



The Use of Mutual Recursion in Matrix Based Nominal Declension Model Following Pāṇinian Method

Manju Mohan¹ & Dr Sampath Kumar L²

1. Research Scholar, CVV
2. CVV

Abstract: Amidst various computational models for Sanskrit grammar, this paper proposes a matrix-based computational model for the generation of subanta forms. The implementation attempts to adhere closely to the rule-based system of Pāṇini, where the applicability of grammatical rules is determined by contextual conditions. In the present system, the context is derived from user inputs and stored using several flags which guide the application of rules during the derivational process (*prakriyā*). A key feature of this approach is the use of a matrix structure representing the combinations of vibhakti and vacana, which reduces the need for repetitive looping of rules and improves computational efficiency. The system also incorporates a mechanism for handling 'it' designations as prescribed in the *Aṣṭādhyāyī* (1.3.2–1.3.9), ensuring that the derivation follows the traditional grammatical process. The model is implemented in Python, which enables easy scalability and potential integration with other computational linguistic frameworks. The system currently generates subanta forms for vowel-ending masculine, feminine and neuter stems. Apart from its computational relevance, the model also provides pedagogical value by illustrating the step-by-step derivational procedure described in the Pāṇinian grammatical tradition.

Keyword: Pāṇini, *Aṣṭādhyāyī*, Matrix Structure, Vibhakti, Vacana.

1.0 Introduction: Previous research efforts on Sanskrit nominal declension were focusing on sandhi formation and inflectional morphology, including generators for *subanta* and *tiṇanta* forms. Although some *subanta* generators are available, they are either incomplete or their computational processes do not strictly follow the procedure outlined by Pāṇini.

In this context, the present work develops a *subanta* generator implemented in Python, based on the procedural framework described in the *Aṣṭādhyāyī*. The system generates *subanta* forms automatically from a given nominal stem by applying the relevant grammatical rules in sequence. Such a system can be useful for students and researchers of *Vyākaraṇa śāstra*, as it demonstrates the step-by-step derivational procedure that leads to the formation of nominal inflected forms.

The forms generated by the proposed model is being validated against the forms generated by Samsadhani, the nominal declension tool developed by University of Hyderabad. We have taken three hundred words from the *Padakośa* as the test data. Only 50 have been tested so far and all of them yielded correct output.

2.0 Word formation in Sanskrit: Pāṇini designates the term *pada* through the rule: *suptiṇantam padam* (A 1.4.14). According to this rule, a *pada* is any form ending with either a *sup* suffix or a *tiṇ* suffix, making two forms *Subanta* (nominal forms) and *Tiṇanta* (verbal forms) respectively.

The rule A 1.2.45, *arthavād adhātu apratyaya prātipadikam* says that any meaningful word that is neither a verb nor a suffix is termed a *prātipadika*. A *subanta* is then formed by adding the appropriate *sup pratyaya* to a *prātipadika* corresponding to the *vibhakti* and *vacana* as given in Table 1. The *category* (noun or pronoun), *gender* (masculine, feminine, or neuter) and *ending letter* of the *prātipadika* (vowel or consonant) plays role in deciding the suffix.

In the Aṣṭādhyāyī, rule A 4.1.2 specifies the *sup* suffixes used for each case and number. In the current implementation, the *sambodhana vibhakti* is not yet handled.

Table 1 vibhakti pratyaya

	<i>Ekavacanam</i>	<i>dvivacanam</i>	<i>bahuvacanam</i>
<i>prathamā</i>	su	au	Jas
<i>Dvītyā</i>	am	ouḥ	ṣas
<i>Tritīyā</i>	ṭā	bhyām	Bhis
<i>Caturthī</i>	ṇe	bhyām	Bhyas
<i>Pañcamī</i>	ṇasim	bhyām	Bhyas
<i>ṣaṣṭhī</i>	ṇas	os	Ām
<i>Saptamī</i>	ṇi	os	Sup

The seven vibhakti category belong to one semantic category each, called *kāraka*. The mapping is given in Table 2. This semantic clarity of the words used in a sentence gives us the freedom to change the place of a word in a sentence, without affecting the meaning at all. Eg: *Rāma vanam gacchati* (Rama goes to the forest), *Hanumān Rāmam paśyati* (Hanuman meets Rama), *Rāmeṇa likhitaṃ pustakam* (The book is written by Rama), *Pustakaṃ Rāmāya dīyate* (The book is given to Rama), *Rāmāt bhayaṃ nāsti* (There is no fear from Rama), *Rāmasya mitraḥ Sītā* (Sita is Rama's friend), *Rāme kathā asti* (This is Rama's story) etc.

Table 2 vibhakti and Karaka mapping

Vibhakti	Kāraka	Meaning	Example
<i>prathamā</i>	<i>Kartā</i>	The subject	<i>Rāma</i>
<i>dvītyā</i>	<i>Karma</i>	The object	<i>Rāmam</i>
<i>Tritīyā</i>	<i>Karaṇa</i>	The means by which the act is performed	<i>Rāmeṇa</i>
<i>Caturthī</i>	<i>Sampradāna</i>	Recipient	<i>Rāmāya</i>
<i>Pañcamī</i>	<i>Apādāna</i>	Source	<i>Rāmāt</i>

<i>ṣaṣṭhī</i>	<i>Sambandh</i>	Possession	<i>Rāmasya</i>
<i>Saptamī</i>	<i>Adhikaraṇa</i>	The place or domain	<i>Rāme</i>

3.0 Literature review: It has been quite inspiring to see that the research on Paninian grammar started more than a century which attributed to phenomenal results and substantial leap in this field. Table 3 gives a snapshot of a few of these works which further explains the need for a more adaptive computational framework for Paninian grammar system.

Table 3: A few remarkable studies on Paninian grammar

Author	Title	Content
Cardona, 1970	“Some principles of Pāṇini’s grammar.”	General and limited blocking of uthsarga apavada rules are discussed. Antaranga and bahiranga rules stating the linear and non linear proximity are also mentioned. Criticizes Staal (1965) for stating that 6.1.88 takes precedence over 6.1.87 by just 1.4.2 (vipratishedhe param kariam). Cardona states that instead, 6.1.88 is an apavada of 6.1.87.
Cardona, 1974	Pāṇini’s kāraṇa rules: agency, animation, identity	vivakṣā (intention of the user) plays an equally important rule in Paninian grammar, along with the syntactical rules of derivation. Emphasizes the statement by Kiparsky and Staal (1969) that the Paninian karaka system is not purely ontologically classified, but depends upon the vivakṣā. Eg: bow can serve as either point of departure (sampradana) or as the instrument (karana), but not as both in a single action. Śabda-bodha (verbal cognition has an important role in identifying the correct ontological classification.
J F Staal, 1965	Context-Sensitive Rules in Pāṇini	The context sensitive rules in Ashtadhyayi like ikoyanaci are discussed to show how Panini anticipated the formal language theory presented by Chomsky, centuries down the line.
Kiparsky and Staal 1969	Syntactic and Semantic relations in Panini	A grammatical derivation passes through four different levels: Meaning, Karaka rules, Morphological categories, Phonological forms
Joshi and Kiparsky 1979	Siddha and asiddha in Pāṇinian phonology.	In a derivation, rules can be applied cyclically but not in the same context.
Joshi and Kiparsky, 2006	The extended siddha-principle.	Discusses the feed and bleed principles of rules, and how to maximize the siddha relation in derivations. Rule A is asiddha wrt rule B if: a)Rule A’s output is invisible to Rule B b)Rule A’s input is visible to Rule B
Penn and Kiparsky	On Pāṇini and the Generative Capacity	If there is a presupposed constraint set in reusing the contexts, Paninian grammar can behave as a Context Sensitive grammar

2012	of Contextualized Replacement Systems.	which is not ambiguous.
S D Joshi, 1977	The ordering of the rules in Pāṇini's grammar	The ordering of the rules is not explicitly stated in the body of the Ashtadhyayi, yet it is assumed by Panini, with the concepts of uthsarga and apavada and facilitating reverse ordering. Panini designed some special devices which eventually give us some ideas of how to order the rules for the correct derivational output.
S. D. Joshi and J. A. F. Roodbergen, 1991	The Siddha-Principle as Presupposed by Some Pāṇinisūtras	Sanskrit grammar has five ways for conflict resolution: They are paravipratishedha, pūrvavipratishedha , nityatva , antaraṅgatva , and utsargāpavādatva. Both one way and two way conflicts are discussed here
Bronkhorst, Johannes. 1980.	Asiddha in the Astadhyayi : a misunderstanding	Pāṇini's grammar uses <i>asiddha</i> and <i>asiddhavat</i> with different technical meanings, leading to confusion among commentators.
Peter Scharf, 1992	The Sanskrit Library: A Proposal for the Development of a Digital Sanskrit Library	Digitizing Sanskrit manuscripts and emphasizing a standardised markup which is a precursor for the later XML. This was an early step in modelling Aṣṭādhyāyī, and paved way for the further research by Malcolm Hyman and Amba Kulkarni.
Scharf and Hyman, 2002	Linguistic Issues in Encoding Sanskrit	Used XML for text encoding in the Sanskrit Library Project.
Peter Scharf, 2007	Modeling Pāṇinian grammar	Study on the history of modelling Sanskrit grammar
Malcolm D Hyman, 2007	From Pāṇinian Sandhi to finite State Calculus	Sandhi rules in Astadhyayi are represented in machine readable XML format, having right and left context information incorporated in to it. A Finite State Transducer is introduced for Pāṇinian sandhi.
Barend A Van Nooten, 1967	Pāṇini's Replacement Technique and the Active Finite Verb	Explains how Paninian grammar is a rule based generative grammar conformable to the modern day linguistics. Shows how ādeśa is following the adhikāra sutras and states that Sanskrit grammar is non linear but context based.
Barend A Van Nooten, 1969	Pāṇini's Theory of Verbal Meaning	Relevance of meaning in grammar as a tool/device to aid in the realization of meaningful elements composing a sentence. A monumental work like Ashtadhyayi is not a mere handbook of

		morphology.
Barend A Van Nooten, 1970	The Vocalic Declensions in Pāṇini's Grammar	The nominal declension in Ashtadhyayi and its generative mechanism is discussed here. Those sections of the grammar are discussed which contribute to the formation of declensional paradigm. The meaning imparted to a nominal is conditional on the user of the grammar. A hypothetical grammar is presented wherein the rules are ordered random way and the only principle of rule ordering is the context created by the stem constituents.
Madhav Deshpande, 1990	The Process of Semantic Interpretation in Pāṇini	Directionality of Paninian grammar as a device for producing expressions, not for decoding a given expression.
Thieme, 1983	Meaning and Form of the 'Grammar' of Pāṇini	Panini's teaching methods approaches the accuracy of a mathematical deduction
Houben, 1999	Meaning statements' in Pāṇini's grammar: on the purpose and context of the Astādhyāyī"	Semantic elements are vague to be considered as the starting point of the grammar to generate words/utterances. So, the semantic level should include both pragmatics and intentions. Paninian grammar presupposes certain cognitive capacities and abilities to perform the derivation.
1974 MD Pandit	Formal and Non formal in Panini	Pāṇinian grammar is not fully formal like a mechanical device or fully non formal like a dictionary; but integration of both formal and non formal elements, combining both form and meaning, thereby making it both precise and comprehensive.
Madhav Deshpande, 1991	Prototypes in Paninian syntax	Pāṇini's kāraṇa terminology is not purely formal but often influenced by prototype-based reasoning. Thus the grammar bridges a gap between pre-formal, culturally evident categories and formalized grammatical categories. This paper explains how vivakṣā (speaker's intention) and cultural logic play a role in assigning kāraṇa designations.
Madhav Deshpande, 1991	Panini in the context of modernity	Rule ordering does not automatically resolve ambiguities; the speaker's intention helps in assigning kāraṇa designations. Compares Pāṇini's system with Chomsky's generative grammar principles, pointing out the similarities in rule-based derivation.
Amba Kulkarni, 2009	Computer Simulation of Ashtadhyayi	Checks Databases like Dhatupatha for assigning designations. Separate modules for it designation, transformations, assidhavat, tripadi and conflict resolutions. Uses arrays for pratyayavidhi rule and angavidhi rule. Yatap infinite loop issue occurs.

		eg: ramaa+yaayaayai
Samir Sohoni & Malhar Kulkarni, Amba Kulkarni, 2018	A Functional Core for the Computational Aṣṭādhyāyī	Rule based computational framework for Aṣṭādhyāyī
Amba Kulkarni, 2018	PAIAS: Pāṇini Aṣṭādhyāyī Interpreter As a Service	Web based interpreter for Paninian grammar. Shows step by step derivation of Sanskrit forms
Madhav M Deshpande 1985	Ellipsis and syntactic overlapping: current issues in Pāṇinian syntactic theory.	He discusses how ellipsis (useful omission to make conversation concise) and syntactic overlapping (conflict of rules) are treated in Paninian grammar, Sanskrit being shifted from a spoken native language to a second language using formal technical devices to address derivational conflicts.
Frits Staal, 1965	Euclid and Pāṇini	Comparison between the methods employed by Greek mathematician Euclid and Indian grammarian Pāṇini. Emphasizes that the grammar is descriptive and not prescriptive. (As Patanjali had mentioned one can go to potter to buy pots but can't go to grammarian to get words.) Staal says how laghava is achieved by various sandhis in Paninian grammar. Panini emphasized on shorter length, whereas Euclid stood for smaller numbers; both advocating for simplicity. The algebraic style of the sutras can assume forms which are hardly intelligible without a commentary.
Barend Faddegon, 1936	Studies on Paninis grammar	Explains a synoptical table of the different arrangements of noun-cases and kārakas, with a western point of view towards Sanskrit.
Peter Scharf, 2007	Modeling Paninian grammar	The XML data structures used to model Pāṇinian sandhi can be augmented to allow derivation of nominal stems. Scharf and Hyman implemented Pāṇinian nominal derivation by introducing an additional parameter. The XML nominal declension procedure selects stems for sets of rules selected in advance by hand for their known application to stems that meet the ruleset's selection criteria. But in Pāṇinian procedure, the rules are auto applied if the input data meets the condition of the rule. But this is not a close model of the Pāṇinian system.
Dhaval Patel, Sivakumari Katuri, 2015	Prakriyāpradarśinī- an open source subanta generator	Imitates the derivation process in Siddhāntakaumudī (SK) of Bhaṭṭojī Dīkṣita (1910) using an open source PHP code.

4.0 Limitations of the above models: Although there were models developed for Pāṇinian procedure, we can see some limitations in these:

1. Amba Kulkarni's XML based model: Cyclic application of some rules. Eg: Yatap infinite loop issue occurs. ramaa+yaayaayai.
2. Amba Kulkarni's PAIAS: A few ambiguities observed and human intervention necessary in some conflict resolution. Also, certain rare rules have not been implemented
3. Peter Scharf and Hyman's XML model: Not a close model of Pāṇinian procedure as the rules are not auto selected if the rule conditions are met for the input data.
4. Dhaval Patel's PHP based model:
 - a) No separate handling for naḍi and ghi designations.
 - b) They use iterative loops not ideal for tree like natural Language structure
 - c) They require patch code rarely, for a few words if automatic conflict resolution is not done.
 - d) They have presented a reconciliation of Siddhāntakaumudī and Aṣṭādhyāyī.
 - e) Calling sasajusoruh twice-inside and outside of tripadi as rutva is not asiddha for successor rule, in their NLP order or rules. But they are ensuring that the rule is not applied in tripadi again. Cardona has mentioned that the physical order of rules in Aṣṭādhyāyī is not followed by Paninian prakriya as most of the times rules derive context from the rules specified somewhere later in the chapters, that is latter rules *feed* the former ones in Aṣṭādhyāyī.
 - f) Have handled the samjna sutras (definition rules) and paribhasa sutras (metarules) separately. One can not separate metarules and heading rules from the operation rules they serve to interpret (Cardona 1976). Though they are separated spatially in the grammar, it is the relation bringing these rules together while applying in a computational model.

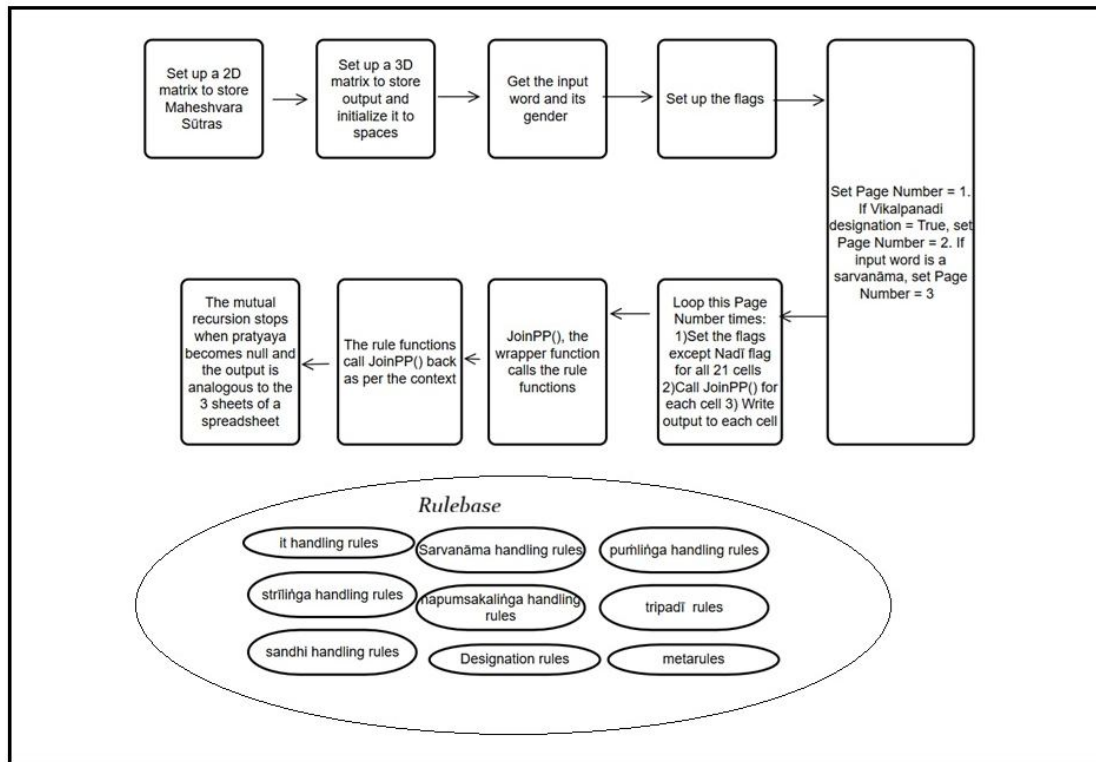
5.0 Relevance of the Proposed model: The proposed model is based on matrix based rule triggering, with the auto selection of applicable rules. Instead of iterative loops, mutual recursion is introduced as the latter give a better match with the tree like structure of a natural language. Furthermore, the recursive calls are more surgical approach to handle the samjna sutras and paribhasasutras. The cyclic application of rules is prevented using flags. We have tested the model with the nouns in Padakośa and found 98% correct output. The theoretical and logical representation of the naḍi and ghi designations is given towards the end of this article.

6.0 The methodology: Just like the grammarians celebrate the use of fewer syllables in conveying the same meaning (arthamātra-lāghavena putrotsavam manyante vaiyākaraṇāḥ), computer programmers celebrate the better UI/UX (User Interface/User Experience). Hence in the proposed model, the user input nama/sarvanama is removed, by using matrix based implementation.

As the first step, Maheshvara Sūtras are implemented with a two dimensional matrix. The function isInPratyahara () runs based on this 2D matrix and returns True or False for any letter saying the letter belongs to that pratyahara or not.

Now consider a three dimensional matrix t with three pages, each having seven rows and three columns, thus making twenty one cells in each page mapped to a particular vibhakti vacana pair. We have modeled till Saptami, not sambodhana vibhakti. Figure 1 shows the high level architecture of the proposed model.

Figure1: Rule based Subanta generation – High level architecture



This matrix based output structure is analogous to a spreadsheet with three sheets.

Figure 2: The 3 D matrix for output

Page 1 normal flow Page 2 for optional forms Page 3 for nāma forms of a sarvanāma

	ekavacana	dvivacana	bahuvacana
prathamā			
dvitīyā			
trītiyā			
caturthī			
pañcamī			
ṣaṣṭhī			
saptamī			

	ekavacana	dvivacana	bahuvacana
prathamā			
dvitīyā			
trītiyā			
caturthī			
pañcamī			
ṣaṣṭhī			
saptamī			

	ekavacana	dvivacana	bahuvacana
prathamā			
dvitīyā			
trītiyā			
caturthī			
pañcamī			
ṣaṣṭhī			
saptamī			

The cell specific attributes can be incorporated in the matrix based design. Initialize this 3 D matrix t: 3 x 7 x 3 with spaces.

As the next step, we need to set all the flags and designations to their initial values. We require flags to denote whether a particular rule is applied or not, to restrict ourselves to the Paninian way of limiting the use of a rule only once in the same context. Such flags are to be reset for each of the twenty one cells. Eg: flag101, flag102, flag7_3_116 etc

Designations are applied to decide the applicability of certain rules. Eg: ghi designation, nadī designation, vikalpa nadī designation (optional nadī designation) etc.

To fill the twenty one cells in each page, we call the function JoinPP (pd,pr) where pd is the prātipadika and pr is the pratyaya.

The wrapper function JoinPP () has all the rules being called one by one; many of them calling joinPP () back using mutual recursion, taking the derivation ahead. In this way, we are eliminating the loops. The timely setting of the flags in each rule decides the applicability of the rules in JoinPP () when being called recursively. Like any recursive function, we need a terminating condition (to come out avoiding infinite recursion) and here the condition is when pratyayā becomes null. Adhering to Pāṇinian procedure, the cyclic application of a rule in a particular context is avoided by means of scope of the intermediate variables used. The global variables retain the context we need throughout a derivation whereas the local variables serve the purpose of intermediate decision making purposes.

I have used the string manipulation functions in Python to implement the transformations throughout. A few rules cause change in the prātipadika, another set of rules cause change in the pratyaya and yet another set of rules cause change in both prātipadika and pratyaya. This recursive procedure continues until the pratyaya becomes null.

In JoinPP (), I have reversed the order of calling of certain sutras to take care of the parasutra precedence (vipratishedhe param kār्याṃ) automatically. As per Cardona (1976), during a derivation, Pāṇini called functions in an order which is different that the spatial order in Astadhyayi, as the latter rules may serve the former rules. Also, one can not separate definition rules or metarules from the rules they serve to interpret. The recursive approach makes this convenient to bring these together. Figure 4 explains the high level architecture of the proposed model.

7.0 Why recursion in language modeling?

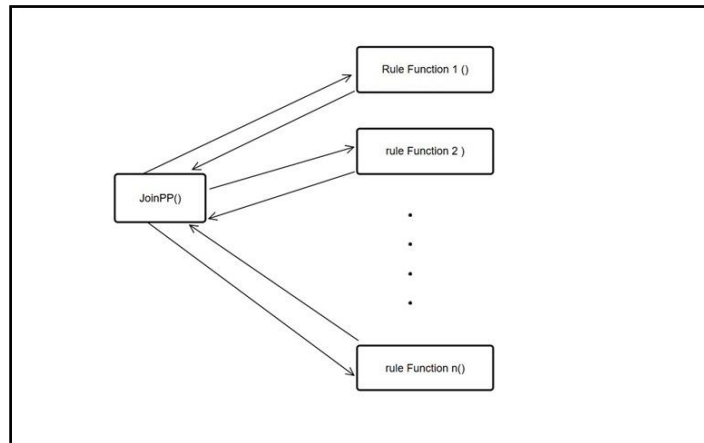
1. Recursion is ideal if the problem has a lot of layers in it. Pāṇinian derivation is a tree like recursive structure and we need to call the joinpp () function to check the applicability of the rules in each of the tree nodes.
2. Recursion makes the code more readable as the lines of code is considerably reduced and tracking is easier.
3. Recursion uses the system's call stack automatically, so you don't need to implement your own stack structure.
4. Automatic backtracking occurs in recursion, whereas in loops we need to implement manual backtracking mechanisms
5. Infinite Generativity: Recursion allows embedding structures within structures
6. Any language syntax is hierarchical, not linear. Recursion naturally represents this hierarchy, whereas loops only move linearly through sequences. Recursion is preferred in linguistics because it reflects the hierarchical, generative nature of language, whereas loops only capture repetition without embedding.
7. Pāṇini used adhikāra + anuvṛtti to build his recursive grammar that can generate infinite expressions from finite rules.

8.0 The Power of Mutual recursion: Mutual recursion is a robust design pattern wherein the higher order functions are calling each other, with a terminating base condition. This is widely used in functional

programming to avoid loops and for better co ordination between the large number of functions. In subanta generation process, we have a large number of functions each symbolizing one sutra in Ashtadhyayi.

Join PP () is the wrapper function used to share states between the sutra functions. It contains all the function calls to the grammar rules. The grammar rules call this Join PP () back , causing mutual recursion. The pratipadika and pratyaya undergo changes by recursively calling these functions. The terminating condition is pratyaya becoming null. The following figure describes the mutual recursion.

Figure 3: Mutual recursion in Pāṇinian grammar



9.0 Relevance of nadī and ghi designations during subanta generation

9.1 The designation nadī

In Aṣṭādhyāyī, nadī designation plays an important rule in deciding the applicability of certain rules. Table 4 shows the rules to decide whether a word get nadī designation or not.

Table 4- Assigning nadī designation

Rule	Purpose	Implementation in the proposed model
A 1.4.3 yu stryākhyo nadī	the strīlinga words ending with ‘ī’ / ‘ū’. eg: ‘nadī’, ‘strī’, ‘vadhū’	If ((linga==‘s’) and endingLetter in [‘i’,‘u’]) Then nadiDes = True
A 1.4.4 neyang uvaṅ sthānāv astrī	This rule excludes certain ‘ī’ / ‘ū’ ending words from getting nadī designation. The ‘ī’ / ‘ū’ ending words other than ‘strī’, which get iyaṅ / uvaṅ ādeśa when joined with a pratyaya starting with vowel, will not get nadī designation. Eg: ‘dhī’, ‘bhū’, ‘sudhī’	If ((linga==‘s’) and (endingLetter in [‘i’,‘u’]) and (inputWord <> ‘strī’) and (inputWord in iyangUvangList)) Then nadiDes = False [Note: iyangUvangList stores a list of words getting iyaṅ / uvaṅ ādeśa when joined with a vowel]
A 1.4.5 vāmi	those sabdas for which nadī designation is prohibited by rule 1.4.4, get nadī designation optionally when the pratyaya is ‘ām’ from ṣaṣṭhī bahuvacana.	If (inputWord in iyangUvangList) and (nadiDes==False) and (pratyaya ==‘ām’) Then vikalpaNadiDes = True

A 1.4.6. niti hrasvaś ca	Along with the words mentioned in rule 1.4.4, the ‘i’/‘u’ ending strilinga words too, optionally get nadī designation in the presence of the pratyaya ‘ne’/ ‘nas’/’nasim’/ ‘ni’ Eg: ‘matī’, ‘dhenu’	If ((endingLetter in [‘i’,’u’]) and (linga ==‘s’)) Then vikalpaNadiDes = True
-----------------------------	---	--

Uses of nadī designation are mentioned in Table 5.

Table 5 – Uses of nadī designation

Rule	Use of nadī designation
A 7.3.107 ambārthanadyoḥ hrasvaḥ	In sambodhana ekavacana, the words with meaning ambā and the words with nadī designation gets hrasva adesa. Eg: he amba, he gauri, he nadi etc. As we are not implemented sambodhana vibhakti, this rule is out of scope in our current system.
A 7.3.112 āṇ nadyāḥ	In the presence of pratyaya ‘ne’/’ nas’/’ nasim’/’ni’ nadī designated word gets ‘āṭ’ āgama
A 7.1.54 hrasvānadyāpo nuṭ	The ‘ām’ pratyaya gets a ‘nuṭ’ āgama when followed by a nadī designated word/word ending in short letter/strilinga words ending with ‘ā’
A 7.3.116 nēram nadyāmnibhyaḥ	The pratyaya ‘ni’ gets ‘ām’ ādesa when it comes after a nadī designated word/strilinga word ending with ‘ā’ / a word that ends with ‘nī’.

9.2 The designation ghi

Table 6 shows the rules used for implementing ghi designation.

Table 6 – Assigning ghi designation

Rule	Purpose	Implementation in the proposed model
A1.4.7śeṣo khyāsakhi	Except for the word sakhi, all ‘i’/‘u’ ending words get the designation ghi whenever they do not get the nadī designation.	If ((nadiDes == False) and (inputWord notin[‘sakhi’,’pati’]) and (endingLetter in[‘i’,’u’])) then ghiDes = True
A 1.4.8 Pati-samāsa eva	The word ‘pati’ gets ghi designation only when used in a samāsa. Eg: Gaṇapati and Bhūpati	See the implementation in the above row

A 1.4.9 ṣaṣṭhī-yuktaś chandasi vā	The word ‘pati’, when associated with a word of ṣaṣṭhī vibhakti, optionally gets the ghi designation in Vedic Sanskrit. Eg: paśūnām pataye namaḥ in Śrī Rudram of Yajurveda.	This is out of scope for the proposed tool as we are not storing user intention for the derivation of a word here. This can be implemented in the next semantic level.
-----------------------------------	---	--

Table 7 shows the uses of ghi designation in subanta generation process.

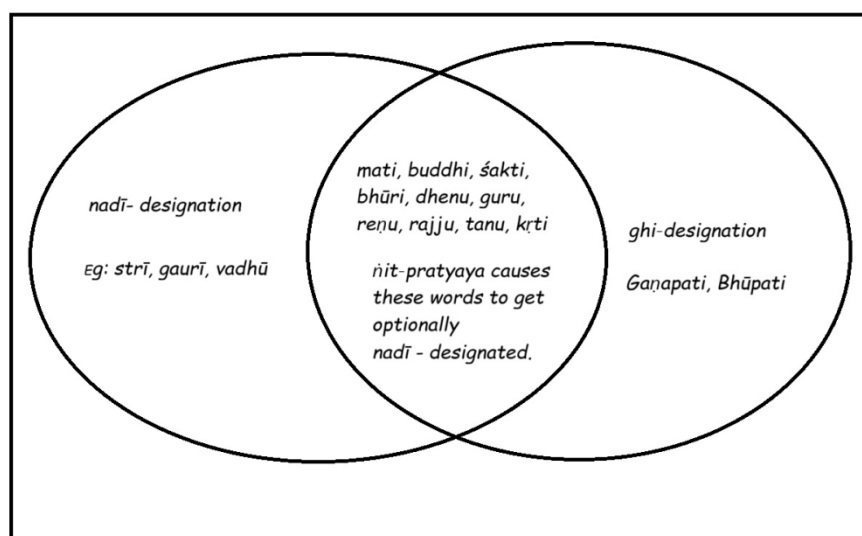
Table 7- Uses of ghi designation

Rule	Use of ghi designation
7.3.111 Gherniti	The pratyayas ñe/ ñas / ñasim/ ñi cause guṇa in ghi designated aṅgas
7.3.119 acca ghe	ghi designated aṅga gets ‘a’ ādeśa , pratyaya becomes ‘au’
7.3.120 aṅo na striyām	For the ‘ghi’ designated aṅga, the pratyaya ‘ ṭā ’ becomes ‘nā’

9.3 Computational method for nadī and ghi designations: Please refer to figure 5 and figure 6 to understand how nadī and ghi designations are treated computationally. To implement nadī designation, we have kept two lists; one for the words which are always getting the designation nadī and the other for the words which are susceptible to optional nadī designation. For optional nadī designated words, there is a dedicated page in the output, for giving room for optional forms without affecting the normal subanta forms. As we are storing nadī designation as a flag which is set only initially, and is not reset for each cell, this context is retained throughout the derivation.

Representation in Boolean Algebra: The designations nadī and ghi can be easily understood from figure 5 in terms of set theory in Mathematics. The words ‘matī’, ‘dhenu’ are placed in the intersection of the sets, but that does not mean that these words get both the designations at the same time. Conditionally, these words get either of these designations. Our model generates both the forms of these words in the 3D matrix output.

Figure 4: nadī and ghi designations



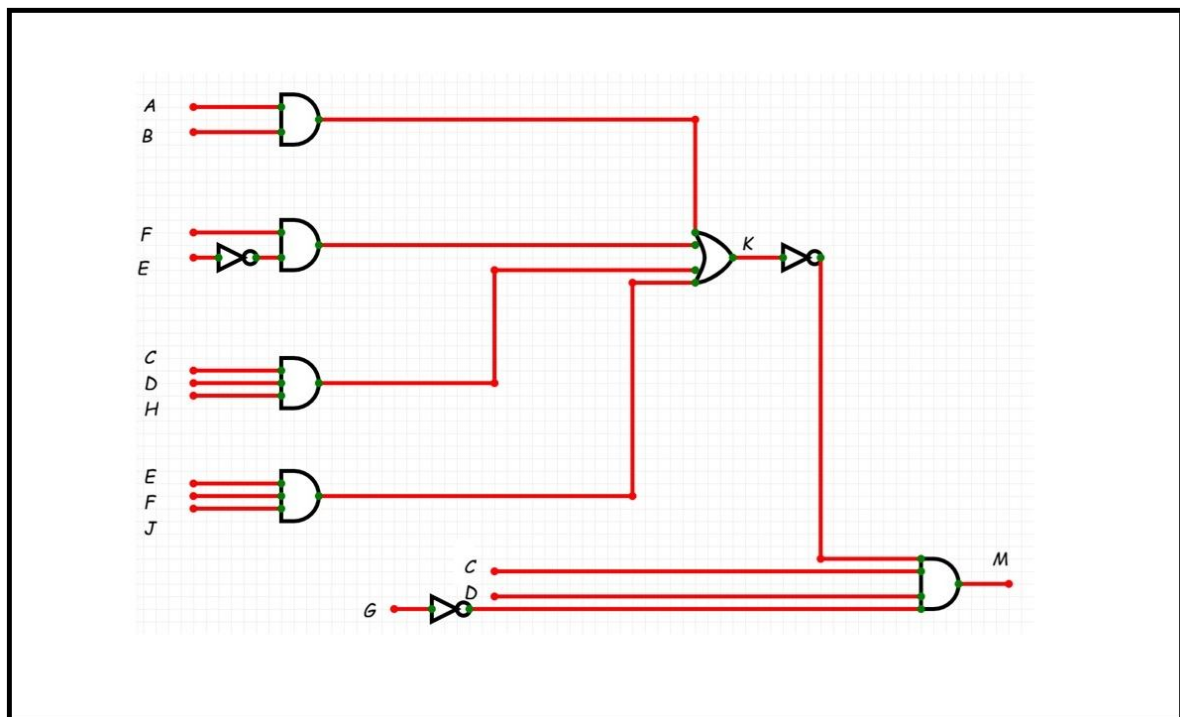
I have tried to represent nadī designation and ghi designation using logic gates in Boolean algebra.

Suppose we have the following user inputs.

Input	Description
A	‘ī’ ending word
B	‘ū’ ending word
C	‘i’ ending word
D	‘u’ ending word
E	words getting iyaṅ / uvaṅ ādeśa
F	word is strī
G	word is sakhi
H	pratyaya is ñe/ ñas/ ñasim/ ñi
J	ām pratyaya
K	nadī designation
M	ghi designation

With these inputs, we can represent nadī and ghi designations as shown in Figure 6, with the help of AND , OR and NOT gates.

Figure 5: Logic gate network for nadī and ghi designations



10.0 Limitations of the model: The current prototype handles only Ajanta sabdas. But the scalable design makes it easy for incorporating the halanta sabdas in future.

We have tried to reduce the user inputs, yet not able to eliminate the user input for linga. Further researchers can think of adding that level of intelligence to the system.

The system keeps the nadī designated words in a list now. The above mentioned logic gate network implementation is targeted in the next prototype.

Further research can leverage our python based framework for halanta generation, tiñanta generation, samāsa add-ons, vārtika add-ons, parser development and integration with various NLP tools to aid in the field of digital humanities.

The proposed model is based on the prescriptive nature of classical Sanskrit. But with the help of artificial neural networks, if the human cognition and decision making can be modelled successfully, the descriptive features of Vedic Sanskrit like vivakṣā and vyangya also can be incorporated in the future implementations.

References

- Amba Kulkarni (2009). Computer Simulation of Aṣṭādhyāyī. Third International Symposium on Sanskrit Computational Linguistics, Hyderabad.
- Amba Kulkarni (2009). Computer Simulation of Aṣṭādhyāyī. Third International Symposium on Sanskrit Computational Linguistics, Hyderabad.
- Amba Kulkarni (2018). PAIAS: Pāṇini Aṣṭādhyāyī Interpreter As a Service, World Sanskrit Conference, Vancouver.
- Barend A Van Nooten (1967). Pāṇini's Replacement Technique and the Active Finite Verb. Language, Vol. 43, No. 4, by the Linguistic Society of America.
- Barend A Van Nooten (1969). Pāṇini's Theory of Verbal Meaning. Foundations of Language, Vol. 5, Issue 2, by Mouton Publishers.
- Barend A Van Nooten (1970). The Vocalic Declensions in Pāṇini's Grammar. Indo-Iranian Journal, Brill Academic Publishers, Leiden, Netherlands.
- Barend Faddegon (1936). Studies on Paninis grammar. North-Holland Publishing Co., Amsterdam.
- Bronkhorst, Johannes (1980). Asiddha in the Astadhyayi : a misunderstanding. Indo-Iranian Journal, Brill Academic Publishers, Leiden, Netherlands.
- Cardona, George (1976). Pāṇini: A Survey of Research. Mouton (The Hague/Paris).
- Dr. Dhaval Patel and Dr. Sivakumari Katuri (2015). Subanta generator. 16th World Sanskrit Conference, Bangkok.
- Frits Staal (1965), Euclid and Pāṇini. Philosophy East and West.
- Frits Staal (1965). Context sensitive rules in Pāṇini. Foundations of Language, Vol. 5, Issue 1, Mouton Publishers (The Hague/Paris).

- George Cardona (1970). Some principles of Pāṇini's grammar. *Journal of the American Oriental Society (JAOS)*, Vol. 90, No. 2
- George Cardona (1974). Pāṇini's kāraka rules: agency, animation, identity. *Journal of the American Oriental Society (JAOS)*, Vol. 94, No. 2
- Gerald Penn and Paul Kiparsky (2012). On Pāṇini and the Generative Capacity of Contextualized Replacement Systems. *Studia Orientalia*, Vol. 112, Finnish Oriental Society (Societas Orientalis Fennica, Helsinki).
- Jan E.M. Houben (1999). Meaning statements' in Pāṇini's grammar: on the purpose and context of the *Aṣṭādhyāyī*".
- Kulkarni, Amba and Bharati, Akshar (2007). *Sanskrit and Computational Linguistics*.
- Kulkarni, Amba and Shukl, Devanand (2009). *Sanskrit Morphological Analyzer- Some issues*. Third International Symposium on Sanskrit Computational Linguistics Hyderabad.
- M D Pandit (1974). Formal and Non formal in Pāṇini. *Indo-Iranian Journal*, Brill Academic Publishers, Leiden, Netherlands.
- Madhav Deshpande (1990). The Process of Semantic Interpretation in Pāṇini. *Proceedings of the International Seminar on Sanskrit and Related Studies (BORI, Pune)*.
- Madhav M Deshpande (1985). Ellipsis and syntactic overlapping: current issues in Pāṇinian syntactic theory. *Bhandarkar Oriental Research Institute (BORI), Pune*.
- Madhav M Deshpande (1991). Pāṇini in the context of modernity. *Indo-Iranian Journal*, Brill Academic Publishers, Leiden, Netherlands.
- Madhav M Deshpande (1991). Prototypes in Pāṇinian syntax. *Indo-Iranian Journal*, Brill Academic Publishers, Leiden, Netherlands.
- Malcolm D Hyman (2003). From Pāṇinian Sandhi to finite State Calculus. *Sanskrit Computational Linguistics (workshop proceedings, Brown University / University of Hyderabad collaborations)*
- Noam, Chomsky (1965). *Aspects of the Theory of Syntax*. The MIT Press.
- Paul Kiparsky, Frits Staal (1969), Syntactic and Semantic relations in Panini. *Foundations of Language*, Vol. 5, Issue 1, Mouton Publishers (The Hague/Paris).
- Paul Thieme (1983). Meaning and Form of the 'Grammar' of Pāṇini. *Indo-Iranian Journal*, Brill Academic Publishers, Leiden, Netherlands.
- Peter M. Scharf, Malcolm D. Hyman (2002). Linguistic issues in encoding Sanskrit. *Indo-Iranian Journal*, Brill Academic Publishers, Leiden, Netherlands.
- Peter Scharf (1992). The Sanskrit Library: A Proposal for the Development of a Digital Sanskrit Library. *International Journal of Lexicography*, Vol. 5, Issue 3, Oxford University Press.
- Peter Scharf (2007). Modeling Pāṇinian grammar. *Indo-Iranian Journal*, Brill Academic Publishers, Leiden, Netherlands.
- S .D. Joshi (2009). *Background of Aṣṭādhyāyī*.

- S. D. Joshi and J. A. F. Roodbergen (1991). The Siddha-Principle as Presupposed by Some Pāṇinisūtras. Indo-Iranian Journal, Brill Academic Publishers, Leiden, Netherlands.
- S.D Joshi (1977). The ordering of the rules in Pāṇini's grammar. Indo-Iranian Journal, Brill Academic Publishers, Leiden, Netherlands.
- S.D Joshi, Paul Kiparsky (1979). Siddha and asiddha in Pāṇinian phonology. Indo-Iranian Journal, Brill Academic Publishers, Leiden, Netherlands.
- S.D Joshi, Paul Kiparsky (2006). The extended siddha principle. Indo-Iranian Journal, Brill Academic Publishers, Leiden, Netherlands.
- Samir Sohoni & Malhar Kulkarni, Amba Kulkarni (2018). A Functional Core for the Computational Aṣṭādhyāyī, World Sanskrit Conference, Vancouver.
- Srisa Chandra Vasu (2009). The Aṣṭādhyāyī of Pāṇini vol II. Motilal Banarsidass
- Subhash Chandra and Diwakar Mishra (2006). Sanskrit Subandha Generator

Citation: Mohan. M. & S. K. L., (2026) “The Use of Mutual Recursion in Matrix Based Nominal Declension Model Following Pāṇinian Method”, *Bharati International Journal of Multidisciplinary Research & Development (BIJMRD)*, Vol-4, Issue-08, August-2026.