Computational Structure of the *Aṣṭādhyāyī* and Conflict Resolution Techniques

Sridhar Subbanna[1] and Varakhedi Shrinivasa[2]

[1] Rashtriya Sanskrit Vidyapeetha, Tirupati, India,
sridharsy@gmail.com,
[2] Samskrita Academy, Osmania University, Hyderabad, India,
shrivara@gmail.com,

Abstract. *Pāṇini*'s *Aṣṭādhyāyī* consists of sūtras that capture fundamentals of Sanskrit language and define its structure in terms of phonology, morphology and syntax. The *Aṣṭādhyāyī* can be thought of as an automaton to generate words and sentences. In object oriented programming terms, prescribing sūtras are objects having its transformation rule as its method. The meta rules or paribhāṣā sūtras and paribhāṣā vārtikās define the flow of the program. During application of sūtras, conflicts may arise among two or more competing sūtras. In this paper, computational structure of the *Aṣṭādhyāyī*, sūtra objects observing the environment, tree representation of sūtras and mathematical representation of conflict resolution techniques are presented.

1 keywords

Pāṇini, Aṣṭādhyāyī, Sanskrit, Vyākaraṇa, Sūtra, Computer Modelling, Conflict Resolution, Object Oriented Programming, Mathematical Representation

2 Introduction

The *Aṣṭādhyāyī*[1] (śabdānuśāsanam) deals with the generation of words and sentences of Sanskrit Language and also provides a base for the analysis of the same in general. Its algebraic nature and comprehensiveness illustrate that its structure can be described as a machine generating words and sentences of Sanskrit.
The *Aṣṭādhyāyī*[9] consists of around 4000 sūtras that describe the fundamentals of Sanskrit language in terms of phonology, morphology and syntax. The structure consists of definitions, rules, and meta-rules that are context-sensitive and operate in sequence or recursively[6].

Generally these rules are classified into three groups:

1. Rules of definition and meta-rules (saṃjña and paribhāshā)
2. Rules of affixation of roots (dhātu and prātipadika) and

3. Rules of transformation for stems and the suffixes.

The computer programs have exactly the same features of context-sensitive rules, recursion and sequential rule application[7]. Prescribing sūtras are context-sensitive rules, paribhāṣās define the flow and hence the *aṣṭādhyāyī* can be thought of as an automaton that generates Sanskrit words and sentences.

3 Computational Structure of the *Aṣṭādhyāyī*

The *Aṣṭādhyāyī* consists of sūtras that are organized in a systematic and mathematical manner. The application of sūtras follows a systematic procedure in deriving the final form of words from roots and affixes. In object oriented programming[8], objects have state and behaviour as two characteristics. State is the data part and behaviour is the method or function. Here, sūtra as an object will be observing the state of environment and applies as its transformation rule if the condition satisfies. Thus, object oriented programming suits best to model the aṣṭādhyāyī.

3.1 Classification and Representation of sūtras

Traditionally the sūtras in the *Aṣṭādhyāyī* are classified into 6 groups.[3]

1. *saṃjñā sūtras*: assign saṃjña-s or labels.
2. *paribhāṣā sūtras*: meta-rules for interpretion and application of sūtras.
3. *vidhi sūtras*: prescribing rules for ādeśa(substitution/deletion) and āgama(insertion).
4. *niyama sūtras*: conditioning rules that confines vidhi sūtra with some additional conditions.
5. *atideśa sūtras*: extension rules to extend the existing rules in another scenario.
6. *adhikāra sūtras*: governing rules adopted for division of contents and gives meaning to the successive sūtras.

For our convenience these sūtras can be grouped as follows.

1. Meta rules or helping rules (2 & 6 above)
2. Prescribing rules (1,3,4 & 5 above)

paribhāṣā and *adhikāra* sūtras are meta rules. *Paribhāṣā* sūtras are utilized to intrepret the sūtras and *adhikāra* sūtras are meant to define the boundary of a particular topic domain. Rest are prescribing rules that define transformation functions. The meta rules help in interpretation[4] and application of rules. This will be discussed in detail in the conflict resolution section.
The prescribing rules can be grouped under each nested topic (*ekavākya*). Each nested group can be represented as a tree of sūtra objects. *utsarga* or general

[3] *saṃjñā ca paribhāṣā ca vidhir niyama eva ca*
atideśodhikāraśca ṣaḍvidham sūtralakṣaṇam

sūtra will be root node and *apavāda* (exception) sūtras will be the child nodes. The sūtras that are having different conditions under the same topic will be the sister nodes. During the application of sūtras a single tree traversal algorithm can be used to determine the sūtra that is to be applied. It has to be seen whether this process of tree building can be automated.
The following examples will explain the nature of representation.

Example 1: The Figure 1 shows representation of mutually exclusive *guru* and *laghu saṃjñas.*

1. *hrashvam laghu 1.4.10*
2. *saṃyoge guru 1.4.11*
3. *dhīrghaṃ ca 1.4.12*

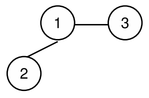


Fig. 1. Tree representation for *guru* and *laghu saṃjña* rules

Example 2: At various different places the *it saṃjña* is used. The Figure 2 shows its definition.

1. *upadeśe ajanunāsika it 1.3.2*
2. *halantyam 1.3.3*
3. *na vibhaktau tusmāḥ 1.3.4*
4. *ādiḥ ñiṭudavaḥ 1.3.5*
5. *ṣaḥ pratyayasya 1.3.6*
6. *cutū 1.3.7*
7. *laśakvataddhite 1.3.8*

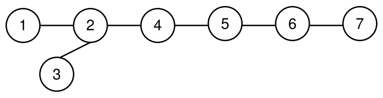


Fig. 2. Tree representation for *it saṃjña* rules

Example 3: The Figure 3 shows the representation of the sandhi sūtras. The sūtra 2, 3, 4 & 6 are sister nodes as their conditions for applying or domain are different. The sūtra 12 is apavāda to both sūtra 2 and sūtra 6. There may be many such cases in whole of the *aṣṭādhyāyī*.

1. *saṃhitāyāṃ 6.1.72*
2. *iko yaṇaci 6.1.77*
3. *ecoyavāyāvaḥ 6.1.78*
4. *vānto yi pratyaye 6.1.79*
5. *dhātostannimittasaiva 6.1.80*
6. *ādgunaḥ 6.1.87*
7. *vṛddhireci 6.1.88*
8. *etyedhatyūṭhsu 6.1.89*
9. *eṅi pararūpaṃ 6.1.94*
10. *omāngośca 6.1.95*
11. *ato gune 6.1.97*
12. *akaḥ savarṇe dīrghaḥ 6.1.101*
13. *prathamayoḥ pūrvasavarṇaḥ 6.1.102*

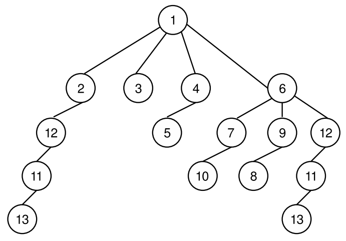
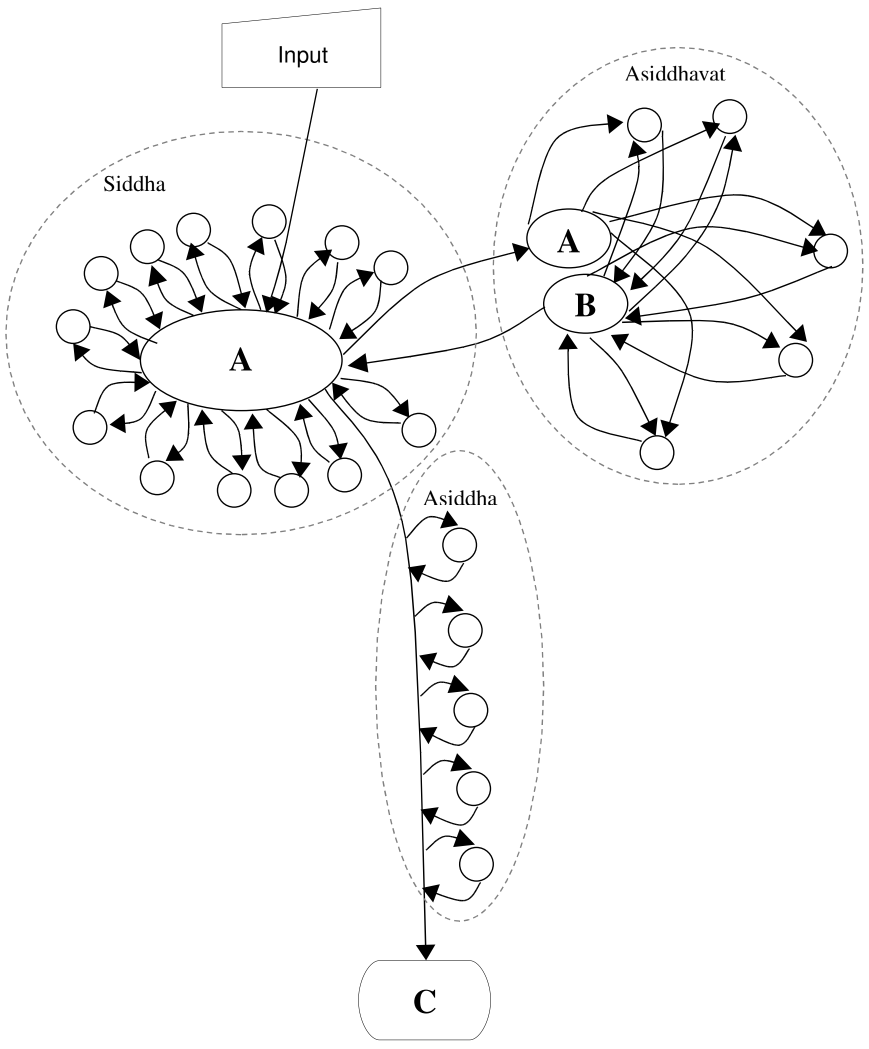


Fig. 3. Tree representation for *ac sandhi* rules

3.2 Computational structure

The root nodes will be observing the environment (subject). All the nodes that find the condition send a request for their application. After getting requests, conflict resolver adopts the resolution techiques and selects only one node among them for application. Then the tree with that as root node will be traversed to find the exact sūtra for application. The sūtra object contains all the information about the sūtra. The sūtra will be applied to update the environment. The Figure 4 represents the overall structure of the aṣṭādhyāyī.
The output states and other sūtras are siddha (visible) to all the sūtras of the *aṣṭādhyāyī* except where it is explicitly mentioned as asiddha (invisible). There are three places where the asiddhatva is explicitly mentioned.



○	Root Level Nodes observing the Environment
A, B	Environment
C	Output

Fig. 4. Computational Structure of the *Aṣṭādhyāyī*

1. *Asiddhavadatrābhāt 6.4.22*
2. *Ṣatvatukorasiddhaḥ 6.1.86*
3. *Pūrvatrāsiddham 8.2.1*

In general, whenever conditions for a sūtra are satisfied, that sūtra will be applied. Only one sūtra will be applied at a time. In other words no two sūtras can be applied simultaneously. However, the sūtras of asidhavat prakaraṇa that are all having the same condition, can be thought of as applied simultaneously. The sūtras in the last three pādas, that is, tripādī need to be applied in the sequential order. The Figure 4 explains this model.

In Figure 4, the siddha block contains the sūtras of sapādasaptādhyāyī minus the asiddhavat prakaraṇa sūtras (6.4.22 to 6.4.175). The asiddhavat block contains the sūtras of asiddhavat prakaraṇa. The asiddha block contains the tripādi sūtras. The ovals A and B represent the output states and the C represents the output of the system (final state). The small circles represent sūtra group that takes the input state and changes the state. There will be a state transition function defined for each sūtra object. While in a particular state, if the conditions for root sūtra are satisfied, then complete tree with this as a root node is traversed in order to find out which sūtra within this tree is to be applied, accordingly it transforms the state by invoking the function defined for that sūtra. In the siddha block, all the states are visible and can be input state to any sūtra in the same block.

While in the siddha blok, if conditions of any of the sūtras in the *asiddhavat* block are met, initially, state B is same as state A. The sūtras here in the asiddhavat block take both A and B as input, and transforms only B. So, all other sūtras check for their condition in state A and operates on B. When there are no more conditions for the sūtras in this block, the state A is made same as state B.

When there are no more conditions for sūtras in the siddha block or in the *asiddhavat* block then the state is sequentially updated by the sūtras in asiddha block that are applicable in that particular state and the final output state will be C.

There is a need to develop techniques that make the explanation of the computational structure of the *aṣṭādhyāyī* easier. The techniques of representing and manipulating knowledge should be developed and create computing algorithms that have computational abilities.

Example 1. When the state A is *vana + ṭā*(case 3,num 1)], the sūtra *ṭā ṅasiṅasām inātsyāḥ* (7.1.12) in the siddha block finds its condition. After its invocation A is changed to *vana + ina.* Then the sūtra *ād gunaḥ* (6.1.87) in the siddha block finds its condition and gets invoked. A now becomes *vanena.* No other sūtra finds the condition, hence A is passed to C. Now C has the final form *vanena.*

Example 2. When the state in A is *śās + hi* sūtras in the Asiddhavat block find the condition. Initially, state B will be same as state A. Now the sūtra *śā hau* (6.4.35) is invoked and then B is changed to *śā + hi.* The sūtra *hujhalbhyo* *herdhiḥ* (6.4.101) also finds the condition in A and is invoked. B is now changed to *śā + dhi.* No more sūtras finds the condition, so A is made same as B. None of the sūtras either in siddha block or the asiddha block find the condition in A. Hence A is passed to C without any tranformation giving the final word [śādhi].

Example 3. When the state A is *dvau + atra.* The sūtra *ecoyavāyāvaḥ* (6.1.78) finds the condition. This sutra is now applied, A is changed to *dvāv + atra.* No sūtra in the siddha block or the asiddhavat block finds condition in A. The sūtra *lopaḥ śākalyasya* (8.3.19) in the asiddha block finds the condition. After its invocation it is changed to *dvā + atra.* This state is not visible to the sutras either in the siddha block or in the asiddhavat block. Even though there is condition for *akaḥ savarṇe dīrghaḥ* (6.1.101) in the siddha block, this state is not visible to this sūtra. Hence this sūtra cannot be applied. The sūtras that are following the sūtra *lopaḥ śakalyasya*(8.3.19) have visibility of this state but do not find condition. Hence none of the sūtras are applied, yielding the final form as *dvā atra.*

The asiddhatva is the base for the above shown computational structure of the aṣṭādhyāyī. This whole structure developed on the basis of Siddha-asiddha principle resolves many conflicts in the whole application.

4 Techniques for Conflict Resolution

In general, it can be thought of, as all the sūtras will be observing the changes in a given state (prakriyā) and wherever they find their condition (nimitta), they would come forward to apply their function - kārya on that state, as a result the state would get modified. However there are possibilities of conflict that arise between many sūtras in this process, as many of them may find condition in a particular state and all of them would try to apply. At any point of time only one modification is possible in the given state. Sometimes all the sūtras may be applied in a particular order or one is applied and others are rejected. Due to this, complexity of the program will not only increase, but also results into a paradoxical situation.

There are different paribhāṣā sūtras, paribhāṣā vārtikās to resolve these conflicts. Considering only the *aṣṭādhyāyī* sūtras it may not be possible to resolve conflicts under all circumstances, hence vārtikas also should be taken into account in conflict resolution. We are trying to represent conflict resolution techniques mathematically, that can be directly adopted in a computer program.

4.1 Conflict resolution through sūtras

There are only few sūtras that directly prescribe the flow or resolve conflict.

4.1.1 *Vipratiṣedhe paraṃ kāryaṃ* 1.4.2 This sūtra says when there is conflict then para (the later one in a sequential order) sūtra is to be applied. According to Patanjali's interpretation *para* means *iśṭa*[2] and not the later one in the sūtra order. There is another controversy among traditional and modern Grammarians in interpretation of this sūtra. The traditional view is that this sūtra is globally applicable across the aṣṭādhyāyī. The modern thinking is that this sūtra is locally applicable to the *ekasaṃjñā* domain which runs through 2.2.38. This needs to be looked into greater detail to see the cases on which they have taken their stands.

4.1.2 *Siddha* and *Asiddha* All the sūtras are treated as *siddha* (visible) to each other unless specified explicitly as *asiddha* (invisible) in the sūtras.[4]
There are two view points on *asiddha* concept namely *Śāstrāsiddha* and *Kāryāsiddha*. If a sūtra is *Śāstrāsiddha*, if sūtra itself is invisible to another sūtra, and it is *Kāryāsiddha* if the sūtra is visible but its *kārya* (result) is *asiddha* to the other. These two alternative ideas need to be examined.

Example *śivacchāyā*[10]

1. *śivachāyā*
2. *śivatchāyā 6.1.73*
3. *śivadchāyā 8.2.39*
4. *śivajchāyā 8.4.40*
5. *śivacchāyā 8.4.55*

The environment is *śivachāyā*. The sūtra *che ca 6.1.73* is applied and is changed to state 2. Now the sūtras *jhalāṃjaśonte* 8.2.39 and *stoḥścunāścuḥ* 8.4.40 find the condition in this context. Since 8.4.40 is asiddha to 8.2.39, 8.2.39 is applied first and state is updated to state 3. Again, 8.4.40 and *khari ca* 8.4.55 has nimmita for application, similar to earlier one, here also since 8.4.55 is asiddha to 8.4.40, 8.4.40 is applied first and state is update to state 4. Now, 8.4.55 gets a chance for its application and environment is moved to state 5. The sūtra *coḥ kuḥ* 8.2.30 cannot see the environment and does not come forward. This way application of rule 8.2.30 is prevented after the final form.

4.2 Conflict resolution through *vartikas*

4.2.1 *Para-nitya-antaraṅga-apavādānām uttarottaraṃ balīyaḥ* [3] This paribhāṣā gives us criterion for conflict resolution. The priority is *apavāda, antaraṅga, nitya* and *para*. We explain below how we model these priorities.

1. *utsarga - apavāda*
   utsarga and *apavāda* (General and Exception) sūtras are static; this information is embedded in the tree structure itself. During the application of

[4] *pūrvatra asiddham 8.2.1, asiddhavadatrābhāt 6.4.22, ṣatvatukorasiddhaḥ 6.1.86*

sūtras the tree is traversed in such a way to determine the apavāda sūtra. When two sūtras have *utsarga* and *apavāda* relation then *apavāda sūtra* is selected and applied, *utsarga sūtra* is rejected.

2. *antaraṅga - bahiraṅga*
   The definition *alpāpekṣaṃ antaraṅgaṃ* and *bahvapekṣaṃ bahiraṅgaṃ* can be used to determine the *antaraṅgatva* and *bahiraṅgatva* of the any two sūtras. When the sūtras have *antaraṅga* and *bahiraṅga* relation then *antaraṅga sūtra* is selected and applied, *bahiraṅga sūtra* is rejected. The *antaraṅga* and *bahiraṅga* are relative to the context and can be mathematically determined. The definition could be formalized as follows.
   Let $f(X, \phi)$ return the number of conditions that are required for sūtra X to apply in the given state ϕ.

   ```
   if $f(X, \phi)$ is less than $f(Y, \phi)$
   then X is Antaraṅga and Y is Bahiraṅga
   else Y is Antaraṅga and X is Bahiraṅga
   endif
   ```

   Example, Let X = *sarvādīni sarvanāmāni* 1.1.27 and Y=*prathamacharama tayalpakatipayanemāśca* 1.1.33. When the state is *ubhaya jas.* Whether *ubaya* gets *sarvanāmasaṃjñā* by X or optionally by Y is the question.
   $f(X, \phi) = 1$ as there is only one condition for X to apply. The condition is ubhaya's existance in *sarvādi gana.*
   $f(Y, \phi) = 2$ as there are two conditions for Y to apply. One condition is *ubhaya* is a *tayappratyayānta* and second condition is *jas pratyaya* in front of it.
   Since, $f(X, \phi) < f(Y, \phi)$, X is *antaraṇga* and Y is *bahiraṇga.* Since *antaraṇga* is preferred, here *ubhaya* will get the *sarvanāmasaṃjñā* by X, but not optional *sarvanāmasaṃjñā* by Y.

3. *Nitya - Anitya*
   The definition of *nitya* and *anitya* is given as *kṛtākṛtaprasaṅgi nityam tadviparītamanityam.* The *nitya* and *anitya* are relative to the context and can be mathematically defined.
   Let there be sūtras X and Y that have the condition for its application in a particular state. If X is applicable irrespective of the application of Y then X is said to be *nitya.* On application of Y if X loses its condition for application then it is said to be *anitya.* It can be defined mathematically as follows.
   Let $f(X, \phi)$ returns $\phi\prime$,transformed state after application of sūtra X in the ϕ state and returns zero if sūtra X is not applicable in the ϕ state.

   ```
   if $f(X, f(Y, \phi))$ is not zero
   X is nitya and Y is anitya
   else
       if $f(Y, f(X, \phi))$ is not zero
       X is anitya and Y is nitya
       else X and Y do not have the nitya anitya relation.
   ```

Comsider the case when the state is *tud ti* then the sūtras *tudādibhyaḥ śaḥ*(3.1.77) and *pugantalaghūpadasya ca* (7.3.86) both find their condition. Let X = *tudādibyaḥ śaḥ(3.1.77)* and Y = *pugantalaghūpadasya ca*(7.3.86). Then $f(X, \phi)$ =*tud sha ti* and $f(Y, f(X, \phi)) = 0$ because Y is not applicable in the state *tud sha ti*. So Y is anitya. Consider, $f(Y, \phi)$ = *tod ti* and $f(X, f(Y, \phi))$ =*tod sha ti* i.e, is not equal to zero. Hence X is nitya.

4. *Para - Apara*
   Para and Apara (Posterior and Prior): The sūtra that is positioned before in the *aṣṭādhyāyī* order is *apara* and the one later is *para*. Between the *para* and *apara* sūtras, *para sūtra* is selected and applied, apara sūtra is rejected. The para-apara relation can be determined based on the *sūtra saṅkhyā*.
   For example, *bahuvacane Jhalyet* 7.3.103 is *para* to *supi ca* 7.3.102 as 7.3.103 is later than 7.3.102. Let $f(X)$ returns the sūtra number in the aṭādhyāyī.

```
if $f(X) > f(Y)$
then X is para and Y is apara
else Y is para and X is apara
```

4.2.2 *Varṇādāṅgaṃbalīyo bhavati* There are two rules one acting on aṅga[5] and other on phoneme. In this case, the sūtra on aṅga should be applied first.

4.2.3 *Lakṣye lakṣaṇaṃ sakṛdeva pravartate* This vartika prevents the recursion. Only once any sūtra should be applied in a particular environment[5]. Here sūtra means group of sūtras under the same topic (ekavākya). In the tradition, analogy of *takrakauṇḍiṇya*[6] is given to expain this concept.

5 Conclusion

The next objective is to implement this as a computer program and see whether we can optimize the sūtras or evaluate the necessity of the vārtikas. Such an implimentation would not only confirm the automata nature of Paninian System but also exhibit the complexities of the system and feasibility of resolutions to them by employing techniques shown by Pāṇinian Tradition. Our current effort could be a first step towards achieving that goal.

Acknowledgment Authors thank Lalit Kumar Tripati and Amba P Kulkarni for useful discussions at various stages of the work.

[5] *yasmāt pratyayavidhiḥ tadādi pratyaye aṅgam - 1.4.13*

[6] *Brāhmaṇebhyo dadhi dīyataṃ, takraṃ kauṇḍiṇyāya.* Here there are two rules. *Brāhmaṇebyo dadhi dīyatāṃ* is first one, *takraṃ kauṇḍiṇyāya* is the second one. Once *takraṃ* is given to *kauṇḍiṇya*, again *dadhi* will not be given according to first rule.

References

[1] Swami Prahlada Giri and Satyanarayanashastri : *Paṇinīyaḥ Aṣṭādhyāyī* Krishnadas Academy, Varanasi (1984)

[2] Bhargava Sastri Bhikaji Josi : The Vyākaraṇamahābhāṣya of Patañjali, Chaukhamba Sanskrit Pratishtan, Delhi(1872)

[3] F. Kielhorn : Paribhāṣenduśekara of Nāgojibhaṭṭa, Parimal Publications, Delhi.

[4] Srinarayana Mishra: *Kāśikā*, Chaukamba Samskrita Samsthan, Varanasi (1979)

[5] Abhyankar K.V. : Paribhāshāsamgraha, Bhandarkar Research Instt, Puna, (1967).

[6] Paul Kiparsky : On the Architecture of Panini's Grammar,the lectures delivered at the Hyderabad Conference on the Architecture of Grammar, (2002).

[7] Saroja Bhate and Subash Kak : Panini's Grammar and Computer Science, Annals of the Bhandarkar Oriental Research Institute 79–94 (1993).

[8] Roy, P. V. and Haridi, S.,: Concepts, Techniques and Models of Computer Programming, MIT Press, Cambridge (2004).

[9] Vasu, S. C.: The Aṣṭādhyāyī of Pāṇini, Motilal Banarasidas Publishers, New Delhi, (2003).

[10] Vasu, S. C.: Siddhānta kaumudi, Motilal Banarasidas Publishers, New Delhi, (2002).