampattisākalyāntavacaneśu* (A2.1.6)

6. Deletion and Non-deletion of case suffixes (*Subluk* and *aluk kārya*)

   There are certain exceptional cases, when the case suffix of the involved words does not get elided[5]. For example in the following case

   yudh +ṅi [6] *sthira + su*

   ṅi is retained. But in the case of

   rājan+ṅas puruṣa+su

   ṅas is deleted. In either case the case suffix of the second word is deleted, This will result in the above cases becoming

   yudh +ṅi sthira and *rājan puruṣa*.

7. Transformation of the first component

   In certain cases, the first component undergoes certain transformations in the presence of the second component. For example *pāda* changes to *pad*[7] if it is followed by either *āji|āti|ga* or *upahata*.

   pāda + āti $\rightarrow$ *pad + āti*

8. Transformation of the second component

   In certain cases, the second component also undergoes certain changes in the presence of the first component. For example *sthira* changes to *ṣṭhira*[8] in the presence of *yudhi*.

   yudhi + sthira $\rightarrow$ *yudhi ṣṭhira*

9. Padakārya

   Each of the components thus transformed obligatorily undergo certain operations which are independent of the other words in the context. For example in the case of *rājan* 'n' gets deleted.[9]

[5] *aluguttarapade* (A6.3.1)
[6] *haladantatsaptamyāḥ saṃjñāyām* (A6.3.9)
[7] *pādasya padājyātigopahateṣu* (A6.3.52)
[8] *gaviyudhibhyāṃ sthiraḥ* (A8.3.95)
[9] *nalopaḥ prātipadikāntasya* (A8.2.7)

10. Joining of the words (*sandhikārya*)

    The two components thus transformed, now are joined together with a sandhi operation. For example

    $$rāma\ ālaya \rightarrow rāmālaya$$

11. Deciding the gender of a derived nominal stem (*liṅganirdhāraṇam*)

    Such a stem is now all set to undergo word formation rules. But before that, this stem should get the proper gender. Gender of a nominal lexicon is the property of the lexical item[10]. An Avyayībhāva compound is always in neuter gender[11]. While in the case of Tatpuruṣa (endocentric) and Dvandva (conjunctive) compounds[12] the gender of the compound is same as the second component. In the case of Bahuvrīhi (exocentric), the gender is same as that of the word it modifies. There are of course a few exceptional cases which are dealt with special rules by Pāṇini.

12. Deciding the number

    The number of the resulting compound depends on its constituents. In the case of Tatpuruṣa the number depends on the primary component and in Bahuvrīhi it depends on the referent of the resulting compound. Avyayībhāva compound has singular number and in the case of itarataradvandva, the number of the resulting compound depends on the nature and number of the components involved. In samāhāradvandva the number will be singular.

13. Deciding the svara (accent)

    A compound is characterized by a single accent. Usually the fi-

[10] ekaṃ dvikaṃ trikaṃ cātha catuṣkaṃ pañcakaṃ tathā
nāmārtha iti sarve'mī pakṣāḥ śāstre nirūpitāḥ - VBhs(nāmārthanirṇayaḥ 216 (1)

[11] *avyāyībhāvaśca* (A2.4.18)

[12] *paravalliṅgam dvandvatatpuruṣayoḥ* (A2.4.26)

nal vowel in a compound is udātta (high pitch)[13]. Tatpuruṣa and Avyayībhāva compounds have a high pitch at the end (antodātta). Bahuvrīhi, and Dvandva compounds will have the same accent as that of the pūrvapada[14]. These are the general rules and there are some exceptions to them.

14. Word Formation

    The final step is the word formation, with appropriate vibhakti.

[13] *samāsasya* (A6.1.223)

[14] *bahuvrīhau prakṛtyā pūrvapadam* (A6.2.1)
rājanyabahuvacanadvandvāndhakavṛṣṇiṣu (A6.2.34)
saṅkhyā (A6.2.35)
ācāryopasarjanasyāntevasī (A6.2.36)
kārtakaujapādayaśca (A6.2.37)

Chapter 3

Theoretical Issues in Compound Formation

In the previous chapter we saw various steps involved in the process of compound formation. There are differences of opinion regarding the order in which these steps are to be carried out. In this chapter, we discuss the issues related to the order of various steps involved, and the consequences thereof.

Typical process of compound formation dealt with by the teachers is sketched below. Pāṇinīyavyākaraṇodāharaṇakośa shows this process for thousands of compounds. The generation process of the compound starts with alaukikavigraha, which gets the designation 'samāsa' followed by another designation 'prātipadika'. Then aluk related operations are performed. This is followed by the elision of suffixes. Then the word order related operations viz. designation of upasarjanasaṃjñā followed by pūrvanipāta take place. Additional suffix called samāsāntā sufffix is added at this point.

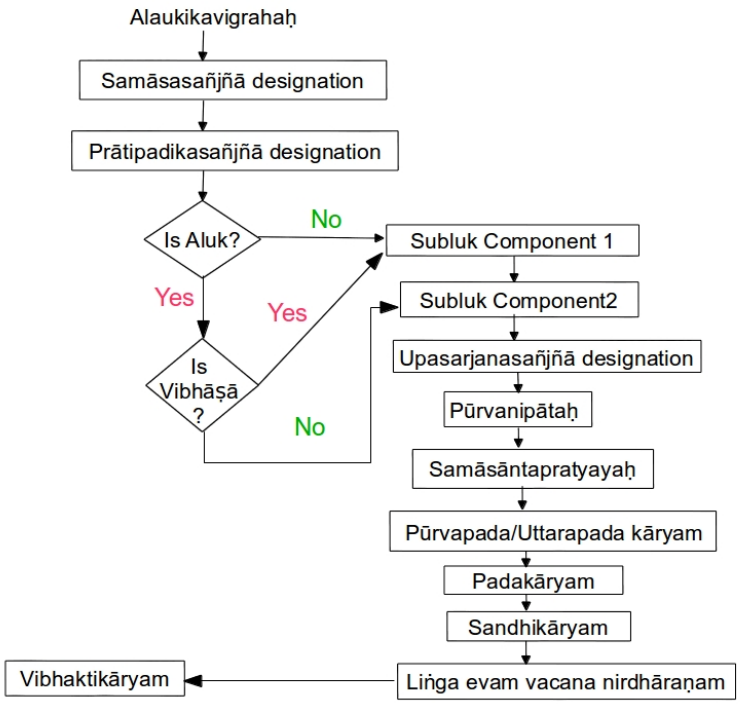

Figure 3.1: Process of Compound Generation (1)

This is then followed by pūrvapadakārya, uttarapadakārya, sandhikārya and finally the gender is decided and appropriate vibhakti is added to get the compound word. This process is shown as a flow chart in Figure 3.1

There are differences of opinion with this order, especially with reference to the deletion of suffixes and addition of samāsānta suffix.

- When should vibhakti suffix (sup) be elided?
  Should it be elided before the designation of upasarjana saṃjñā or after?

- At what stage should the samāsānta suffix be added ?
  Should we add this suffix at the starting point of the samāsa prakriyā or after deciding the word order of the components ?

Let us see what the commentators have to say on the samāsānta suffix and the arguments in favour and against the deletion of nominal suffix (subluk) before and after the assignment of upasarjanasaṃjñā.

3.1 When should subluk take place?

If one applies the subluk before assigning the upasarjana saṃjñā then it creates a problem. The case suffix is the basis of information for deciding upasarjana saṃjñā. Hence once this is deleted, there is no way to decide which of the components will get an upasarjanasaṃjñā.

We illustrate this with an example. Consider the alaukikavigraha rājan + ṅas puruṣa + su. The word rājan + ṅas gets the designation upasarjana. The governing sūtra is *ṣaṣṭhī* (A2.2.8). The sūtra *prathamā nirdiṣṭaṃ samāsa upasarjanam* (A1.2.43) says that in the samāsavidhāyakasūtra (Asūtra which dictates the type of the resulting compound), the word which is in nominative case will get designated as upasarjana. Now in the sūtra *ṣaṣṭhī* (A2.2.8), there is only one word, and it is in nominative. Hence the word in genitive case in the alaukikavigraha viz. rājan gets designated as upasarjana.

Now if the case suffix is deleted first, then in the absence of the genitive suffix, it is not possible to designate rājan an upasarjana saṃjñā. This is the objection. In response, some argue that with the help of a sūtra *pratyayalope pratyayalakshaṇam* (A1.1.62)(an operation conditioned by an affix applies

even if the affix has been replaced by zero[1]), we can designate it as upasarjana even after the subluk. There is no bādhyabādhakabhāva between the two operations - subluk and upasarjana.

However, as is evident from this discussion even if there is no bādhyabādhakabhāva, in order to designate upasarjana saṃjñā after the deletion of case suffixes, one needs to invoke another sūtra *pratyayalope pratyayalakshaṇam* (A1.1.62). This is a gaurava (prolixity). If we designate the word with an upasarjanasaṃjñā based on the case suffix and then we delete it, we need not invoke *pratyayalope pratyayalakshaṇam* (A1.1.62). This brings in lāghava (brevity).

3.2 Meaning of the sūtra samāsāntāḥ (A5.4.68)

Pāṇini has introduced 21 suffixes called samāsānta suffixes in the Aṣṭādhyāyī from 5.4.68 to 5.4.160. They are: (1) ḍac (2) a (3) ac (4) ahnādeśa (5) ṭac (6) ṣac (7) ṣa (8) ap (9) ac (10) asic (11) anic(12) ic (13) jñu (14) anaṅ (15) niṅ (16) it (17) lopa (18) datṛ (19) lopa (20) hṛt (21) kap

The sūtra *samāsāntāḥ* (A5.4.68) says that "The taddithа affix introduced, henceforth, occur at the end (anta) of a nominal stem termed compound (samāsa)"[2]. Patañjali interprets the word samāsa in the sūtra *samāsantāḥ* (A5.4.68) as an analytical paraphrase (alaukikavigrahaḥ) and Bhaṭṭoji Dīkṣita interprets the word samāsa as an analytical paraphrase as well as an uttarapada.

[1] *The Aṣṭādhyāyī of Pāṇini Vol. II Ramanath Sharma, Page No.63, 2000*
[2] *The Aṣṭādhyāyī of Pāṇini Vol. IV Ramanath Sharma, Page No.704, 2000*

3.3 At what stage should the samāsānta suffix be added?

We present below the interpretation offered by Bhaṭṭoji, followed by Patañjali and Nāgeśa Bhaṭṭa.

Bhaṭṭoji Dīkṣita in his book Praudhamanoramā[3] writes

> Samāsāntā iti |Samāsasya samāsārthottarapadasya vā caramāvayava ityarthaḥ |tenopaśaradamityatra "nāvyayībhāvāt" (A2.4.83) ityam, dvipurītyādau "dvigoḥ" (A4.1.21) iti ṅīp ca sidhyati |

According to Dīkṣita the samāsānta suffix can become a part of either the analytical paraphrase or an uttarapada. According to him, there are cases in which one needs to treat the samāsānta suffix as a part of the complete paraphrase and cases where one needs to consider the samāsānta suffix as a part of an uttarapada. We explain both these cases with an example each.

Example 1:

alaukikavigrahaḥ : upa + su śarad + ṅas

samāsānta pratyayāḥ : [upa + su śarad + ṅas [ṭ]a[c]]

where [ṭ] and [c] are 'it' markers, and the whole expression gets the samāsasaṃjñā.

prātipadikasaṃjñā : 'upa + su śarad + ṅas a' , now it gets prātipadikasaṃjñā

upasarjanasaṃjñā : **upa** + su śarad +ṅas a, upa gets upasarjana saṃjñā

[3]PMa A5.4.68 (I.577-578)

pūrvanipātaḥ : upa +su śarad + ṅas a

subluk : upa śarad a

sandhikāryam : upaśarada

liṅganirdhāraṇam : upaśarada , neuter gender

vibhaktikāryam : upaśarada + su $\rightarrow$ upaśaradam, due to the replacement of 'a' by 'am'.

Now instead, if the samāsānta suffix is added to the uttarapada the situation will be

alaukikavigrahaḥ : [upa + su śarad + ṅas]

samāsānta pratyayāḥ : [upa + su (śarad + ṅas] [ṭ]a[c]) where [ṭ] and [c] are 'it' markers. upa + su śarad + ṅas gets samāsasaṃjñā and ṭac is not part of the samāsasaṃjñā. But śarad + ṅas ṭac is an uttarapada. We have marked the expression that gets samāsasaṃjñā by [ ] and an expression in ( ) is the uttarapada. Note here that 'a' of the samāsānta suffix is now part of the uttarapada but not of the samāsa.

prātipadikasaṃjñā : [upa + su (śarad + ṅas] a)

upasarjanasaṃjñā : [**upa** + su (śarad +ṅas] a)

pūrvanipātaḥ : [upa +su (śarad + ṅas] a)

subluk : [upa (śarad] a)

sandhikāryam : [upa(śarad] a)

liṅganirdhāraṇam : [upa (śarad] a), neuter gender

vibhaktikāryam : [upa(śarad] a)

Thus here 'a' of [ṭ]a[c] is part of uttarapada but is not a part of the samāsa. And hence the replacement of 'a' by 'am' is prohibited.

Now we look at another example, which Bhaṭṭoji Dikśita used to show the necessity of considering samāsānta suffix as a part of uttarapada.

Example 2:

alaukikavigrahaḥ : [dvi + os pur + os]

samāsānta pratyayāḥ : [dvi + os (pur + os] a), As in the previous case, we mark the boundaries of samāsa saṃjñā by [ ] and of uttarapada by ( ).

prātipadikasaṃjñā : [dvi + os (pur + os] a)

upasarjanasaṃjñā : [**dvi** + os (pur + os] a)

pūrvanipātaḥ : [dvi + os (pur + os]a)

subluk : [dvi (pur] a)

sandhikāryam :[dvi(pur] a)

liṅganirdhāraṇam : [dvi(pur] a), feminine gender

vibhaktikāryam : [dvi(pur] a)→ dvipura + ṅīp → dvipurī.

In this example the samāsānta suffix ‘a’ is a part of an uttarapada. Hence by the vārtika ‘akārāntottarapado dviguḥ striyām bhāṣyataḥ iti vaktavyam’ the feminine suffix will be added to the entire compound. The vārtika says that in a dvigu compound if the uttarapada is a prātipadika ending in ‘a’, then the feminine suffix ṅīp is added to the compound. On the other hand if we add the samāsānta suffix to the entire compound then the uttarapada ‘pur’ does not end in ‘a’. So there is no scope for the addition of feminine suffix.

From these two examples, it is obvious that in one example, to arrive at the desired form one needs to accept that the samāsānta suffix as a part of the alaukikavigraha while in the other, as a part of the uttarapada. But if

we look at Patañjali's commentary, then it is obvious that he subscribes to only the first view viz. that the samāsānta suffix is a part of complete alaukikavigraha, and not the uttarapada.

The discussion on the status of samāsānta pratyayas in Mahābhāṣya appears under the commentary of the sūtra *'gostriyorupasarjanasya'* (A1.2.48). In the case of a compound formation 'bahu + jas kumārī + jas' there is a scope to introduce a samāsānta suffix 'kap'. There is also a possibility of shortening of the vowel ī in kumārī. In the desired form there is no shortening of the vowel ī. So Patañjali discusses which sūtra prohibits the shortening of the vowel ī, and why.

First Patañjali rules out the possibility that the shortening of the vowel ī is prohibited due to the vārtika 'kapi ca'. In fact he questions the very necessity of this vārtika and claims that this vārtika is not needed and the sūtra *'na kapi'* (A7.4.14) should operate. In such cases the objection that *'na kapi'* (A7.4.14) is applicable only to prohibit the results of *'ke'ṇaḥ'* (A7.4.13) is ruled out by him and he interprets *'na kapi'* (A7.4.14) to prohibit vowel shortenings in all the cases in the presence of a 'kap' suffix.

Now the question is when both the vowel shortening as well as introduction of samāsānta suffix 'kap' are possible, which operation takes prcedence? An operation being conditioned by an inside cause or word-internal is antaraṅga and an operation having an outside cause or word-external is bahiraṅga. The operation of vowel shortening requires the prātipadikam to be feminine. Since this condition is word-internal, this is an antaraṅga operation. The addition of samāsānta suffix 'kap' takes place provided the

compound is of type Bahuvrīhi and the second component of compound ends in either 'ī' or 'ū' (which is designated as nadī). This is a bahiraṅga operation. Following the paribhāṣā *asiddham bahiraṅgamantaraṅge* the vowel shortening should take place first. But Patañjali calls addition of 'kap' as an antaraṅgataraḥ, and hence the samāsānta suffix is added first, and then the sūtra *'na kapi'* (A7.4.14) will prohibit the vowel shortening.

Patañjali further says that the words that are to be compounded will wait for the samāsānta suffix because it is a part of the compound.[4]
Similarly, in the commentary on Prauḍhamanoramā, we note the following observation under the commentary on *samāsāntāḥ* (A5.4.68).

> evañcālaukikavigrahavākye samāsasaṁjñā samakālameva samāsānta iti siddhāntaḥ[5] |

Nageśa reinforces[6] the interpretation of Bhāṣyakāra Patañjali that in sūtras *'samāsāntāḥ'* (A5.4.68) and *'antaḥ'* (A6.2.92), the samāsānta suffix is added at the end of an analytical paraphrase and becomes a part of the compound and then the entire string gets the designation samāsa.

So, based on the above discussions we can say that the samāsānta suffix is a part of an analytical paraphrase (alaukikavigraha) only, and the alaukikavigraha along with the samāsānta suffix gets the designation samāsa. Dīkṣita's view on samāsānta suffix as a part of an uttarapada is not

[4]tādarthyāttācchabdyaṁ bhaviṣyati. yeṣām padānām samāsaḥ na tāvatteṣāmanyadbhavati kapaṃ tāvat pratīkṣate ||MBh A1.2.48 (I.57)

[5]PMa A5.4.68 (I.578)

[6]samāsaśabdena tadarthamuttarapadaṁ lakṣyata iti bhāva iti kaścit ||vastutastu samāsārthaṁ vigrahavākyaṁ lakṣyata iti bhāṣyāśayastathaivānyatra dṛṣṭatvāt MBh A1.2.48 (I.57)||

acceptable. The reason is there is no such evidence (pramāṇa) that tells us the word samāsa in samāsānta as a uttarapada. Dīkṣita has extended the meaning of the word samāsa in samāsānta as a uttarapada by lakṣaṇa to explain the correct forms such as dvipurī, dvidhurī. But Bhāṣyakāra does not give this explanation. Then the question may arise how does he handle the cases like dvipurī etc. Patañjali rejects the vārtika 'akārāntottarapado dviguḥ striyām bhāṣyataḥ iti vaktavyam[7]', because in the sūtra 'dvigoḥ' (A4.1.21) he says that the feminine suffix is added to the entire dvigu compound which ends in 'a'. So, there is no need to add the 'a' suffix to uttarapada based on the vārtika. Therefore, the samāsānta suffix becomes a part of the analytical paraphrase only.

In the implementation of the compounds we follow the Bhāṣyakāra's interpretation i.e the samāsānta suffix is a part of the analytical paraphrase.

Now the generation process of the compounds which is inferred from the traditional commentators like Patañjali, Nageśa etc, starts with the alaukikavigraha, and then samāsāntā suffix is added (additional suffixes), and it gets the designation samāsa, such a samāsa gets the designation prātipadika saṃjñā, after that aluk (in the case of aluk compounds), upasarjana saṃjñā, pūrvanipāta, subluk, transformation of the components (pūrvapadakārya and uttarapadakārya, sandhikārya, liṅganirdhāraṇam and finally vibhaktikāryam) are carried out to get the compound word.

Figure 3.2 shows the process of the generation of the compounds, implemented by us.

[7]MBh A2.4.30 (II.486)

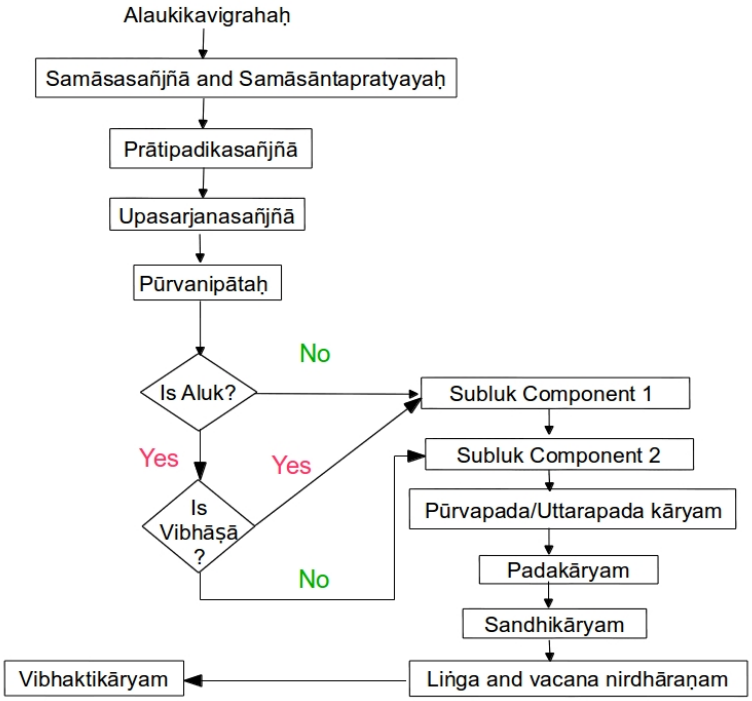

Figure 3.2: Process of Compound Generation (2)

Chapter 4

Analysis and Classification of Sūtras

Aṣṭādhyāyī has around 400 sūtras that deal with the compound formation. As we noticed in previous chapters, the compound formation process is completely algorithmic. So the questions that arose in our mind were

1. Is it possible to automate the process of compound formation?
2. If yes, how many sūtras are completely formal, and how many of them can't be automated?
3. In case of sūtras which can't be automated, what kind of information is needed for processing?

In order to answer these questions, we classified the sūtras based on the operation they correspond to in the compound formation. Then, we looked at each sūtra and analysed them to find out what kind of information is needed by the sūtra for its application.

While classifying the sūtras according to their function, we observed

the following:

1. Most of the sūtras are conditional expressions specifying certain action if a certain condition is satisfied. For example:

   pādasya padājyātigopahateṣu A6.3.52

   This sūtra says if the first constituent is *pāda* and the second constituent is either *āti*, *ga* or *upahata* then *pāda* changes to *pad*.

2. There are some sūtras which require a rich structure involving deeper analysis than just the sequence of phonemes. We could classify them in four types.

(a) The morphological analysis of a word

Some sūtras require morphological information of components, for example whether the component is a word in a feminine gender with a suffix ṅīp, or whether it is a primary derivative with a *'kta'* or *'lyap'* and so on. For example the rule *'svayaṃ ktena'* A2.1.25 says that if the first word is 'svayam' and the second word ends in a primary derivative suffix *'kta'* then a compound may be formed.

(b) The semantic category of a word

Many sūtras need semantic information to form a compound. For example: The sūtra *'ktenāhorātrāvayavāḥ'* A2.1.45 says that if the first constituent is a word denoting part of either day or night and if it is followed by a word with primary derivative suffix *'kta'* then a compound may be formed. Here the information that the word is a part of day/night is semantic in nature.

(c) Some sūtras require pragmatic information. For example the sūtra

'ṣaṣṭyā ākrośe' A6.3.21 says that in a ṣaṣṭhī tatpuruṣa compound if a word which is in genetive case indicates the sense ākroṣa (censure) then the non-elision of case suffix is ordained. As is evident, the information whether the word is being used with certain emotions can not even be associated with a lexicon and needs to be obtained only through the context. Hence this type of information is pragmatic by nature.

(d) Finally Pāṇini has listed a set of compounds which are exceptional cases. There are a few of such lists such as *'tiṣṭadguprabṛtīni ca'* A2.1.17, *'mayūravyaṃsakādayaśca'* A2.1.72 etc. These lists are called gaṇapāṭhas.

Thus we observed that for computational implementation of the sūtras, we need

- A list of gaṇapāṭhas,
- A lexicon tagged with semantic information, and
- A morpholocal analyser to analyse the components.

The sūtras which need pragmatic information are difficult to process automatically.

We have classified all the compound related sūtras based on the type of information each sūtra needs. Following tables list all the sūtras, grouped under the operation they correspond to. Agianst each sūtra we have noted down what kind of information each sūtra needs.

In this chapter we discuss the semantic and pragmatic information needed for the implementation.

samāsavidhāyakakārya	
Input needed for implementation	**sutras**
gaṇapāṭhaḥ	tiṣṭhadguprabhṛtīni ca 2.1.16
	yājakādibhiśca 2.2.9
	saptamīśauṇḍaiḥ 2.1.40
	pātresamitādayaśca 2.1.48
	upamitaṃ vyāghrādibhiḥ sāmānyāprayoge 2.1.56
	śreṇyādayaḥ kṛtādibhiḥ 2.1.59
	kumāra śramaṇādibhiḥ 2.1.70
	mayūravyaṃsakādayaśca 2.1.72
	sākṣātprabhṛtīni ca 1.4.74
	ūryādicviḍācaśca 1.4.60
Semantic category	
saṃkhyā	akṣaśalākāsaṃkhyāḥ pariṇā 2.1.10
	saṅkhyā vaṃśyena 2.1.18
	saṅkhyāpūrvo dviguḥ 2.1.51
guṇavacana	tṛtīyā tatkṛtārthena guṇavacanena 2.1.29
nadīvācakāḥ	nadībhiśca 2.1.19
vyañjana	annena vyañjanam 2.1.33
bhakṣya	bhakṣyeṇa miśrīkaraṇam 2.1.34
dūrārtha	stokāntikadūrārthakṛcchrāṇi ktena 2.1.38

pūraṇa/guṇa/avyaya	pūraṇa-guṇa-suhitārtha-sadavyaya-tavya-samānādhikaraṇena 2.2.11
parimāṇa	kālāḥ parimāṇinā 2.2.5
sañjñā	sañjñāyām 2.1.44
	anyapadārthe ca sañjñāyām 2.1.20
ahorātrāvayava	ktenāhorātrāvayavāḥ 2.1.45
dik/saṅkhya/sañjñā	diksaṅkhye sañjñāyām 2.1.50
jāti	katarakatamau jātipariprašne 2.1.63
	poṭāyuvati 2.1.65
	kṛtyatulyākhyā ajātyā 2.1.68
praśaṃsāvacana	praśaṃsāvacanaiśca 2.1.66
varṇa	varṇo varṇena 2.1.69
catuṣpād	catuṣpādo garbhiṇyā 2.1.71
saṅkhyā evam avyayā	saṅkhyāvyayāsanna-dūrādhika-saṅkhyāḥ saṅkhyeye 2.2.25
dik	diṅnāmānyantarāle 2.2.26
prāṇi/tūrya/senāṅga	dvandvaśca prāṇitūryasenāṅgānām 2.4.2
caraṇa	anuvāde caraṇānām 2.4.3
jati/aprāṇi	jātiraprāṇiṇām 2.4.6
nadī/deśa/grāmāḥ	viśiṣṭaliṅgo nadīdeśo'grāmāḥ 2.4.7
kṣudrajantava	kṣudrajantavaḥ 2.4.8
adhvaryukratuḥ	adhvaryukraturanapuṃsakam 2.4.4

vṛkṣa/mṛga/tṛṇa/dhānya/vyañjana	vibhāṣā vṛkṣa-mṛga-tṛṇa-dhānya-vyañjana-paśuśakunyaśvavadava-pūrvāparādharottarāṇām 2.4.12
āyāmaḥ	yasya cāyāmaḥ 2.1.15
sādṛśya	yathā'sādṛśye 2.1.7
kṣepa	khaṭvā kṣepe 2.1.25
	kṣepe 2.1.47
	kiṃ kṣepe 2.1.64
atyantasaṃyoga	atyantasaṃyoge ca 2.1.28
adhikārthavacana	kṛtyairadhikārthavacane 2.1.32
ṛṇa	kṛtyairṛṇe 2.1.43
upamāna	upamānāni sāmānyavacanaiḥ 2.1.55
pūjā	ktena ca pūjāyām 2.2.12
	sanmahatparamottamotkṛṣṭāḥ pūjyamānaiḥ 2.1.61
	vṛndārakanāgakuñjaraiḥ pūjyamānam 2.1.62
adhikaraṇavācī	adhikaraṇavācinā ca 2.2.13
	vipratiṣiddhaṃ cānadhikaraṇavāci 2.4.13
ekadeśī	pūrva-parādharottaram-ekadeśinaikādhikaraṇe 2.2.1

Semanticosynatic information is needed	
karma	karmaṇi ca 2.2.14
kartari	tṛjakābhyāṃ kartari 2.2.15
	kartari ca 2.2.16
kartṛkaraṇe	kartṛkaraṇe kṛtā bahulam 2.1.31
Pragmatics	
kutsita	kutsitāni kutsanaiḥ 2.1.53
	pāpāṇake kutsitaiḥ 2.1.54
No extra semantic information needed for implementation	
saha supā 2.1.4	
svayaṃ ktena 2.1.24	
sāmi 2.1.26	
kālāḥ 2.1.27	
pañcamī bhayena 2.1.36	
apetāpoḍha-mukta-patitāpatrastairalpaśaḥ 2.1.37	
ṣaṣṭhī 2.2.8	
nityaṃ krīḍājīvikayoḥ 2.2.17	
na nirdhāraṇe 2.2.10	
ardhaṃ napuṃsakam 2.2.2	
dvitīya-tṛtīya-caturtha-turyāṇyanyatarasyām 2.2.3	
prāptāpanne ca dvitīyayā 2.2.4	
siddhaśuṣkapakvabandhaiśca 2.1.41	
dhvāṅkṣeṇa kṣepe 2.1.42	
tatra 2.1.46	
ktena nañviśiṣṭenānañ 2.1.60	

yuvākhalatipalitavalinajaratībhiḥ 2.1.67
kugatiprādayaḥ 2.2.18
tatropapadaṃ saptamīstham 3.1.92
upapadamatiṅ 2.2.19
amaivāvyayena 2.2.20
tṛtīyāprabhṛtīnyanyatarasyām 2.2.21
ktvā ca 2.2.22
anekamanyapadārthe 2.2.24
tatra tenedamiti sarūpe 2.2.27
tena saheti tulyayoge 2.2.28
cārthe dvandvaḥ 2.2.29
yeṣāṃ ca virodhaḥ śāśvatikaḥ 2.4.9
taddhitārthottarapadasamāhāre ca 2.1.52
īṣadakṛtā 2.2.7
lakṣaṇenābhipratī ābhimukhye 2.1.13
yāvadavadhāraṇe 2.1.8
suppratinā mātrārthe 2.1.9
anuryatsamayā 2.1.14
adhyayanato'viprakṛṣṭākhyāyām 2.4.5

samāsāntāḥ	
Input needed for implementation	**sutras**
gaṇapāṭhaḥ	avyayībhāve śaratprabhṛtibhyaḥ 5.4.107
	pādasyalopo'hastyādibhyaḥ 5.4.138
	kumbhapadīṣu ca 5.4.139
	dvidaṇḍyādibhyaśca 5.4.128
	uraḥ prabhṛtibhyaḥ kap 5.4.151
Semantic category	
svāṅgavācī	nāḍītantryoḥ svāṅge 5.4.159
	bahuvrīhau sakthyakṣṇoḥ svāṅgātṣac 5.4.113
prāṇivācī	upamānādaprāṇiṣu 5.4.97
janapadavācī	brahmaṇo janapadākhyāyām 5.4.104
saṅkhyeyavācī	bahuvrīhau saṅkhyeye ḍajabahugaṇāt 5.4.73
sañjñāvācī	aññāsikāyāḥ sañjñāyāṃ nasaṃ cāsthūlāt 5.4.118
saṅkhyāvācī	tatpuruṣasyāṅguleḥ saṅkhyāvyayādeḥ 5.4.86
kṣepavācī	kimaḥ kṣepe 5.4.70
jāti evam sañjñāvācī	ano'śmāyassarasāṃ jātisaṃjñayoḥ 5.4.94

sañjñāvācī	vā sañjñāyām 5.4.133
	na sañjñāyām 5.4.155
	agrākhyāyāmurasaḥ 5.4.93
	striyāṃ sañjñāyām 5.4.143
karmavyatihārah	ickarmavyatihāre 5.4.127
āyāmavācī	anugavamāyāme 5.4.83
upamāna	upamānācca 5.4.137
saṅkhyā	saṅkhyā supūrvasya 5.4.140
vayaḥ	vayasi dantasya datṛ 5.4.141
parvata	trikakutparvate 5.4.147
Pragmatics	
pūjā	na pūjanāt 5.4.69
adarśanam	akṣṇo'darśanāt 5.4.76
vandanam	vandite bhrātuḥ 5.4.157
mitra/amitra	suhṛd durhṛdau mitrāmitrayoḥ 5.4.150
avasthā	kakudasyāvasthāyāṃ lopaḥ 5.4.146
lubdhayogaḥ	dakṣiṇermā lubdhayoge 5.4.126
No extra semantic information needed for implementation	
uttaramṛgapūrvācca sakthnaḥ 5.4.98	
anaśca 5.4.108	
napuṃsakādanyatarasyām 5.4.109	
nadīpaurṇamāsyāgrahāyaṇībhyaḥ 5.4.110	
jhayaḥ 5.4.111	
gireśca senakasya 5.4.112	

ahassarvaikadeśa saṅkhyātapuṇyācca rātreḥ 5.4.87
rājāhaḥ sakhibhyaṣṭac 5.4.91
grāmakauṭābhyāṃ ca takṣṇaḥ 5.4.95
ateḥ śunaḥ 5.4.96
nāvodvigoḥ 5.4.99
khāryāḥ prācām 5.4.101
dvitribhyāmañjaleḥ 5.4.102
kumahadbhyāmanyatarasyām 5.4.105
nadyṛtaśca 5.4.153
aṅgulerdāruṇi 5.4.114
dvitribhyāṃ ṣa mūrdhnaḥ 5.4.115
antarbahirbhyāṃ ca lomnaḥ 5.4.117
upasargācca 5.4.119
suprātasuśvasudiva 5.4.120
nañ duḥsubhyo halisakthyoranyatarasyām 5.4.121
nityamasicprajāmedhasoḥ 5.4.122
dharmādanickevalāt 5.4.124
jambhāsuharitatṛṇasomebhyaḥ 5.4.125
prasaṃbhyāṃ jānunorjñuḥ 5.4.129
ūrdhvādvibhāṣā 5.4.130
dhanuṣaśca 5.4.131
jāyāyā niṅ 5.4.134
gandhasyedutpūtisusurabhibhyaḥ 5.4.135
vibhāṣā śyāvārokābhyām 5.4.144

agrānta-śuddha-śubhra-vṛṣa-varāhebhyaśca 5.4.145
udvidbhyāṃ kākudasya 5.4.148
pūrṇādvibhāṣā 5.4.149
inaḥ striyām 5.4.152
īyasaśca 5.4.156
niṣpravāṇiśca 5.4.160
dvandvāccudaṣahāntātsamāhāre 5.4.106
ṛkpūrabdhūḥ pathāmānakṣe 5.4.74
acpratyanvavapūrvātsāmalomnaḥ 5.4.75
acaturavicatura 5.4.77
brahmahastibhyāṃ varcasaḥ 5.4.78
avasamandhebhyastamasaḥ 5.4.79
anvavataptādrahasaḥ 5.4.81
praterurasaḥ saptamīsthāt 5.4.82
dvistāvātristāvā vediḥ 5.4.84
gorataddhitaluki 5.4.92
alpākhyāyām 5.4.136
upasargādadhvanaḥ 5.4.85
nañastatpuruṣāt 5.4.71
patho vibhāṣā 5.4.72
śeṣādvibhāṣā 5.4.154

pūrvanipātaḥ	
Input needed for implementation	**sutras**
gaṇapāṭhaḥ	rājadantādiṣu param 2.2.31
	vā"hitāgnyādiṣu 2.2.37
	kaḍārāḥ karmadhāraye 2.2.38
	upamitaṃ vyāghrādibhiḥ sāmānyāprayoge 2.1.56
Semantic category	
viśeṣaṇavācī	saptamīviśeṣaṇe bahuvrīhau 2.2.35
upamānavācakāh	upamānāni sāmānyavacanaiḥ 2.1.55
No extra semantic information needed for implementation	
upasarjanaṃ pūrvam 2.2.30	
dvandve ghi 2.2.32	
ajādyadantam 2.2.33	
alpāctaram 2.2.34	
niṣṭhā 2.2.36	
pūrvakālaika-sarva-jarat-purāṇa-nava-kevalāḥ samānādhikaraṇena 2.1.49	
pūrvāpara-prathama-carama-jaghanyasamāna-madhya-madhyamottama-vīrāśca 2.1.58	
yuvā-khalati-palita-valina-jaratībhiḥ 2.1.67	

aluk samāsaḥ	
Input needed for implementation	**sutras**
Pragmatics	
ākroṣa	ṣaṣṭhyāḥ ākrośe 6.3.21
Semantic category	
saṃñāvācī	haladantātsaptamyāḥ sañjñayām 6.3.9
	manasaḥ saṃñāyām 6.3.4
	vaiyākaraṇākhyāyāṃ caturthyāḥ 6.3.7
	parasya ca 6.3.8
kāravācī	kāranāmni ca prācāṃ halādau 6.3.10
svāṅgavācī	amūrdhamastakāt svāṅgādakāme 6.3.12
pūraṇavācī	ātmanaśca 6.3.6
kālavācī	śayavāsavāsiṣvakālāt 6.3.18
kāla	ghakālataneṣu kālanāmnaḥ 6.3.17
vidyāvācī/yonivācī	ṛto vidyā-yoni-sambandhebhyaḥ 6.3.23

No extra semantic information needed for implentation
pañcamyāḥ stokādibhyaḥ 6.3.2
ojaḥ saho'mbhas-tamasas-tṛtīyāyāḥ 6.3.3
ājñāyini ca 6.3.5
madhyād gurau 6.3.11
bandhe ca vibhāṣā 6.3.13
prāvṛṭcharatkāladivāṃ je 6.3.15
vibhāṣa vṛkshaśaravarāt 6.3.16
sthe ca bhāṣāyām 6.3.20
putre'nyatarasyām 6.3.22
nensiddhabadhnātiṣu ca 6.3.19
vibhāṣā svasṛpatyoḥ 6.3.24
tatpuruṣe kṛti bahulam 6.3.14

pūrvapadakārya	
Input needed for implementation	**sutras**
vṛddhi	idvṛddhau 6.3.28
	vṛddhinimittasya ca taddhi-tasyāraktavikāre 6.3.39
Semantic category	
devatāvācī	ānaṅṛto dvandve 6.3.25
	devatādvandve ca 6.3.26
	īdagneḥ somavaruṇayoḥ 6.3.27
	divo dyāvā 6.3.29
	uṣāsoṣasaḥ 6.3.31
sañjñā/pūraṇavācī	sañjñāpūraṇyośca 6.3.38
svāṅgavācī	svāṅgāccetaḥ 6.3.40
jātivācī	jātesca 6.3.41
kālavācī	avyayībhāve cākāle 6.3.81
pūraṇa/priyādi	striyāḥ puṃvat bhāṣitapuṃskādanūṅ samānādhikaraṇe striyāmappūraṇīpriyādiṣu 6.3.34

No extra semantic information needed for implementation
vopasarjanasya 6.3.82
vibhaṣā catvāriṃśatprabhṛtau sarveśām 6.3.49
trestrayaḥ 6.3.48
dvyaṣṭanaḥ saṃkhyāyāmabahuvrīhyaśītyoḥ 6.3.47
puṃvatkarmadhāraya-jātīya-deśīyeṣu 6.3.42
ānmahataḥ samānādhikaraṇajātīyayoḥ 6.3.46
divasaśca pṛthivyām 6.3.30
mātarapitaravudīcām 6.3.32
tasilādiṣvākṛtvasucaḥ 6.3.35
kyaṅmāninośca 6.3.36
na kopadhāyāḥ 6.3.37

uttarapadakārya	
Input needed for implementation	**sutras**
gaṇapāṭhaḥ	kṣubhnādiṣu ca 8.4.39
Semantic category	
sañjñāvācī	pūrvapadātsañjñāyāmagaḥ 8.4.3
saṅkhyā	na saṅkhyādeḥ samāhāre 5.4.89
No extra semantic information needed for implementation	
ahno'hna etebhyaḥ 5.4.88	
uttamaikābhyāṃ ca 5.4.90	
upasargācca 5.4.119	
ahno'dantāt 8.4.7	
upasargād bahulam 8.4.28	
puragāmiśrakā-sidhrakā-sārikā-kotarāgrebhyaḥ 8.4.4	
pranirantaḥśarekṣu-plakṣāmra-kārṣya-khadira-pīyūkṣābhyo' saṃjñāyāmapi 8.4.6	
jyotirāyuṣaḥ stomaḥ 8.3.83	

samāsāśraya prakaraṇam	
Input needed for implementation	**sutras**
gaṇapāṭhaḥ	vanagiryoḥ sañjñāyāṃ koṭarakiṃśulukādīnām 6.3.117
	matau bahvaco'najirādīnām 6.3.119
	śarādīnāṃ ca 6.3.120
Semantic category	
ṛk vācakāḥ	ṛcaḥ śe 6.3.55
sañjñāvācī	udakasyodaḥ sañjñāyām 6.3.57
	ṅyāpoḥ sañjñāchandasorbahulam 6.3.63
	sahasya saḥ sañjñāyām 6.3.78
	aṣṭanaḥ sañjñāyām 6.3.125
	nare sañjñāyām 6.3.129
pūrayitavyavācī	ekahalādau pūrayitavye'nyatarasyām 6.3.59
granthavācī	granthāntādhike ca 6.3.79

anumeyaḥ	dvitīye cānupākhye 6.3.80
caraṇavācī	caraṇe brahmacāriṇi 6.3.86
jātivācī	tṛṇe ca jātau 6.3.103
	kustumburuṇi jātiḥ 6.1.143
ṛṣivācakāḥ	mitre cārṣau 6.3.130
Morph analysis is needed	
gharūpa-kalpa-celaḍ-bruva-gotra-mata-hateṣu ṅyo'nekāco hrasvaḥ 6.3.43	
ugitaśca 6.3.45	
hṛdayasya hṛllekhayadaṇlāseṣu 6.3.50	
vā śokaṣyañrogeṣu 6.3.51	
iko hrasvo'ṅyo gālavasya 6.3.61	
ṣyaṅaḥ samprasāraṇaṃ putrapatyostatpuruṣe 6.3.13	
rātreḥ kṛti vibhāṣā 6.3.72	
tīrthe ye 6.3.87	
vibhāṣodare 6.3.88	
dṛgadṛśavatuṣu 6.3.89	

No extra semantic information needed for implementation
nadyāḥ śeṣasyānyatarasyām 6.3.44
himakāṣihatiṣu ca 6.3.54
vā ghoṣamiśraśabdeṣu 6.3.56
peṣaṃvāsavāhanadhiṣu ca 6.3.58
manthaudana-saktu-bindu-vajrabhāra-hāra-vīvadhagāheṣu ca 6.3.60
pādasya padājyātigopahateṣu 6.3.52
iṣṭakeṣīkamālānāṃ citatūlabhāriṣu 6.3.65
kāre satyāgadasya 6.3.70
samānasya cchandasyamūrdhaprabhṛtyudarkeṣu 6.3.84
jyotirjanapada 6.3.85
idaṅkimorīśkī 6.3.90
aṣaṣṭhyatṛtīya 6.3.99
koḥ kattatpuruṣe'ci 6.3.101
rathavadayośca 6.3.102
kā pathyakṣayoḥ 6.3.104
īṣadarthe 6.3.105
vibhāṣā puruṣe 6.3.106
kavaṃ coṣṇe 6.3.107
karṇe lakṣaṇasyā 6.3.115
nahivṛtivṛṣivyadhirucisahitaniṣu kvau 6.3.116
vale 6.3.118
iko vahe'pīloḥ 6.3.121
upasargasya ghañyamanuṣye bahulam 6.3.122
ikaḥ kāśe 6.3.123
citeḥ kapi 6.3.127

aparasparāḥ kriyāsātatye 6.1.144
goṣpadaṃ sevitāsevitapramāṇeṣu 6.1.145
āspadaṃ pratiṣṭhāyām 6.1.146
āścaryamanitye 6.1.147
varcaske'vaskaraḥ 6.1.148
apaskaro rathāṅgam 6.1.149
viṣkiraḥ śakunau vā 6.1.150
pratiṣkaśaśca kaśeḥ 6.1.152
praskaṇvahariścandrāvṛṣī 6.1.153
maskaramaskariṇau veṇuparivrājakayoḥ 6.1.154
kāstīrājastunde nagare 6.1.155
kāraskaro vṛkṣaḥ 6.1.156

liṅganirdhāraṇakārya	
Input needed for implementation	**sutras**
gaṇapāṭhaḥ	ardharcāḥ puṃsi ca 2.4.31
	na dadhipayaādīni 2.4.14
	gavāśvaprabhṛtīnī ca 2.4.11
Semantic category	
jātivācī	jātyākhyāyām-ekasmin-bahuvacanam-anyatarasyām 1.2.48
niravasitāḥ	śūdrānāmaniravasitānām 2.4.10
vṛkṣa/mṛga/tṛṇa/dhānya	vibhāṣā vṛkṣa-mṛga-tṛṇa-dhānya-vyañjana-paśuśakunyaśvavadava-pūrvāparādharottarāṇām 2.4.12
adhikaraṇavāci	vipratiṣiddhaṃ cānadhikaraṇavāci 2.4.13
	adhikaraṇaitāvattve ca 2.4.15
sañjñāyā vācī	sañjñāyāṃ kanthośīnareṣu 2.4.20
prāṇi/tūrya/senāṅga	dvandvaśca prāṇitūryasenāṅgānām 2.4.2
caraṇa	anuvāde caraṇanām 2.4.3
adhvaryukratu	adhvaryukraturanapuṃsakam 2.4.4
viprakṛṣṭākhyāyā	adhyayayanato'viprakṛṣṭākhyāyām 2.4.5
jāti/prāṇivācī	jātiraprāṇinām 2.4.6
nadī/deśa/grāmāḥ	viśiṣṭaliṅgonadīdeśo'grāmāḥ 2.4.7

kṣudrajantavaḥ	kṣudrajantavaḥ 2.4.8
nakṣatra vācākāḥ	phalgunīproṣṭhapadānāñca nakṣatre 1.2.60
No extra semantic information needed for implementation	
paravalliṅgaṃ dvandvatatpuruṣayoḥ 2.4.26	
pūrvavadaśvavaḍavau 2.4.27	
rātrāhnāhāḥ puṃsi 2.4.29	
apathaṃ napuṃsakam 2.4.30	
asmado dvayośca 1.2.49	
sa napuṃsakam 2.5.17	
upajñopakramaṃ tadādyācikhyāsāyām 2.4.11	
sabhā rājā'manuṣyapūrvā 2.4.23	
vibhāṣā samīpe 2.4.1	
aśālā ca 2.4.24	
vibhāṣā senāsurācchāyāśālāniśānām 2.4.25	
chāyā bāhulye 2.4.22	
yeṣāṃ ca virodhaḥ śāśvatikaḥ 2.4.9	

vacananirdhāraṇakārya	
Input needed for implementation	**sutras**
gaṇapāṭhaḥ	na dadhipayaādīni 2.4.14
Semantic category	
jāti vācī	jātyākhyāyām-ekasmin-bahuvacanam-anyatarasyām 1.2.48
adhikaraṇavāci	vipratiṣiddhaṃ cānadhikaraṇavāci 2.4.13
prāṇi tūryasenāṅgānāḥ	dvandvaśca prāṇitūryasenāṅgānām 2.4.2
caraṇa	anuvāde caraṇanām 2.4.3
adhvaryukratu	adhvaryukraturanapuṃsakam 2.4.4
viprakṛṣṭākhyāyā	adhyayayanato'viprakṛṣṭākhyāyām 2.4.5
jāti evam prāṇi vācī	jātiraprāṇinām 2.4.6
nadī deśa evam grāmāḥ	viśiṣṭaliṅgonadīdeśo'grāmāḥ 2.4.7
nakṣatra vācākāḥ	phalgunīproṣṭhapadānāñca nakṣatre 1.2.60
Pragmatics	
niravasitāḥ	śūdrānāmaniravasitānām 2.4.10

gatikārya	
Input needed for implementation	**sutras**
Semantics	
anukaraṇavācī	anukaraṇaṃ cānitiparam 1.4.61
ādara/anādara	ādarānādarayoḥ sadasatī 1.4.62
bhūṣaṇe	bhūṣaṇe'lam 1.4.63
aparigraha	antaraparigrahe 1.4.64
śraddhā/pratīghāta	kaṇemanasi śraddhāpratīghāte 1.4.65
anupadeśa	ado'nupadeśe 1.4.69
antardhi	tiro'ntardhau 1.4.70
upayamana	nityaṃ haste pāṇāvupayamane 1.4.77
bandhana	prādhvaṃ bandhane 1.4.78
aupamya	jīvikopaniṣadāvaupamye 1.4.79
No extra semantic information needed for implementation	
purovyayam 1.4.66	
astaṃ ca 1.4.67	
accha gatyarthavadeṣu 1.4.68	
vibhāṣā kṛñi 1.4.72	
upāje'nvāje 1.4.73	
anatyādhāna urasimanasī 1.4.75	
madhye pade nivecane ca 1.4.76	

Chapter 5

Semantic and Pragmatic Information Needed for Compound Generation

In the previous chapter we noted down various kind of extra linguistic information needed for the implementation of sūtras. This extra linguistic information falls under two different categories viz semantic and pragamatic. The semantic type of information is the one which can be associated with the lexicon and is context independent, while the pragmatic information is context dependent. To give an example, "a component in the sense of distance (dūrārtha)" is a semantic condition, whereas "referent of a component is respectable" is a pragmatic condition because the referent depends on the context.

In order to provide the necessary semantic information with lexicon, we noticed that Pāṇini uses several semantic features. Those are

nadīvācī, deśavācī, grāmavācī, vidyāvācī, yonivācī, devatāvācī, vyañjanavācī, bhakṣyavācī, parvatavācī, ṛṣivācī, nakṣatravācī, saṅkhyāvācī, saṅkhyeyavācī, pūraṇavācī, catuṣpādvācī, varṇavācī, kṣudrajantu, vṛkṣavācī, tṛṇavācī, dhānyavācī, ṛṇavācī, janapadavācī, granthavācī, jāti, guṇavācī, dūrārthavācī, parimāṇavācī, dikvācī, kālavācī.

These features fall under the following sub-categories

(a) Meaning

(b) Synonym / antonym

(c) Part-whole relation

(d) Modifier-modified relation

(e) Generic property / Universal it belongs to

There have been several efforts to classify the objects in the real world based on the metaphysical properties of these objects. This classification is known as ontological classification. Some of the important efforts in this direction in the West are Roget's thesaurus, CYC[1] and SUMO[2].

In India the importance of such a classification had been recognized long-back by the Vaiśeṣikas. This classification is only a top level classification classifying the objects into seven classes viz dravya, guṇa, karma, sāmānya viśeṣa, samavāya and abhāva and each of these are further subclassified. However this classification needs to be further extended so as to classify all the real objects showing the hierarchical classification. But in order to extend the Vaiśeṣika classification, we need to understand the concept of generic property / universal (jāti).

[1] http://www.cyc.com/

[2] http://www.adampease.org/OP/

The sūtra *'samāna prasavātmikā jātiḥ* N2.2.71' about jāti says that Universal is that which produces the same conception. Cowhood is an universal which resides in all cows. Assuming that if a person sees a cow somewhere then he acquires a general form of cows. This general form enables him to recognise individual cows, anywhere in the world. Any general form such as an X-ness can't be considered to be a jāti. Because this will lead to multiple inheritance.

Udayana, the great Naiyāyika, has discussed about the six impediments (jātibādhakas) in Nyāyasiddhantamuktāvali to decide whether a given general property is an universal or not.

The six impediments are described in Muktāvali as

> "vyakterabhedastulyatvaṃ saṅkaro'thā'navasthitiḥ
> rūpahānirasambandho jātibādhakasaṃgrahaḥ"

Vyakteḥ abhedaḥ (Oneness of the individual), tulyatvam (Identity), saṅkaraḥ (cross-division), anavasthitiḥ (Infinite Regression), rūpahāniḥ (Violation of Nature) and asambandhaḥ (Non relation) are the six impediments (jātibādhakāḥ).

Vyakteḥ abhedaḥ (Oneness of the individual) : When there is a single entity of that kind, then the property of resident in only that object can't be an Universal. For example in Nyāya-Vaiśeṣika system ākāśa is only one hence ākāśatva cannot be a jāti.

Tulyatvam (Identity) : If two general properties occur in the same individuals, then these two properties are equivalent and hence only one among

these two will be considered to be a jāti. For example ghaṭa and kalaśa refer to the same set of objects. Hence only one of them will be a jāti.

Saṅkara (Cross-division) : Physicalness (bhūtatva) and mortality (mūrtatva) are not universals because they involve cross-division. Bhūtatva is the universal property that exists in five elements.The five elements are pṛthvī (earth), jala (water), tejas (fire) and vāyu (air) and ākāśa. The first four also have mūrtatva (mortality). Apart from these four elements, the manas (mind) is also mūrta and hence it has mūrtatva. Now the question is whether the general properties mūrtatva and bhūtatva both represent a genre / class character (jāti) or not? Udayana puts a constraint that among the genre there should not be a saṅkara. In other words a padārtha should not have two generic properties residing in them. The four elements pṛthvī (earth), jala (water), tejas (fire) and vāyu (air) have two universals (bhūtatva and mūrtatva) in them and this leads to saṅkara, and hence both mūrtatva and bhūtatva can't be considered to be jātis simultaneously. This condition is an essential characterisation of single inheritance in the Western terminology.

Anavasthitiḥ (Infinite regress): A universal shouldn't contain another universal, as it may lead to anavasthitiḥ (Infinite regress).

Rūpahāniḥ (Violation of Nature) : Differentials, cannot have an universal, like differential-ness (viśeṣatva). If we consider viśeṣatva as an identity or universal, then mutual difference would be there.

Asambandhaḥ (Non relation): Inherence cannot have a universal, like inherence-ness (samavāyatva). Inherence is a relation which holds universal in a particular. Assume that if inherence-ness is regarded as a universal, then inhering in inherence (samavāya) will lead to infinite regress.

So an abstract property of an object can be a jāti provided it does not fall under any of these impediments. Any generic property that has any one of the above faults (jātibādhakās) will then be termed as an upādhi.

The Nyāya-Vaiśeṣika school of philosophy in Indian tradition has dealt with the ontological classification of the real objects which is accepted to all the branches in Indian knowledge system. All the words in the Amarakośa were classified following and extending this scheme further by Nair (2011). To this classification, she also added various other properties, called as *upādhi*s a word may have other than its ontological status. For example to be an eatable, or to be drinkable are the special properties which can not be classified under the ontology without violating the essential property of single inheritance. We examined all the semantic conditions found in all the sūtras related to compound generation. Some ontological categories (*jāti*s as well as *upādhi*s) which are used by Pāṇini were missing in the tables developed by Nair, since these were basically prepared using the lexicon from Amarakośa. We added these missing categories to the table of ontological classification[3] (Figure 5.1) and the missing properties to the table of upādhis (Figure 5.2). The lexicon enriched with the ontological and special properties belonging to the upādhi chart is used for the compound generation.

[3]The words which have '*' sign in the figures (5.1 and 5.2) indicate that these are newly introduced by us over and above Sivaja's classification

5.1 List of jātis needed for Compound Generation

nadīvācī

nadībhiśca 2.1.19

viśiṣṭaliṅgonadīdeśo'grāmāḥ 2.4.7

deśavācī

viśiṣṭaliṅgonadīdeśo'grāmāḥ 2.4.7

grāmāvācīḥ

viśiṣṭaliṅgonadīdeśo'grāmāḥ 2.4.7

devatāvācī

ānaṅṛto dvandve 6.3.25

devatādvandve ca 6.3.26

īdagneḥ somavaruṇayoḥ 6.3.27

divo dyāvā 6.3.29

uṣāsoṣasaḥ 6.3.31

ṛṣi

mitre cārṣau 6.3.130

nakṣatravācī

phalgunīproṣṭhapadānāñca nakṣatre 1.2.60

saṅkhyāvācī

akṣaśalākāsaṃkhyāḥ pariṇā 2.1.10

saṅkhyā vaṃśyena 2.1.18

saṅkhyāpūrvo dviguḥ 2.1.51

saṅkhyāvyayāsanna-dūrādhika-saṅkhyāḥ saṅkhyeye 2.2.25

tatpuruṣasyāṅguleḥ saṅkhyāvyayādeḥ 5.4.86

saṅkhyā supūrvasya 5.4.140

na saṅkhyādeḥ samāhāre 5.4.89

diksaṅkhye sañjñāyām 2.1.50

saṅkhyeyavācī

bahuvrīhau saṅkhyeye ḍajabahugaṇāt 5.4.73

pūraṇa

ātmanaśca 6.3.6

sañjñāpūraṇyośca 6.3.38

pūraṇa-guṇa-suhitārtha-sadavyaya-tavya-samānādhikaraṇena 2.2.11

striyāḥ puṃvat bhāṣitapuṃskādanūṅ samānādhikaraṇe striyāmappūraṇīpriyādiṣu 6.3.34

catuṣpādvācī

catuṣpādo garbhiṇyā 2.1.71

varṇavācī

varṇo varṇena 2.1.69

kṣudrajantu

kṣudrajantavaḥ 2.4.8

vṛkṣavācī

vibhāṣā vṛkṣa-mṛga-tṛṇa-dhānya-vyañjana-paśuśakunyaśvavadava-pūrvāparādharottarāṇām 2.4.12

tṛṇavācī

vibhāṣā vṛkṣa-mṛga-tṛṇa-dhānya-vyañjana-paśuśakunyaśvavadava-pūrvāparādharottarāṇām 2.4.12

janapada

brahmaṇo janapadākhyāyām 5.4.104

grantha

granthāntādhike ca 6.3.79

kāla

ktenāhorātrāvayavāḥ 2.1.45

śayavāsavāsiṣvakālāt 6.3.18

avyayībhāve cākāle 6.3.81

svāṅga

amūrdhamastakāt svāṅgādakāme 6.3.12

svāṅgāccetaḥ 6.3.40

nāḍītantryoḥ svāṅge 5.4.159

bahuvrīhau sakthyakṣṇoḥ svāṅgātṣac 5.4.113

jāti

katarakatamau jātipariprаśne 2.1.63

poṭāyuvati... 2.1.65

kṛtyatulyākhyā ajātyā 2.1.68

jātiraprāṇinām 2.4.6

jātyākhyāyām-ekasmin-bahuvacanam-anyatarasyām 1.2.48

ano'śmāyassarasāṃ jātisaṃjñayoḥ 5.4.94

jātesca 6.3.41

tṛṇe ca jātau 6.3.103

kustumburuṇi jātiḥ 6.1.143

jātyākhyāyām-ekasmin-bahuvacanam-anyatarasyām 1.2.48

jātiraprāṇinām 2.4.6

guṇavācī

tṛtīyā tatkṛtārthena guṇavacanena 2.1.29

dūrārtha

stokāntikadūrārthakṛcchrāṇi ktena 2.1.38

parimāṇa

kālāḥ parimāṇinā 2.2.5

dik

diṅnāmānyantarāle 2.2.26

prāṇi

dvandvaśca prāṇitūryasenāṅgānām 2.4.2

upamānādaprāṇiṣu 5.4.97

caraṇa

anuvāde caraṇānām 2.4.3

āyāmavācī

anugavamāyāme 5.4.83

ṛk vācakāḥ

ṛcaḥ śe 6.3.55

pūrayitavya vācī

ekahalādau pūrayitavye'nyatarasyām 6.3.59

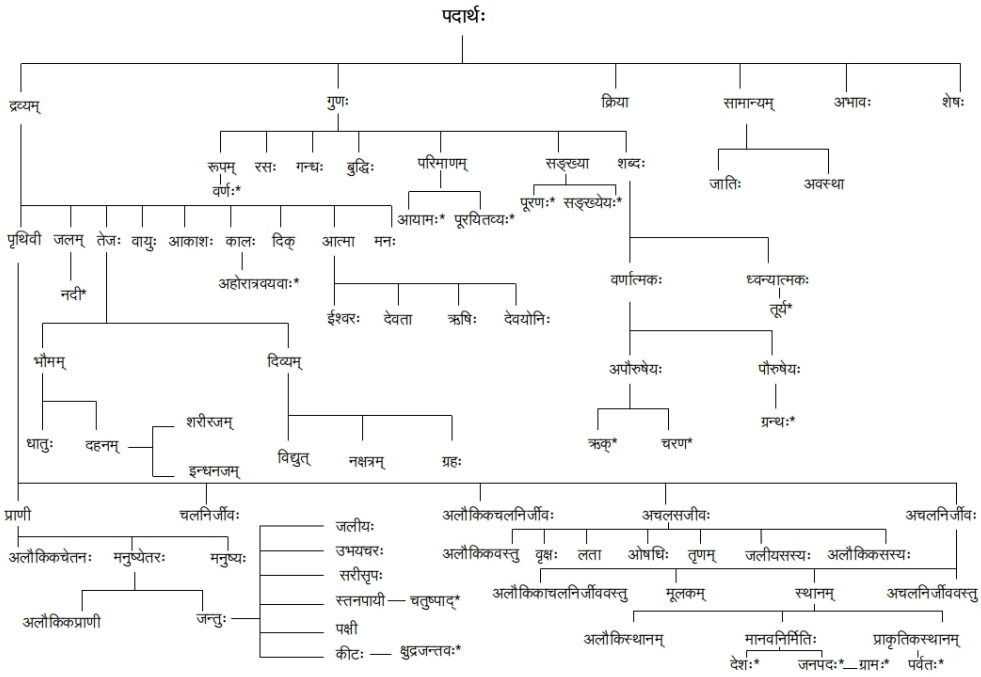

Figure 5.1: Ontological classification

5.2 List of Upādhis needed for Compound Generation

vidyāvācī

ṛto vidyāyonisambandhebhyaḥ 6.3.23

yonivācī

ṛto vidyāyonisambandhebhyaḥ 6.3.23

dhānyavācī

vibhāṣā vṛkṣa-mṛga-tṛṇa-dhānya-vyañjana-paśuśakunyaśvavadava-

pūrvāparādharottarāṇām 2.4.12

vyañjanavācī

annena vyañjanam 2.1.33

vibhāṣā vṛkṣa-mṛga-tṛṇa-dhānya-vyañjana-paśuśakunyaśvavadava-pūrvāparādharottarāṇām 2.4.12

bhakṣyavācī

bhakṣyeṇa miśrīkaraṇam 2.1.34

vayas

vayasi dantasya datṛ 5.4.141

ṛṇa

kṛtyairṛṇe 2.1.43

niravasitāḥ

śūdrānāmaniravasitānām 2.4.10

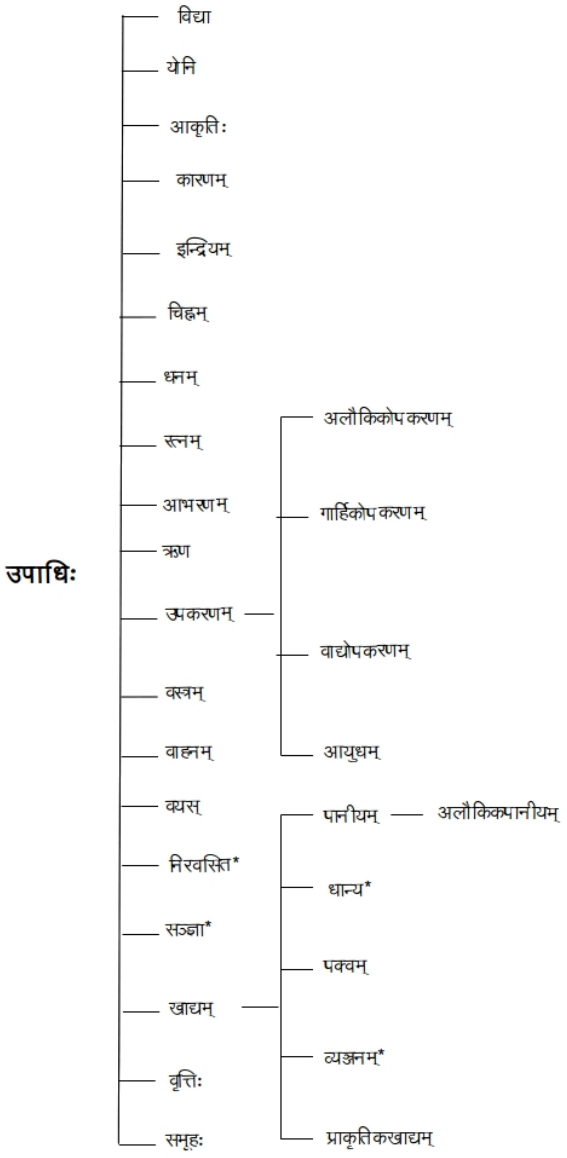

Figure 5.2: Other properties

5.3 Pragmatics

There are several cases we encountered in the derivation process which required context dependent information, which can't be available through a lexicon. For example if the resulting compound denotes a name saṃñā or not. This is purely context dependent and we need to get this information from the user who desires to make a compound. Various categorial information that is context dependent needed for compound generation is listed below, along with the list of relevant sūtras which need this information.

saṃjñāvācī

haladantātsaptamyāḥ saṃjñayām 6.3.9
manasaḥ saṃñāyām 6.3.4
vaiyākaraṇākhyāyāṃ caturthyāḥ 6.3.7
parasya ca 6.3.8
saṃjñāyām 2.1.44
anyapadārthe ca saṃjñāyām 2.1.20
añnāsikāyāḥ saṃjñāyāṃ nasaṃ cāsthūlāt 5.4.118
vā saṃjñāyām 5.4.133
na saṃjñāyām 5.4.155
agrākhyāyāmurasaḥ 5.4.93
striyāṃ saṃjñāyām 5.4.143
pūrvapadātsaṃjñāyāmagaḥ 8.4.3
saṃjñāyāṃ kanthośīnareṣu 2.4.20
udakasyodaḥ saṃjñāyām 6.3.57
ṅyāpoḥ saṃjñāchandasorbahulam 6.3.63

sahasya saḥ saṃjñāyām 6.3.78

aṣṭanaḥ saṃjñāyām 6.3.125

nare saṃjñāyām 6.3.129

kāravācī

kāranāmni ca prācāṃ halādau 6.3.10

praśaṃsāvacana

praśaṃsāvacanaiśca 2.1.66

adhvaryukratuḥ

adhvaryukraturanapuṃsakam 2.4.4

sādṛśye

yathā'sādṛśye 2.1.7

kṣepe

khaṭvā kṣepe 2.1.25

kṣepe 2.1.47

kiṃ kṣepe 2.1.64

kimaḥ kṣepe 5.4.70

atyantasaṃyoge

atyantasaṃyoge ca 2.1.28

adhikārthavacana

kṛtyairadhikārthavacane 2.1.32

upamānavācakāh

upamānāni sāmānyavacanaiḥ 2.1.55

upamānācca 5.4.137

pūjā

ktena ca pūjāyām 2.2.12

na pūjanāt 5.4.69

pūjyamāna

sanmahatparamottamotkṛṣṭāḥ pūjyamānaiḥ 2.1.61

vṛndarakanāgakuñjaraiḥ pūjyamānam 2.1.62

adhikaraṇavācī

adhikaraṇavācinā ca 2.2.13

vipratiṣiddhaṃ cānadhikaraṇavāci 2.4.13

adhikaraṇaitāvattve ca 2.4.15

ekadeśi

pūrva-parādharottaram-ekadeśinaikādhikaraṇe 2.2.1

karmavyatihārah

ickarmavyatihāre 5.4.127

adarśanam

akṣṇo'darśanāt 5.4.76

vandanam

vandite bhrātuḥ 5.4.157

lubdhayogaḥ

dakṣiṇermā lubdhayoge 5.4.126

viśeṣaṇavācī

saptamīviśeṣaṇe bahuvrīhau 2.2.35

anumeyaḥ

dvitīye cānupākhye 6.3.80

viprakṛṣṭākhyā

adhyayayanato'viprakṛṣṭākhyāyām 2.4.5

avasthā

kakudasyāvasthāyāṃ lopaḥ 5.4.146

mitra

suhṛd durhṛdau mitrāmitrayoḥ 5.4.150

amitra

suhṛd durhṛdau mitrāmitrayoḥ 5.4.150

ākrośe

ṣaṣṭyāḥ ākrośe 6.3.21

Chapter 6

Implementation Details

In the second chapter we saw the steps involved in the compound word /stem generation. The process starts with an alaukikavigraha (analytical paraphrase) and undergoes a series of transformations, resulting into a compound. Each transformation corresponds to a well defined operation in the process. Now we describe each module. We give below the output of each module the range of sūtras that correspond to this operation, the linguistic resources needed for their implementation. The output of the previous module is the input for the successive module.

6.1 Description of modules and their implementation

A) **Module**: Designating a saṃjñā **alaukikavigraha**

Input: A string in either of the following forms

stem '+' sup stem '+' sup

avyaya stem ‘+’ sup

stem ‘+’ sup avyaya

Output: Designation of the saṃjñā alaukikavigraha.

At this stage the input string is tokenized into stem1, stem2, sup1, sup2, avyaya etc as the case may be. Every stem acquires the semantic features such as gender, meaning, gaṇa, jāti and upādhi.

Resources: A structured lexicon with the following information for each stem/prātipadikam: liṅgam, meaning, gaṇa, jāti, upādhi, vacanam

Pāṇinian sūtra: *saha supa* (A2.1.4)

This step may well be implemented as a context free grammar rewrite rule in say, Yacc, with a lexer to tokenize the input. The lexer returns a feature structure from the lexicon along with each token identified. The grammar is written as

alaukikavigrahaḥ : stem ‘+’ sup stem ‘+’ sup
|avyaya stem ‘+’ sup
|stem ‘+’ sup avyaya
;

B) **Module**: Designating a saṃjñā **samāsa** for the analytical paraphrase

Output: Designation of the samāsasaṃjñā for the alaukikavigraha

Pāṇinian sūtra: *prākkadārātsamāsaḥ* (A2.1.3)

This also can be implemented simply by a context free rewriting rule that assigns a new designation as

samāsa : alaukikavigrahaḥ
;

C) **Module**: Introduction of the **samāsanta suffixes**

The introduction of samāsānta suffix is under specific conditions. As we saw in the previous chapter, these conditions are either simple conditions on the input string comprising of alaukikavigraha or may need some morphological, semantic or pragmatic information. Further, among all the sūtras at the most one sūtra is applicable. So we decided to implement this module as a lexer which has a powerful regular expression matching which can take care of the conditions on the input string. Other conditions on morphology, semantics and pragmatics are implemented as an Ansi C code. At this stage a samāsānta suffix is added to the alaukika vigraha.

Output: alaukikavigraha samāsāntasuffix.

Pāṇinian sūtra: The sūtras from A5.4.68 to A5.4.160 are related to this samāsanta module.

D) **Module**: Designating **prātipadikasaṃjñā** to the compound

Output: Designation of the prātipadikasaṃjñā for the compound.

Pāṇinian sūtra: *kṛttaddhitasamāsāśca* (A1.2.46)

This again is just assignment of a prātipadikasaṃjñā which can be implemented naturally as a rewrite rule as

```
prātipadikam : samāsa
                ;
```

E) **Module**: Designating **upasarjana saṃjñā** to decide the word order

Output: index of the stem getting an upsarjana designation.

Pāṇinian sūtras: *prathamānirdiṣṭam samāsa upasarjanam* (A1.2.43), and all samāsavidhāyaka sūtras.

In this Module, the order of the components in the compounded form is decided. The relevant sūtras are 'prathamānirdiṣṭam samāsa upasarjanam' (A1.2.43) and *'upasarjanam pūrvam'* (A2.2.30). By the first sūtra first the word that takes the position is assigned an upasarjanasaṃjñā and later it is moved to the first place. The upasarjanasaṃjñā is assigned to the word referred to by the nominative case ending term in the sūtra. In case of Bahuvrīhi and Dvandva, there is a problem in assigning upasarjanasaṃjñā. The governing sūtra in Bahuvrīhi is *'anekamanyapadarthe'* A2.2.24. Here the word in nominative case is 'anekam' which refers to all the words together (undergoing the compound formation). And thus, all the words (which are in nominative case) in Bahuvrīhi gets upasarjanasaṃjñā. In the case of Dvandva, the governing rule is *'cārthe dvandvaḥ'* A2.2.29. In this sūtra there is no such word which ends in nominative case as in other governing sūtras. Hence, there is no scope of upasarjanasaṃjñā in this compound type. In such cases Pāṇini gives rules to decide the first component (pūrvanipata).

In case of tatpuruṣa compounds one sūtra deserves special treatment. The sūtra is *'kugatiprādayaḥ*' A2.2.18. Here in this sūtra Pāṇini makes use of a special technical term 'gati'. This sūtra says in an alaukikavigraha, the component with gatisaṃjñā gets an upasarjana saṃjñā. Hence the algorithm for deciding the position of the component assuming first place is as follows.

1. If the samāsaprakāra is Tatpuruṣa and the samāsavidhāyaka sūtra is *'kugatiprādayaḥ'* A2.2.18 then get the position of the component with gati saṃjñā assign upasarjanasaṃjñā to the component with gati saṃjñā
2. In all other cases of tatpuruṣa, and in the case of Avyayībhāva

get the position of a component qualifying the upasarjanasaṃjñā.

3. In case of Bahuvrīhi
   (1) If there is a term in 7^{th} case, it gets the 1^{st} position.
   (2) If there is viśeṣaṇa-viśeṣyabhāva, the component which is viśeṣaṇa gets the 1^{st} position.
4. In case of Dvandva
   (1) If a word, which has the designation 'ghi' gets the 1^{st} position.
   (2) If a word, which starts with a vowel and ends in 'a' then it gets the 1^{st} position.
   (3) If a word which contains less number of vowels will get the 1^{st} position.
   (4) The compounds which contain words from specific gaṇa like rājadantādi are treated as exceptions.

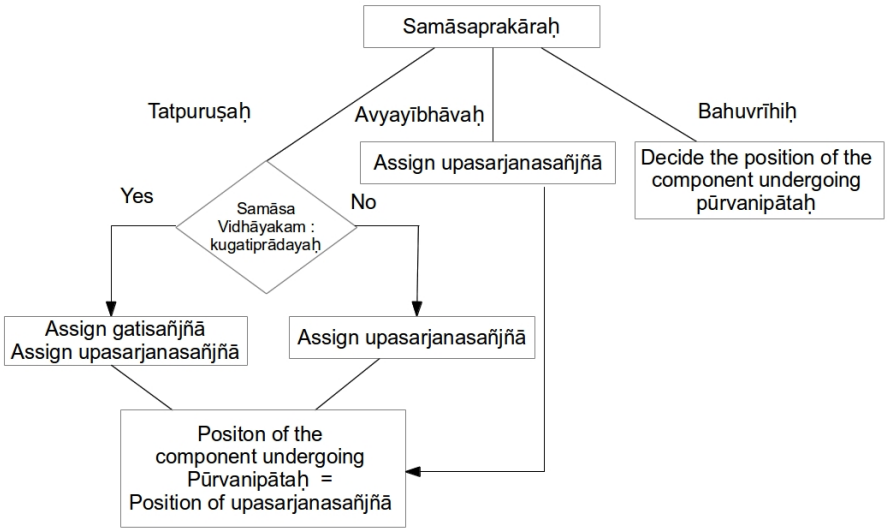

Figure 6.1: Deciding the position of the component

F) **Module**:**Pūrvanipāta**

Output: The alaukikavigraha in desired word order.

If an upasarjanasaṃjñā has been assigned, it gets the 1^{st} position by *'upasarjanam pūrvam'* (A2.2.30). In the absence of upasarjanasaṃjñā the word marked as pūrvanipāta assumes the 1^{st} position.

G) **Module**: **Aluk** (Non-elision of the Vibhakti)

Output: The suffix of the 1^{st} component is retained if certain conditions are satisfied.

Pāṇinian sūtra: The sūtras from A6.3.1 to A6.3.25 are related to this Aluk Module.

H) **Module**: **Subluk** (Elision of the vibhaktis)

Output: If either no sūtra related to aluk is applicable, or if aluk related sūtra is optionally applicable, then the deletion of the case suffixes takes place.

Pāṇinian sūtra: *supo dhātuprātipadikayoḥ* (A2.4.71)

This is achieved just by assigning a null value to the suffix.

I) **Module**: Transformation of the first component

In certain cases the first component undergoes certain transformations in the presence of the second component.

Pāṇinian sūtras: This module deals with the sūtras starting from the 1^{st} and 3^{rd} quarter of the 6^{th} chapter

These are implemented using a lexer with pattern matching for the condition on the 2^{nd} component and Ansi C functions for morphological, semantic or pragmatic conditions on the 2^{nd} component.

J) **Module**: Transformation of the second component

In some cases the second constituent undergoes certain transformations in the presence of the first component

Pāṇinian sūtras: This module deals with some sūtras in the 8^{th}chapter's 4^{th} pada

of the Aṣṭādhyāyī. These sūtras belong to the tripādī section of Aṣṭādhyāyī. The rules in this section behave in a different way than those in the rest part of the Aṣṭādhyāyī. The main guiding sūtra for this section is *'pūrvatrāsiddham'* (A8.2.1). This says that a sūtra in this section is invisible to all the sūtras preceding it. In other words, these sūtras are to be operated sequentially. Now a lexer allows application of only one sūtra. If

any other sūtra is applied on the output, we need to call the lexer again and this time we also have to remember the previous sūtra number. In the case of compound generation we observed that on a given string only one sūtra will be applicable and hence we could again implement these sūtras using lexer.

K) **Module**: Padakārya
In this module a special sūtra that tranforms a pada is implemented using lex.
Pāṇinian sūtra: *nalopaḥ prātipadikāntasya* (A8.2.7)

L) **Module**: Sandhikārya
Here the two constituents are now joined following the sandhi operation.
We used the existing sandhi module that was developed by the University of Hyderabad[1] for this compound generation.

6.2 Scope of Generator

We have implemented the modules upto sandhi. After this there are other operations such as deciding the gender (Liṅganirdhāraṇakārya), deciding the number (Saṅkhyā), deciding the accent (Svara) which are not implemented. To implement the liṅganirdhāraṇakārya module we require a morphological analyser which can analyse the words with feminine ending suffixes and identify the feminine suffix. Because, the current implementation of the compound generator takes the alaukikavigrahavākya as the input where the derivation of the prātipadika is not mentioned. In some cases such as 'sabhā', when it forms a compound with 'yama', sabhā

[1] http://sanskrit.uohyd.ac.in/scl

looses its feminine suffix. So in order to delete this feminine suffix, we need to know the prātipadika prior to the addition of feminine suffix. So treating gender as a semantic condition at input is not enough. Since such an analyser is not in place, we decided to limit ourselves to the formation of the prātipadikam.

6.3 Some specific issues

1. Order of implementation of rules

   While some of the modules were written as rewrite rules, most of the others are implemented using a lexer. Each module implemented with a lexer is thus a finite state automaton. When we decided to use the lexer such as flex (fast lexical analyser for GNU linux), we had to take into account the conflict resolution strategy used in flex. The conflict resolution in flex works as below.

   i) In case of more than one applicable rules, the one with largest matching string is tried.

   ii) In case of two rules with same length of matching expression, the earlier rule is tried.

   Pāṇini typically states the general rule (utsarga) first and then he states the exception (apavāda). In case the exceptional rule involves some extra conditions then we can follow the given sequence of the rules. But in cases where the exception is stated for specific words, then we cannot write the exception rule after the general rule, in lex. Because in that case, since the length of the matching string is

same, the earlier rule, i.e, the general rule will be tried and not the exceptional one. In such cases we have inverted the order of rules. In all other cases the order of the order in Aṣṭādhyāyī is followed. Here is an example:

The sūtra *'ateḥ śunaḥ* 5.4.96' says that if the first word is ati and second word is śvan then an additional suffix 'tac' is added to the entire string. The second sūtra *'na pūjanāt* 5.4.69' says that if the first word is either su or ati and if it is in the sense of pūjā and is followed by any word then the suffix 'tac' is not added. If we write these rules in Lex as per Aṣṭādhyāyī then 5.4.69 takes place first and it does not allow the other rule 5.4.96 for the 'tac' operation by 5.4.96.

6.4 Evaluation

There is no such book apart from Pāṇinīyavyākaraṇodāharaṇakośa which deals with the entire process of compounds. So this is the only authenticated and manually tested data. Hence this is a gold data for our purpose. We have tested around 500 examples from this book and compared the output generated by our Sanskrit Compound Generator with it. The final results match. However, there are differences in the prakriyā part. The reasons for the differences are already discussed in the 3^{rd} chapter. So we did not compare the prakriyā. We limited ourselves to 500 examples. Because, for most of the examples, a morphological analyser dealing with the feminine suffixes, or secondary derivative suffixes (taddhitas) was needed. In the absence of such an analyser, we could cross check our generator with only a limited cases.

Chapter 7

Conclusion

The present work is an attempt on building an Automatic Sanskrit Compound Generator. While implementing these rules we observed that most of the rules fall under the category of regular grammar. In our implementation we observed that most of the rules require a sequence of phonemes. We implemented those rules in Lex (Lexical Analyser). For other rules additional linguistic conditions such as morphological analysis, lexical meaning etc were needed and in some cases extra-linguistic conditions were required. Such rules are implemented using a C code to handle the additional linguistic conditions and an user interface to get the extra linguistic information from the user. There are around 90 rules which require additional information apart from the given input string. For the remaining rules pattern matching is enough to generate the compound.

Limitations of Sanskrit Compound Generator

The current implementation has following limitations.

- The input for the generator should be an analytical paraphrase.
- This system handles only binary compounds. Thus the generation of Dvandva with more than two components and bahupada Bahuvrīhi are not being handled in the current implementation.
- Morphological analyser is required to deal with feminine suffixes, and secondary derivative suffixes (taddhitas). Hence such cases where this information is needed are also out of scope of these current implementation.
- In case of upapadatatpuruṣa, we still do not have a morphological analyser that generates the kṛdanta forms required in the upapadatatpuruṣa compounds and hence our compound generation in such cases start with the form. For example we start with

  kumbha + ṅas kāra

  and not

  kumbha + ṅas kṛ + aṇ
- We also do not deal with the accents.
- This generator produces the final output in the form of prātipadika and not a finished subanta. Since to generate a subanta, we need the feminine forms in certain cases and we still do not have a feminine form generator.

Future directions

- The above mentioned limitations need to be overcome.
- The current lexical analyser uses pattern matching only on a string. However, a lexical analyser with pattern matching on records having different fields such as the phonemic sequence, meaning, morphological analysis, genre, gender etc will be more appropriate to implement the rules in a natural way. This implementation has also given us confidence that we can, on similar lines, develop a generator for other derived forms such as kṛt and taddhita.

Abbreviations

A: Pāṇini's *Aṣṭādhyāyī*. See Pande.

MBh: Patañjali's *Mahābhāṣya*. See Bhargava Shastri.

PBs: Paribhāṣenduśekharaḥ. See Ganesha Shastri.

PMa: Prauḍhamanoramā. See Sadasivasarma Shastri.

VBhs: Bhūṣaṇasāraḥ. See Gangadhar Bapurao kale.

Bibliography

Bhaṭṭojidīkṣitaḥ. *Vaiyākaraṇasiddhantakaumudī*. Motilal Banarsidass, New Delhi, 2001.

V. V. Bhandare. Structural and Semantic Aspects of the Dvandva Compound. *Annals of the Bhandarkar Oriental Research Institute*, 76(1-4): 89–96, 1995.

Akshar Bharati and Amba Kulkarni. Information Coding in a Language: Some Insights from Pāṇinian Grammar. *Dhimahi, Journal of Chinmaya International Foundation Shodha Sansthan*, 80(1):77–91, 2010.

Saroja Bhate and Subhash Kak. Pāṇini's Grammar and Computer Science. *Annals of the Bhandarkar Oriental Research Institute*, 1991-1992(1/4): 79–94, 1992.

George Cardona. Studies in Indian Grammarians I: The Method of Description Reflected in the śivasūtras. In *Transactions of the American Philosophical Society*, pages 3–48. JSTOR, 1969.

George Cardona. Pāṇini: His Work and its Traditions vol 1. In *Background and Introduction. 2nd ed.* Motilal Banarsidass, 1997.

George Cardona. On the Structure of Pāṇini's System. In *Sanskrit Computational Linguistics, First and Second International Symposia, Rocquencourt, France*, pages 1–32. Springer, 2009.

Ashwini Deo. Derivational Morphology in Inheritance-based Lexica: Insights from Pāṇini. In *Lingua*, volume 117.1, pages 175–201. Elsevier, 2007.

Brendan S. Gillon. Exocentric Compounds in Classical Sanskrit. In Amba Kulkarni and Gérard Huet, editors, *Sanskrit Computational Linguistics*, 2007.

Brendan S. Gillon. Tagging Classical Sanskrit Compounds. In Amba Kulkarni and Gérard Huet, editors, *Sanskrit Computational Linguistics 3*, pages 98–105. Springer-Verlag LNAI 5406, 2009.

Pawan Goyal and Gérard Huet. Completeness Analysis of a Sanskrit Reader. In *Proceedings, 5th International Symposium on Sanskrit Computational Linguistics*. D. K. Printworld(P) Ltd, 2013.

Pawan Goyal, Amba Kulkarni, and Laxmidhar Behera. Computer Simulation of Aṣṭādhyāyī: Some Insights. In *Sanskrit Computational Linguistics, First and Second International Symposia, Rocquencourt, France*, pages 139–161. Springer, 2009.

F. Grimal, V. Venkataraja Sarma, and S. Lakshminarasimham. *Pāṇinīyavyākaraṇodāharaṇakośaḥ*. Prentice-Hall of India, New Delhi, 1995.

Rodney Huddleston and Geofrey K. Pullum. *The Cambridge Grammar of the English Language*. Cambridge University Press, 2002.

Gérard Huet. Design of a Lexical Database for Sanskrit. In *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries*, pages 8–14. Association for Computational Linguistics, 2004.

Malcolm D. Hyman. From Pāṇinian Sandhi to Finite State Calculus. In *Sanskrit Computational Linguistics, First and Second International Symposia, Rocquencourt, France*, pages 253–265. Springer, 2009.

Girish Nath Jha and Sudhir K. Mishra. Semantic Processing in Pāṇinis Kāraka System. In *Sanskrit Computational Linguistics, First and Second International Symposia, Rocquencourt, France*, pages 239–252. Springer, 2009.

S. D. Joshi. *Patañjali's Vyākaraṇa - Mahābhāṣya Samarthāhnika (P. 2.1.1)*. Centre of Advanced Study in Sanskrit, University of Poona, Poona, 1968.

S. D. Joshi. Background of Aṣṭādhyāyī. pages 1–5, 2009.

S. D. joshi and J. A. F. Roodbergen. On Siddha, Asiddha and Sthānivat. In *Annals of the Bhandarkar Oriental Research Institute*, volume 68, pages 541–549. Bhandarkar Oriental Research Institute, 1987.

S. D. Joshi and J. A. F. Roodbergen. On Kāraka, Vibhakti and Samāsa. *Annals of the Bhandarkar Oriental Research Institute*, 80(1-4):95–112, 1999.

J.P Gangadhar Bapurao kale. *Vaiyākaraṇabhūṣaṇasāraḥ by kaundabhaṭṭa with the commentary of śāṅkarī by śankaraśāstrī*. Anandashram, Pune, 1935.

Kauṇdabhaṭṭaḥ. *Vaiyākaraṇabhūṣaṇasāraḥ*. Chaukamba Sanskrit Pratisthan, 2005.

Kiparsky. Pāṇini is slick, but he isn't mean. *Nagoya Studies in Indian Culture and Buddhism, Sambhāṣā*, 26(1-28), 2007.

András Kornai. *Mathematical Linguistics*. Springer-Verlag London Ltd, 2008.

Amba Kulkarni and Anil Kumar. Statistical Constituency Parser for Sanskrit Compounds. In *Proceedings of ICON 2011*. Macmillan Advanced Research Series, Macmillan Publishers India Ltd., 2011.

Malhar Kulkarni. Phonological Overgeneration in Pāṇinian System. In *Sanskrit Computational Linguistics, First and Second International Symposia, Rocquencourt, France*, pages 306–319. Springer, 2009.

Anil Kumar. *An Automatic Sanskrit Compound Processor*. University of Hyderabad, 2012.

Anil Kumar, Vipul Mittal, and Amba Kulkarni. Sanskrit Compound Processor. In G N Jha, editor, *Proceedings of the 4th International Sanskrit Computational Linguistics Symposium*. Springer-Verlag LNAI 6465, 2010.

Mahavir. Treatment of Samāsa in Pāṇini. *Annals of the Bhandarkar Oriental Research Institute*, 67(1-4):147–158, 1986.

Anand Mishra. Simulating the Pāṇinian System of Sanskrit Grammar. In *Sanskrit Computational Linguistics, First and Second International Symposia, Rocquencourt, France*, pages 127–138. Springer, 2009.

Anand Mishra. Modelling Aṣṭādhyāyī: An Approach Based on the Methodology of Ancillary Disciplines (vedanga). In *Sanskrit Computational Lin-*

guistics, Fourth International Symposium, Delhi, India, pages 239–258. Springer, 2010.

M. Srimannarayana Murti. *Sanskrit Compounds. A Philosophical Study*. The Chowkamba Sanskrit series Office, Varanasi, India, 1974.

Sivaja S. Nair. *The Knowledge Structure in Amarakośa*. University of Hyderabad, 2011.

Sivaja S. Nair, Shrinivasa Varakhedi, and V. Sheeba. Extended Nyāya-Vaiśeṣika Ontology as Applied to Amarakośa Knowledgenet. In Malhar Kulkarni and Chaitali Dangarikar, editors, *Recent Researches in Sanskrit Computational Linguistics Fifth International Symposium Proceedings*, pages 223–242. D. K Printworld (P) Ltd, New Delhi, 2013.

Gopal Dutt Pande. *Aṣṭādhyāyī of Pāṇini, elaborated by M. M. Panditaraj Dr. Gopal Shastri*. Chaukhamba Surbharati Prakashan, Varanasi, 2004.

Bhagyalata Pataskar. Some Observations about the Compound Structure of Aṣṭādhyāyī. *Annals of the Bhandarkar Oriental Research Institute*, 77 (1-4):121–131, 1996.

Gerald Penn and Paul Kiparsky. On Pāṇini and the Generative Capacity of Contextualized Replacement Systems. In *Proceedings of COLING 2012: Posters*, pages 943–950, 2012.

Wiebke Petersen. A Mathematical Analysis of Pāṇini‘s śivasūtras. 13: 471489, 2004.

Rama. Inheritance-based Lexica: Insights from Pāṇini. In *Lingua 117.1*, 2007.

K. V. Ramakrishnamacharya. *Bhūṣaṇasāratattvaprakāśikā*. Rashtriya Sanskrit Vidyapeetam, Tirupati, 2010.

Pavankumar Satuluri and Amba Kulkarni. Extra Linguistic Information needed for Automatic Generation of Sanskrit Compounds: A Study. In *The recent developments in Sanskrit Computational Linguistics', at SALA-30, Hyderabad*, 2014.

Peter Scharf. Modeling Pāṇinian Grammar. In *Sanskrit Computational Linguistics, First and Second International Symposia, Rocquencourt, France*, pages 95–126. Springer, 2009.

Peter Scharf and Malcolm Hyman. *Linguistic Issues in Encoding Sanskrit*. Motilal Banarsidass, Delhi, 2009.

Peter Scharf, Pawan Goyal, Anuja Ajotikar, and Tanuja Ajotikar. Voice, Preverb, and Transitivity Restrictions in Sanskrit Verb use. In *Sanskrit Syntax, Selected papers presented at the seminar on sanskrit syntax and discourse structures (forthcoming)*, 2015.

Peter M Scharf. Rule-Blocking and Forward-Looking Conditions in the Computational Modelling of Pāṇinian Derivation. In *Sanskrit Computational Linguistics, Fourth International Symposium. New Delhi*, 2010a.

Peter M Scharf. Rule Selection in the Aṣṭādhyāyī, or Is Pāṇinis Grammar Mechanistic? In *Studies in Sanskrit Grammars: Proceedings of the Vyakarana Section of the 14th World Sanskrit Conference*, 2010b.

Bhargava Shastri. *The Vyākaraṇamahābhāṣya Of Patañjali with the Commentary of Bhāṣyapradīpa of Kaiyatopādhyāya and The Super Com-*

mentary Bhāṣyapradīpoddyota of Nageśa Bhaṭṭa, volume I Navāhnika,. Chaukhambha Sanskrit Pratishthan, Delhi, 1987a.

Bhargava Shastri. *The Vyākaraṇamahābhāṣya Of Patañjali with the Commentary of Bhāṣyapradīpa of Kaiyatopādhyāya and The Super Commentary Bhāṣyapradīpoddyota of Nageśa Bhaṭṭa*, volume II Navāhnika,. Chaukhambha Sanskrit Pratishthan, Delhi, 1987b.

Ganesha Shastri. *Paribhāṣenduśekharaḥ with the commentary of Gada by Vaidyanātha*, volume 72. Anandashramgranthāvalī,Pune.

Sadasivasarma Shastri. *The Prauḍhamanoramā by M. M. Bhattoji Dīkṣita Vol. I avyayībhāvānta, with the commentary Laghuśabdaratna by Hari Dīkṣita and śabdaratna commentary by M. M. Bhairava Miśra and Prabha Notes by M. M. Sri Madhav Sastri Bhandari and Śabdaratna Pradīpaka Notes on avyayībhāvānta Portion by Pandit Jagannatha Shastri Pande*. Chaukhambha Sanskrit Series Office, Vidya Vilas Press, 1935.

Fritz Staal. Context-Sensitive Rules in Pāṇini. 1(1):63–72, 1965.

Fritz Staal. Encyclopedia of Languge and Linguistics. volume 6, page 2917, 1994.

Sridhar Subbanna and Shrinivasa Varakhedi. Computational Structure of the Aṣṭādhyāyī and Conflict Resolution Techniques. In *Sanskrit Computational Linguistics, Third International Symposium, Hyderabad, India*, pages 56–65. Springer, 2009.

Sridhar Subbanna and Shrinivasa Varakhedi. Asiddhatva Principle in Computational Model of Aṣṭādhyāyī. In *Sanskrit Computational Linguistics,*

Fourth International Symposium, Delhi, India, pages 231–238. Springer, 2010.