Towards automating the generation of derivative nouns in Sanskrit by simulating Pāṇini

Thesis submitted to
Indian Institute of Technology Kharagpur
for the partial fulfillment of the requirements
for the award of the degree of

Master of Technology
In
Computer Science & Engineering
by

Amrith Krishna
13CS60R12
M.Tech CSE

Under the guidance of
Prof Pawan Goyal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

Declaration

I, **Amrith Krishna (13CS60R12)** hereby declare that thesis on ``**Towards automating the generation of derivative nouns in Sanskrit by simulating Pāṇini**'' is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. The work was done under the guidance of **Dr. Pawan Goyal**, at Indian institute of Technology, Kharagpur.

_____

Amrith Krishna

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

Certificate

This is to certify that the project entitled ``**Towards automating the generation of derivative nouns in Sanskrit by simulating Pāṇini''** is a bonafide record of the work carried out by Mr. Amrith Krishna (Roll No.13CS60R12) under my supervision and guidance for the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science & Engineering during the academic session 2013-2015 in the Department of Computer Science & Engineering, Indian Institute of Technology, Kharagpur.
The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Prof. Pawan Goyal
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, India 721302

Acknowledgment

I express my sincere gratitude to **Dr. Pawan Goyal**, Assistant Professor, Computer Science & Engineering Department, Indian Institute of Technology, Kharagpur for his supervision and guidance during the project. His valuable advices, encouragement, suggestions and friendly attitude towards me, during this work helped a long way in bringing this project report to this stage. I am also thankful to him for extending all kind of help and for providing the necessary facilities required during this work.

I am very much thankful to **Prof. Rajib Mall**, Head, Department of Computer Science & Engineering Department, IIT Kharagpur for providing necessary facilities during the research work.

I am thankful to Faculty Advisor **Prof. Sudeshna Sarkar** and all the faculty members of the department for their valuable suggestions, which helped me improve on this research work.

I would like to extend my gratitude towards **Dr. Gérard Huet and Dr. Peter Scharf** for their support and for their valuable inputs from the discussions we had related to the work. I would also like to thank the experts **Ms. Sukhada, Ms. Anuja Ajotikar, Ms. Tanuja Ajotikar, Mr. Pavankumar Satuluri and Mr. P Sanjeev** who had participated in the system evaluation we had conducted and shared their valuable comments.

Amrith Krishna

Abstract

About 1115 rules in Aṣṭādhyāyī from *A.4.1.76* to *A.5.4.160* deal with generation of derivative nouns, making it one of the largest topical sections in Aṣṭādhyāyī, called as the Taddhita section owing to the head rule *A.4.1.76*. This section is a systematic arrangement of rules that enumerates various affixes that are used in the derivation under specific semantic relations. We propose a system that automates the process of generation of derivative nouns as per the rules in Aṣṭādhyāyī. The proposed system follows a completely object oriented approach, that models each rule as a class of its own and then groups them as rule groups. The rule groups are decided on the basis of selective grouping of rules by virtue of *anuvṛtti*. The grouping of rules results in an inheritance network of rules which is a directed acyclic graph. Every rule group has a head rule and the head rule notifies all the direct member rules of the group about the environment which contains all the details about data entities, participating in the derivation process. The system implements this mechanism using multilevel inheritance and observer design patterns. The system focuses not only on proper affixation of Taddhita affix for a given semantic sense and other constraints, but also on the correctness of sequence of rules applied to make sure that the derivation has taken place in strict adherence to Aṣṭādhyāyī. The proposed system's design allows to incorporate various conflict resolution methods mentioned in authentic texts and hence the effectiveness of those rules can be validated with the results from the system. Our findings are backed by results obtained by conducting an expert evaluation on the system output. We also present cases where we have checked the applicability of the system with the rules which are not specifically applicable to derivation of derivative nouns, in order to see the effectiveness of the proposed schema as a generic system for modeling Aṣṭādhyāyī.

Keywords: – Sanskrit Computational Linguistics, Pāṇinian Studies, Astadhyayi.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Aṣṭādhyāyī, the central part of Pāṇini's grammar, is a classic and seminal work on descriptive linguistics. Aṣṭādhyāyī provided a complete description of the Sanskrit language spoken at that time period, and is also often praised for the computational principles and programming concepts used in it. Pāṇini's grammar for Sanskrit has Aṣṭādhyāyī as its central component, a collection of about 4000 rules along with other supplementary materials like gaṇapāṭha and dhātupāṭha which are lexical lists, Unadi sutras etc. Some of these materials are adopted from earlier works on Sanskrit grammar that prevailed before his work. Aṣṭādhyāyī literally translates to 8 chapters, and as the translation goes Aṣṭādhyāyī has 8 chapters, each of which is divided into 4 quarters or pāda. Linguists across the world prize the brevity achieved by Aṣṭādhyāyī in its organization of rules, owing largely to tradition of passing information orally that prevailed in India at that point of time. But brevity has not come at the cost of completeness and is often praised for computational insights it provides, which amuses not only linguists but mathematicians and computer scientists across the globe.

Approximately one fourth (about 1115) of rules in Aṣṭādhyāyī deal with generation of derivative nouns (and adjectives), which are derived by affixation from other nouns (or adjectives). The affixes that are used for derivation of derivative nouns are enumerated in the Taddhita section of Aṣṭādhyāyī and hence the affixes that come under this section are called Taddhita. Taddhita section starts with rule *A.4.1.76* तद्धिताः of Aṣṭādhyāyī which is an adhikāra rule and its influence is till the end of fifth chapter i.e. *A.5.4.160* निष्प्रवाणिः च. Though Pāṇini did not provide any semantic definition for Taddhita, it is based on rule *A.5.1.5* तस्मै हितम् which means ``beneficial to that'' (?), where ``that''(तस्मै) is intended for the base

nominal from which the derivation will take place. Aṣṭādhyāyī considers both nouns and adjectives as a single category called as Prātipadika (प्रातिपदिक), and hence the affixes used in Taddhita are not category changing affixes (?).

1.1 Motivation

Aṣṭādhyāyī has received praises from domain experts in the fields of linguistics, mathematics and computer science, for the deep computational insights it carries. Rules of Aṣṭādhyāyī are often compared to a computer program for its rigor and coverage (?). In fact, Hyman has shown that the individual rules of Pāṇini may be reframed such that a finite state transducer can be compiled, owing to the fact that the language that it generates will be a regular language (?). Kiparsky, proved that the Pāṇinian formalism when taken as it is, is at least as powerful as a context sensitive language (?). All these research substantiates the plausibility of automation of Aṣṭādhyāyī. In fact there have been various efforts in the form of formal frameworks, design schemas and implementations from the researchers in computational Sanskrit. Some of those works will be discussed in Chapter 2. But there have been no implementation efforts on automation of Taddhita section of Aṣṭādhyāyī. Generation of derivative nouns from any given valid nominal gives a person the power to expand the vocabulary of the language which helps him/her to express in the most concise way, what he/she has intended. With an automated generator of derivative nouns, the programmer can always keep a finite list of base nouns (both common and proper) along with the finite rules as stated in Aṣṭādhyāyī and then generate new nouns. In fact once new nouns are generated, those can be feeded as inputs to generate newer nouns. By this the user is never limited by a finite set of vocabulary at his disposal, and always new word-forms can be generated.

A derivative noun generator helps us to preserve etymological information of the nouns so generated. Given a generated nominal, we can find the base noun from which it has been derived and also the relation it holds with base noun also can be inferred. This helps to maintain relations between various words within the same language, and will be a valuable addition in the form of supplementary information in Sanskrit dictionaries and other lexical databases like Wordnet or IndoWordnet (?) to be more specific. For e.g.: If someone needs to express a statement ``Son of Upagu", a new nominal can be derived as shown in Figure 1.1. Here the nominal **औपगव** is derived from the base nominal '**उपगु**',and with the base it shares the relation '**अपत्यं**', which is the patronymic relation.

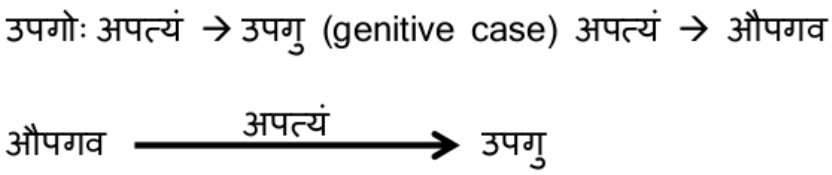

Figure 1.1: Relation between a Taddhita nominal and its base nominal

1.2 Objective

Though Aṣṭādhyāyī was intended for human understanding and usage, this work attempts to automate the process of deriving Taddhitas in complete adherence to Aṣṭādhyāyī. The work aims to generate the correct form of affixation for any given nominal by following the sequence of rules applied as per Aṣṭādhyāyī in the derivation process. In this thesis, we propose an implementation model that helps in organizing the rules of Aṣṭādhyāyī, such that the organization leads to an automated triggering of rules for proper affixation of nouns. For the derivation process, Pāṇini uses a rich set of linguistic features ranging from phonetic features to semantic features. The system tries to keep record of all those features that was acquired at each step of the derivation and stores them as object attributes such that, the feature information is not lost, and hence can be used for further analysis, reusing the for other derivations and linking various relations between different nouns. This approach doubles as a pedagogy tool for the learners and enthusiasts of Sanskrit language and its grammar. Understanding of prakriya or the derivation process is very crucial in understanding of Sanskrit grammar. As the generator has potential of deriving the final form as well as the sequence of rules triggered, a new learner can understand the various transformations that occur in the process for the derivation from base to final form.

In ancient India, teaching has been primarily done orally, where the disciples learn from their teacher through reciting of the sutras or rules. Aṣṭādhyāyi is no exception to it, but due to lack of a well-documented written form, some of its information that was assumed that a learner should possess as a prerequisite has been lost as time passed by from generations to generations. The unavailability of those information especially related to Rule selection, conflict resolution and blocking among rules of Aṣṭādhyāyi, led to academic debates among scholars. There has been different schools of thought, and there are different methods proposed, though there is no single method which has converged to a consensus. The proposed system is open to various conflict resolution techniques that have gained some

ground over the period. We will be applying various conflict resolution methods and more importantly the schema of design allows adding many more conflict resolution techniques as per the the programmer's choice as a pluggable module. Once a conflict resolution method is implemented, then the method's effectiveness may be validated through the results it produces and then its merits and demerits be compared with results so obtained from other conflict resolution methods as well.

The objectives of this thesis can be enumerated into the following three tasks:

I Automation of Taddhita section of Aṣṭādhyāyī, so as to simulate the affixation process of derivative nouns and to evaluate how the rule organization helps to handle affix polysemy, homonymy and synonymy.

II Implementation of various methods for rule selection, conflict resolution and blocking of rules and to check its effects in Taddhita section and in other general cases.

III Implementation of a model for storage of linguistic features that are obtained by the entities participating in derivation and how the same can be used for later derivations and analysis.

Hence the work requires to automate rules in Aṣṭādhyāyī that deal with Taddhita section as well as other associated rules, which help in the derivation process.

1.3 Implementation Model

The proposed system adopts a completely object oriented approach in modeling Aṣṭādhyāyī. The rules of Aṣṭādhyāyī are modeled as classes and so is the environment that contains the entities for derivation. The rules are then grouped based on the notion of topicality by virtue of anuvṛtti. In our proposed system the rule group formation is achieved through formation of inheritance network i.e. multilevel inheritance formed between individual rule classes, which has been inspired from the inheritance network that Pāṇini used in Aṣṭādhyāyī (?). Pāṇini uses anuvṛtti to carry forward the inherited components to child rules, though it needs to be noted that signifying of the inheritance is one of the aspects of anuvṛtti, and hence our proposed model does not form the inheritance network over all the usages of anuvṛtti, but rather a subset of it. The principles discussed in Aṣṭādhyāyī that enable rule selection and rule application like *A.1.4.2* विप्रतिषेधे परं कार्यम्, *siddha* and *asiddha* are built right into the core architecture of proposed system. In addition to these principles, the system provides a functionality to adopt

different conflict resolution methods that are discussed outside of Aṣṭādhyāyī. The system does not restrict itself by adopting any particular conflict resolution method, instead it facilitates to try out various methods that have been mentioned in various scholarly works. This will thus help to evaluate various methods and report on their accuracy as no single set of conflict resolution methods has gained consensus among the scholars.

1.4 Organization of thesis

In Chapter 2, we will be discussing about various attempts towards formalizing rules of Aṣṭādhyāyi, and modeling Aṣṭādhyāyi in part or full to a automated system. In Chapter 3, we will look into the linguistic and structural features of Taddhita section. We will be discussing about various linguistic characteristics that the affixes in the domain possess. We will also be looking into the arrangement of rules in Taddhita section and how the arrangement forms an inheritance hierarchy. In Chapter 4 we will be describing the working of the proposed system, tools and techniques used for the implementation and also modeling of data environment and rule classes. The chapter will also talk about the various conflict resolution methods for rule selection which we have employed and its effect on some of the well known instances in conflict resolution (?), (?). Chapter ?? will show the results of the experts evaluation we have conducted. We have also discussed the analysis of those wrong cases that the experts have observed. Finally, Chapter ?? discusses the applicability of the proposed schema as a general framework to model entire Aṣṭādhyāyi by considering how the proposed system handles rules that are not specific to the derivation of a derivative noun. We also discuss the salient features and application domains of the system along with directions for future research.

Chapter 2

Related Work

2.1 Introduction

Aṣṭādhyāyi has received much attention from computational linguists from the latter half of 20th century. Aṣṭādhyāyi was much lauded for its brevity, completeness and computational insights it provides. There have been works from eminent scholars about the formalism of Pāṇini's Grammar, its expressive power, and the derivation process (prakriyā) it follows. In Section 2.2 we will be discussing about the works on linguistic aspects and computational insights that Aṣṭādhyāyi provides. In Section 2.3 we will be looking into the various attempts in automating Aṣṭādhyāyi. The chapter will be concluded with Section 2.4 discussing about the saliency and the ingenuity of the work described in this thesis how those works referred in this chapter have influenced the line of thought for thesis.

2.2 Linguistic Aspects of Aṣṭādhyāyi

Seminal works from ?) and ?) on formalizing rules of the grammar with stress on the meta-rules that state about the context sensitive aspects was a starting point with further enhancements from ?) where he applies his formalization for more rules that is related to phonetic changes.

?) highlights the relevance of affixation and how it is well integrated with syntax as a continuum in Pāṇini's derivational system. He points out the contrast that Pāṇini's system bears with the system that western grammarians follow, where morphology and syntax are

treated as independent components in derivation. ?) focus on the expressive power of Aṣṭādhyāyi rules and also the expressive power of formalism that Pāṇini used to design Aṣṭādhyāyi. They demonstrate that the formalism of the grammar has far more expressive power than that of regular languages and Context Free Languages. Their work emphasizes on the power of formalism that has built-in capacity for disambiguation at syntactic level.

2.2.1 Taddhita

?) talks about the organization of Taddhita rules specifically and goes into detail about the domain of influence of default affixes and then the sub-domain of semantic senses inside the affix domain. The work also discusses in detail about how the Taddhita section differs from other kind of noun derivations like the kṛdanta section which is, nouns derived from verbs and also from the samāsa which is, compound words formation. ?) in her work coins the constrained separatism approach, which she claims what Pāṇini has followed in Taddhita. She also states idea of networks of rules that helps in the derivation process.

2.3 Automation of Aṣṭādhyāyi

?) developed a finite state transducer after re-framing individual rules in Aṣṭādhyāyi, resulting in generation of strings that belong to regular languages and performs sandhi (सन्धि) at word boundaries for any two given word combinations. Hyman had introduced an XML vocabulary for encoding rules in Aṣṭādhyāyi that helped him in implementing the finite state transducer. ?) discusses about Scharf's and Hyman's combined efforts in developing XML formalization that not only deals with sandhi but also with nominal declensions and verb conjugations.

There have been various attempts to automate Aṣṭādhyāyi in parts as well as modeling it entirely. ?) implemented an inflectional morphology generator that takes as input a noun from the user and then generates all 21 forms of noun declensions, known as *vibhakti* system in Sanskrit grammar. The authors talk about the programming perspectives that need to be considered when encoding rules in Aṣṭādhyāyi, and various computational aspects that Aṣṭādhyāyi possesses. They also talk about the need of conflict resolution methods for competing rules that can be applied in the same context. ?) has developed a system that is an inflectional morphology analyser. They have developed independent systems for verb and noun forms and their corresponding inflections. Though their work was not related to

simulating Aṣṭādhyāyi, but they claim that they take into account the Pāṇinian way of analysis. ?) takes on generation of Sanskrit compounds called as samāsa that deals with about 400 rules of Aṣṭādhyāyi which helps them to form compound words from independent words in Sanskrit. Their work talks about various kinds of semantic features that act as the constraints governing compound formation. The Language Technologies Research Centre (LTRC) of IIIT Hyderabad has developed Dependency parsers for many Indian languages including Hindi based on the Pāṇini's kāraka system. They have used a modified version of kāraka system of Pāṇini to capture the free-word order property that some of the Indo-European languages possess (?).

?) have talked in length about the Computational Structure of the Aṣṭādhyāyi, and introduced the concept of rules that continuously observe the environment or the subject to which modifications are to be made. They have also talked about *Siddha, Assidhavat and Asiddha* principles used in Aṣṭādhyāyi.There has been an in-depth study on *siddha and asiddha* principles by ?), where they talk about the order in which rules need to be applied. ?) mentioned about the grouping of rules based on the general-exception relation between rules and formation of rules as a tree structure, but commented that the feasibility of automation needs to be checked. They also talk about various conflict resolution methods that are mentioned in the sūtras as well as in other vārttikas. ?) presented a computational model based on the principle of *asiddhatva*, an improvised model over the one discussed in ?). ?) talks of the nature of grammar which performs the analysis of constituent elements and then its reconstitution using various set of operational rules as mentioned in Aṣṭādhyāyi. ?) in his work discusses about *vedāṅga* principles and extends his work by considering the common methodological approach of ancillary disciplines for rule application. His work provides a good walk-through for the entire derivation process that begins with introduction of atomic elements to coming up with the desired final form. ?) establishes the issue with phonological over-generation that can occur, if one is to strictly adhere to the rules defined by Pāṇini. ?) sheds some light in formalizing semantic categorization rules when he deals with kāraka systems. These kind of issues have been a matter of debate among linguists for quite long. Many principles that are not stated in Aṣṭādhyāyī but in other texts written at various points of time have surfaced to deal with such issues, mainly those which concern about conflict resolution in rule selection. ?) discusses about various principles like *utsarga-apāvada*, *antaraṅga-bahiraṅga*, *nitya-anitya* etc. in detail.

2.4 Conclusion

To the best of the authors' knowledge, the system which we are going to propose is the first of its kind, that is focused specifically on generation of derivative nouns. The proposed system embraces a unique approach of forming rule groups where similar rules are grouped together to form a Directed Acyclic Graph (DAG). The similarity of rules is based on the notion of topicality present among the rules by virtue of usage of anuvṛtti. The approach can be treated as analogous to the model, what is proposed in ?), but they propose the formation of similar topic DAG through utsarga-apāvada relations. One cannot comment on the similarity of the approaches without a proper comparison of DAGs formed from both them. Moreover ?) mentioned that they had not checked the feasibility of automating their notion of rule group formation. Another important feature that our system uses is that it automatically notifies the relevant rule classes whenever the data environment state changes. This eliminates the overhead of linear searching over each rule, or each rule polling the environment to find its application. Instead the mechanism allows the rules to be updated about environment state changes whenever there is one and yet the rule classes can refrain from state dependency issues with the environment (?).

Chapter 3

Linguistic and Structural aspects of the Taddhita Section

In the Taddhita section, Pāṇini identifies about 300 semantic relations under which Prātipadikas can be generated with Taddhita affixes. It is mentioned as a sub-section to *pratyayādhikāra* that deals with all kinds of affixes. The rules in Taddhita section deal with three entities namely semantic relations, affixes and stems or collection of stems from *gaṇapāṭha*. The rules are defined in such a way so as to facilitate affixation of the proper Taddhita affix with respect to semantic relation intended. The rules often deal with properties of the entities involved at various levels from phonological, morphological, syntactic and semantic levels. There are two types of derivations possible that involve Taddhita affixes (?).

Prātipadika + Taddhita-affix

Prātipadika + Taddhita-affix + strī-affix

3.1 Linguistic Phenomena in Derivational Morphology

Derivational morphology in Sanskrit, like in many other languages poses some of the well known facts about many to many correspondences between forms and affixes. Taddhita affixes exhibit affix polysemy, homonymy, synonymy and non-compositionality. Considering a single affix and multiple senses we can discuss about affix polysemy and homonymy and with multiple affix and same semantic relation we can talk about affix synonymy. In fact, in Sanskrit non-compositionality of the taddhita affixes are also well discussed . In affix polysemy, the same affix is used to denote related senses like in the case of patronymic and

provenance relation. In affix homonymy the same affix is used in distinct and unrelated semantic contexts like in the case of personal nouns or abstract nouns. Affix synonymy deals with the same semantic sense but uses different affixes. For example, for the patronymic relation **अपत्यम्**, multiple affixes can be used, like **अ(ण्)**, **अयन(फक्)**, **इ(ञ्)** depending on the stems used (?). Table 3.2, Table 3.1, Table 3.3 show some instances of each of these phenomena, with analogy to the same phenomena in English language.

Table 3.1: Affix Polysemy in Sanskrit and English

Affix Polysemy			
Sanskrit (affix - **अ**)		English (affix -ian)	
Base	Derived	Base	Derived
मथुर	**माथुरा**	India	Indian
उपगु	**औपगव**	Orwell	Orwellian

Table 3.2: Affix Homonymy is Sanskrit and English

Affix Homonymy			
Sanskrit (affix - **अ**)		English (affix -ian)	
Base	Derived	Base	Derived
निपुण	**नैपुण**	Library	Librarian
शुक	**शौका**	Comedy	Orwellian

Table 3.3: Affix Synonymy in Sanskrit and English

Affix Synonymy					
Sanskrit (Patronymic)			English (Provenance)		
Base	Derived	Affix	Base	Derived	Affix
उपगु	**औपगव**	**अ**	India	Indian	-ian
अश्वल	**अश्वलायन**	**अयन**	Kerala	Keralite	-ite

3.2 Organization of Taddhita rules

Taddhita section is primarily subdivided into five *pratyayādhikāras* or domain of control of five affixes.

- The five rules are:

  I *A.4.1.83* **प्राक् दीव्यतः अण्** - **अण्** suffix is the default affix to be used for affixation for all the rules till *A.4.4.1.* The influence of *A.4.1.83* is till the term **दीव्यति** is found (or till another pratyayādhikāra is found) and **दीव्यति** appears in rule *A.4.4.2* **तेन दीव्यति खनति जयति जितम्.**

  II *A.4.4.1* **प्राक् वहतेः ठक् ।** - **ठक्** suffix is the default affix to be used for affixation for all the rules till *A.4.4.76.* **वहति** appears in rule *A.4.4.76* **तत् वहति रथयुगप्रासङ्गम्.**

  III *4.4.75* **प्राक् हितात् यत्** - **यत्** is the default affix to be used for affixation for all the rules till *A.5.1.5.***हितम्** appears in rule *5.1.5* **तस्मै हितम्.**

  IV *5.1.1* **प्राक् क्रीतात् छः** - **छः** is the default affix to be used for affixation for all the rules till *A.5.1.37* **क्रीतम्** appears in rule *5.1.37* **तेन क्रीतम्.**

  V *5.1.18* **प्राग् वतेः ठञ्** - **ठञ्** is the default affix to be used for affixation for all the rules till *A.5.1.115.* **वतिः** appears in rule *5.1.115* **तेन तुल्यं क्रिया चेत् वतिः** .

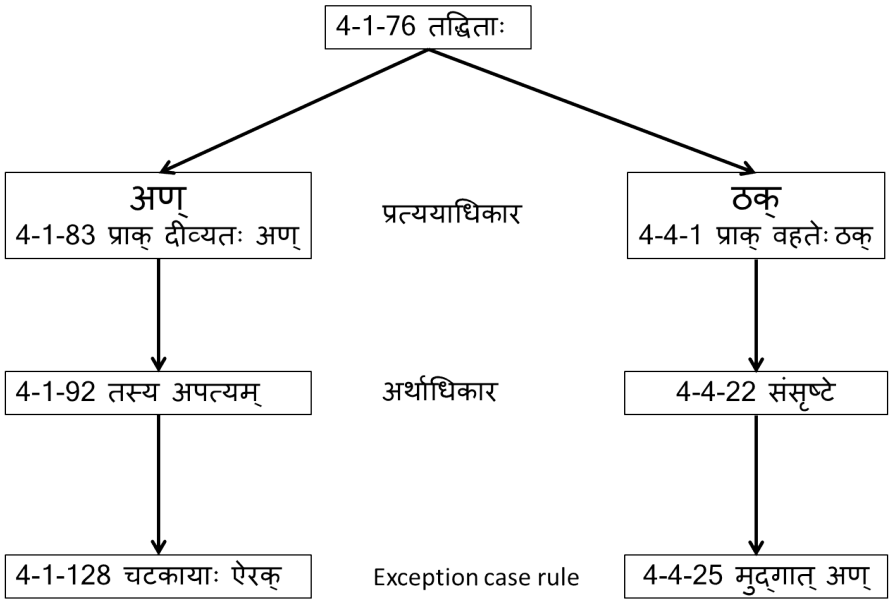

Figure 3.1: An instance of inheritance hierarchy in Taddhita section

Now each of the above given rules, which can be called as the pratyayādhikāra rules, specifies what is the most prominent affix or the default case affix that can be applied to all the rules which are under its domain. For the set of rules under a single pratyayādhikāra, they are further categorized on the basis of arthādhikāra rules. Each arthādhikāra heads a set of rules which are a proper subset of rules that come under the pratyayādhikāra. The arthādhikāra rules state the semantic conditions or the semantic rules under which the default affix can be attached. Arthādhikāra rules serve the purpose of a topic head that groups a set of rules as a rule group based on a topic, which is a semantic relation here and it also acts as an operational rule (विधि), as it carries the semantic sense under which the affix can be attached. Rules *A.4.1.92* तस्य अपत्यम्, *A.4.2.1* तेन रक्तं रागात्, *A.4.2.69* तस्य निवासः etc. denote the semantic senses such as patronymic, coloured by means of that and provenance respectively. This kind of arrangement handles affix polysemy and affix homonymy. But to handle affix synonymy, there are other operational rules, that come under the domain of arthādhikāra rules. They can be seen as exception rules or rules to handle special cases. These rules limit the application of default affix specified in pratyayādhikāra and mention what other affixes can be used instead, under special conditions. In this way, affix synonymy and non-compositionality is taken care of.

?) calls this form of arrangement as constrained seperationism, where the rules form a multilevel single inheritance network. The template for the multilevel inheritance network will be of the type, ``Default Affix rule'' -``Semantic Sense rule'' - ``Special case rules''. Figure 3.1 shows an instance of how the hierarchy in Taddhita works. As already discussed, *A.4.1.83* states about the default affix अण् for rules under its domain, *A.4.1.92* states about the patronymic relation under which a nominal will get the default affix. Now rule *A.4.1.128* states about a special case when the stem चटका is used with the patronymic relation. In such a case the suffix ऐरक् needs to be attached. This rule overrides the default case rule. Now consider rule *A.4.4.25.* This rule talks about usage of अण् as an exception or special case when a stem मुद्र is used. The rule has *A.4.4.1* as its default affix rule and the default affix for the domain is ठक्. The rule's semantic sense is संसृष्टे, which means `properly mixed with'. Here the rule *A.4.4.25* accounts for the non-compositionality of the affix अण् and is not specified in its domain but as an exception rule in another default affix domain. In this way Pāṇini is able to achieve constrained separationism (?) which is an alternative to both the separationist and non-separationist viewpoints in the case of derivational morphology. Here Pāṇini has employed a multilevel but single inheritance hierarchy through pratyayadhikāra and arhtadhikāra, where arthadhikāra inherits from only one pratyaya, and affix synonymy that may occur is being handled through exception handling rules.

It is to be noted that there are some more rules in the Taddhita section which do not come under any of the five pratyayadhikāra rules. Those rules have arthādhikāras of their own but has no default affix to be attached. This group of rules is treated as extraneous to otherwise systematic network of the section (?). As we will be discussing about the implementation schema of proposed generation system in the subsequent section, it will become evident that this anomaly in no way is going to affect the system.

3.3 Techniques of rule arrangement in Taddhita and Aṣṭādhyāyi

Aṣṭādhyāyi is a linear arrangement rules across eight chapters as mentioned. Aṣṭādhyāyi was supposed to be passed on in oral form. So Pāṇini had to take care of two aspects. He needed to somehow define thematic domain and the boundary of the domain for the rules, and also he needs to achieve brevity as he needed to keep the entire Aṣṭādhyāyi as short as possible for ease of recitation. He has used special kind of rules called adhikāra rules for specifying thematic domain and further sub-domains for a given set of rules. He has efficiently used the concept of ellipsis in the form of anuvṛtti to achieve brevity

3.3.1 anuvṛtti

1-3-2 उपदेशेअच् अनुनासिकः इत्
1-3-3 हल् अन्त्यम्
1-3-4 न विभक्तौ तुस्माः
1-3-5 आदिः ञिटुडवः
1-3-6 षः प्रत्ययस्य आदिः
1-3-7 चुटू
1-3-8 लशकु अतद्धिते

Figure 3.2: anuvṛtti of ‘it’ section in Aṣṭādhyāyī

In anuvṛtti or what we can call as recurrence, in order to keep the individual sutras short, some information is omitted from those sūtras which can be obtained from subsequent rules. So a rule when read alone doesn't provide the complete information and hence needs to be read with the entire anuvṛtti. For eg the rule number *A.4.1.123* is **शुभ्रादिभ्यः च** which means ``for stems in śubra and co. as well", and does not makes any sense if not provided with any other information. But the same rule when read with anuvṛtti will be **शुभ्रादिभ्यः च प्रत्ययः परः च आद्युदात्तः च तद्धिताः समर्थानां प्रथमात् वा प्राक् दीव्यतः अण् स्त्रीपुंसाभ्यां नञ्स्नञौ भवनात् तस्य अपत्यम् ढक्** . Here the rule translates to ``for stems in śubhra and co. as well, that come under adhikāra of pratyaya, under the sub domain of taddhita, and pratyayādhikāra of aṇ where the semantic relation is apatyam and if the stem is in genitive, apply the pratyaya ḍhak. " anuvṛtti may be passing down of entire rule as in *A.4.1.82* or a partial component inheritance as in case of *A.4.1.120*. The components being passed down as anuvṛtti can be a semantic relation, a condition to be satisified, a context in which it needs to be applied, an adhikāra or a particular operation. anuvṛtti or recurrence can be seen as block of control as well. To visualize consider the snippet given below in Figure 3.2. Here anuvṛtti is denoted via indentation and those which are inherited is made bigger in size.

3.3.2 Adhikāra

It is a form of inheritance where classes of rules belonging to same thematic domain inherit thematic head so as to be grouped together (?). The pratyayadhikāra *A.4.1.83* is one such adhikāra rule. The rule infact is a sub-domain inside *A.4.1.76*, which is a subdomain of *A.3.3.1*. Similarly rule *A.4.4.1* share common ancestor adhikāra rules as of that for *A.4.1.83*. Hence we claim that the Aṣṭādhyāyī is a multilevel inheritance hierarchy. The device of adhikāra indicates the homogeneity of topic (?). It classifies the rules that come in its domain with rules that come in a different domain. A rule can never be under the influence of two adhikāra rules where one adhikāra is not a subset of the other in terms of domain reach. A rule can have multiple adhikāra as shown in that for *A.4.1.92*, it has adhikāra rules starting from *A.4.1.76*. But no two adhikāra rules are at equal level, each adhikāra rule is a subset of other rule. Hence there can be multilevel inheritance but no multiple inheritance in Aṣṭādhyāyī.

Chapter 4

System Implementation and the Derivation Process

4.1 Overview of the Implementation System

For automating the derivation process, we suggest the following method which is based on object oriented concepts. The derivation process happens in a central object called as environment, which essentially has methods and attributes that takes care of conversion of user input to suitable data structures, triggering of rules and book-keeping of rules applied. Each rule forms a class and each instance of the rule class (henceforth to be referenced as rule itself) is registered with the environment, such that whenever there is a change in the environment, the rules are notified. Each rule checks for the possibility of it being applied on the environment and for a rule, if all its conditions are satisfied, ideally the rule can be applied on the environment. However, in a general scenario multiple rules may claim their competency for application on the environment. To handle such scenarios we keep those competing rules in a list called candidate list. Then a conflict resolution method is employed which decides the winner rule. The winner rule gets to apply on the environment and other rules are removed from list. By removal of rules other than the winner rule, we mean that the removed ones are not applied on the present instance of environment, although they are notified when the environment change happens again. Figure 4.1 shows the schema of the implementation system.

As it is evident from discussion about Taddhita section, Pāṇini among various applications of anuvṛtti, uses it for carrying the topicality between different rules as well. It is to be noted

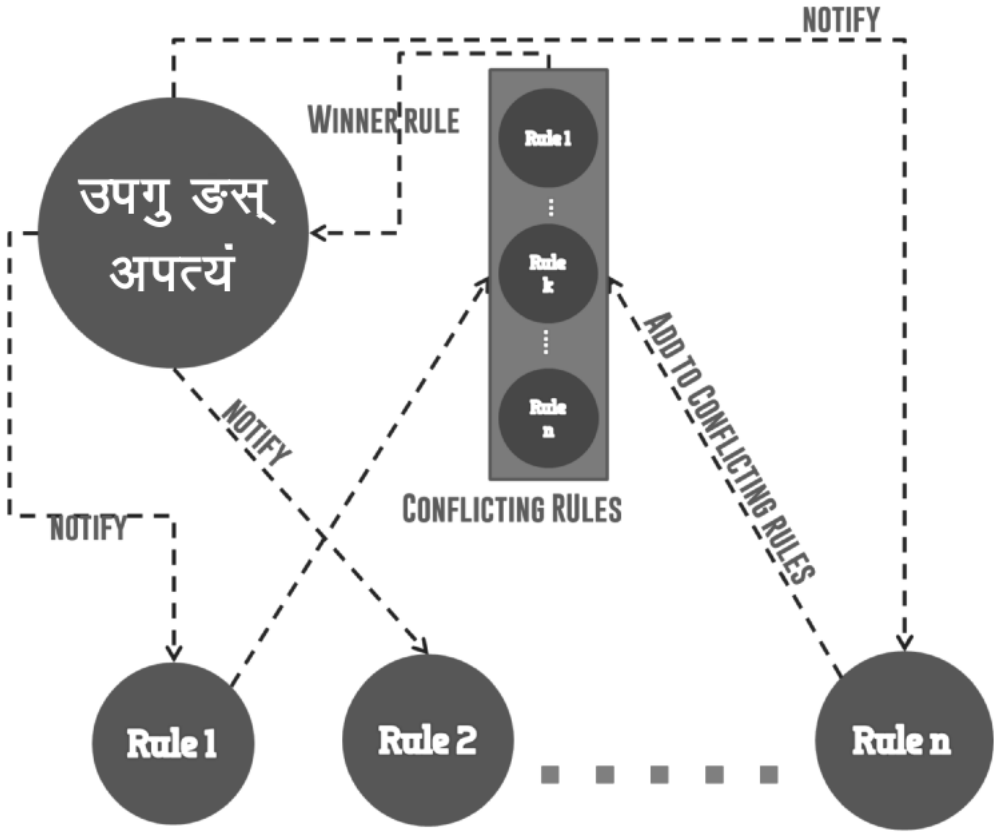

Figure 4.1: Overview of Implementation System

that adhikāra rules are rules whose sole purpose is mentioning the topicality of the rules under its domain of influence. But even for adhikāra rules, anuvṛtti itself is used for carrying the domain's influence to other rules. Apart from the adhikāra rules, there are other rules as well in which anuvṛtti is used to carry the topicality. For example consider the rules concerning **इत्**, i.e. rules from 1.3.2 to 1.3.8. Here **उपदेशे** and **इत्** are being carried forward to other rules as well. **उपदेशे**, which means `when an upadeśa is encountered', perform some action. This condition leads to a common topicality for all the rules under the anuvṛtti. This is similar to the notion of arthādhikāra rules in Taddhita where the notion of semantic sense is being carried forward to subsequent rules through anuvṛtti. Both the techniques, anuvṛtti and adhikāra are employed in the entire Aṣṭādhyāyī and are not unique to the Taddhita section. We can infer that a subset of the anuvṛtti rules, mostly the ones which carry the notion of topicality can be used to design an inheritance hierarchy of classes. For the implementation

we will be using this notion of topicality via anuvṛtti, in grouping of rules to form rule groups which is an inheritance hierarchy among rules and the rule in which the portion of anuvṛtti appears, becomes the head rule.

4.1.1 Tools and Techniques Used

We are following a completely object oriented approach for the implementation of the system. The following tools and techniques will be employed to achieve the principles discussed in section 4.1

Observer Design Patern

Observer design pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically (?). The object to which the state changes occur is called subject and it maintains a list of its dependents, called observers. As shown in Figure 4.2, each observer object which is the sūtra or rule in our case, is registered to an object called as "Subject" class which represents the environment in our case. Subject class has a method to register the observers. Whenever a change in value of some attribute of subject occurs, it calls the method `notify' of observer abstract class, which is implemented for each object of the Observer.

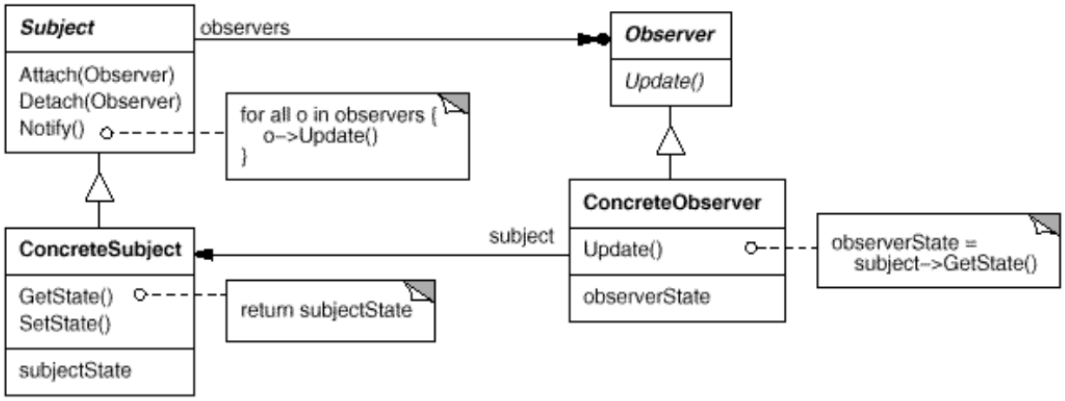

Figure 4.2: UML diagram for observer design pattern.(?)

Multilevel Inheritance

To form rule groups i.e. inheritance network among rules we use multilevel inheritance between the rules. Though multilevel inheritance is allowed, multiple inheritance is not allowed in the system and hence no single rule will inherit from two distinct rules directly. Figure 4.3 shows the inheritance achieved for each class in Aṣṭādhyāyī. All classes inherit from a base class called ``sūtra'', which is a generic class that defines all possible features that a rule or sūtra can possess. All other rules inherit from it. The adhikāra rules, anuvṛtti of interpretive rules, default condition rules etc. form super class for other operational rule classes.

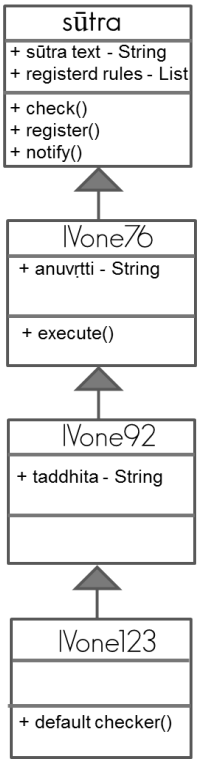

Figure 4.3: UML diagram for multilevel inheritance

4.1.2 Rule Triggering and Propagation

The environment forms the subject class for the observer design pattern. However, not all rules are observing the environment but only those rules that do not come under the domain of any adhikāra rules or a controlling head of a rule group through anuvṛtti (which are mostly

rules that assigns the technical terms to assignments). Those are represented by classes R1, R2 and so on in Figure 4.4. Then there are rule groups as represented by RG1, RG2 and so on. These are collections of rules with same head. The rule classes that come under the same head in the group, inherit from the head rule class. The head rule object, as per observer design pattern registers the inherited classes' objects as observers. Now the head class notifies the observers whenever the environment satisfies the head rule's conditions. The conditions checked are those conditions that need to be satisfied by all the rules registered under the head. Here, by `head', we mean either an adhikāra or a component passed on by anuvṛtti. Now top level rules are those rules which observe the environment directly and get notified whenever the environment changes. For a rule, if the environment satisfies its conditions, it will be added to the candidate list. But if the environment satisfies the conditions that is applicable to an entire rule group, then the environment object is passed on to the next level and this continues till an exception or specific rule is encountered or else returns back to where the default rule resides. By this model, we can employ conflict resolution at each level, and resolve some of them locally and only rules that have no common head at any level come to the candidate list at top-most level, from where the winner rule will be selected.

Once a winner rule is selected, the rule's intended action is executed first and then its parent object is called which performs its relevant portion in execution, if any. This continues until all the rules in hierarchy are called. It is to be noted that many a times certain rules like the adhikāra rules do not have anything to execute of their own, in such cases the rule object just passes on the environment to its parent object. By this design, redundancy of hard coding the same rules again and again per rule object is saved, just like the way anuvṛtti helps a person to avoid repeating the rules when reciting Aṣṭādhyāyī.

Working

Let us consider the working of the system with reference to affixation for semantic relation **अपत्यम्**, i.e. the patronymic relation. Figure 4.5 shows the trace for the affixation process of Taddhita affix when the semantic relation is patronymic and the stem is **चटका**. In Figure 4.5, the solid lines show inheritance hierarchy between the rule classes. The dashed line on the left of each block shows traversing of the hierarchy by triggering of rules from head to specific rules. The traversal checks for eligible rule to apply on environment within the rule group, and this can be called as the checking phase. The dotted lines on the right of each block show the trace of rules that get executed. The starting node, i.e. the node that heads the dotted edge

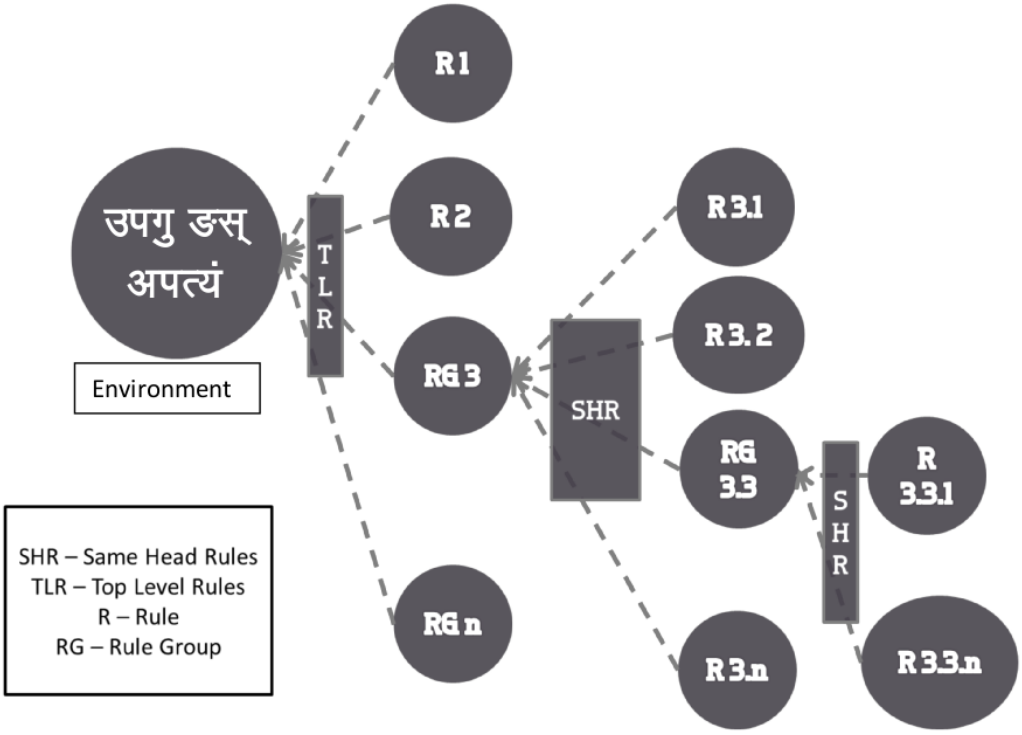

Figure 4.4: Triggering schema.

with label 1 is the exact rule that gets executed and other nodes in the path show those rules that have been executed due to anuvrtti. This can be called as the execution phase. As an example let us consider the case where the environment is initialized with चटकायाः अपत्यम्. The triggering starts from head rule at the top level i.e. from *A.3.1.1* to rule *A.4.1.128* as shown in Figure 4.2. Here in rule *A.3.1.1*, it does not have any condition to check, so it directly notifies all of its direct descendants. Now among the direct descendants, rule *A.4.1.1* checks for presence of any of the two affixes ङि, आप् or if the environment has a prātipadika in it, i.e. it checks for conditions mentioned directly in the rule. As it is a prātipadika, the condition will evaluate to `true',and all its descendants are notified. In due course, *A.4.1.76*, *A.4.1.83* are also notified. These rules as well do not have any extra checks as they are adhikāra rules and hence all its direct descendants are notified. When *A.4.1.92* is notified, it checks for semantic condition and the checking turns out to be true for *A.4.1.92*, while the checking will evaluate to `false' for all of its sister nodes, i.e. other direct descendants of *A.4.1.83*. The special case rules registered under *A.4.1.92* are notified, of which *A.4.1.128* satisfies the remaining conditions. As it does not have any rules registered to it, it becomes the terminal node and hence it is added to the

candidate list. Since for this case no other rule is contesting, the rule emerges winner and starts its execution from *A.4.1.128*. The rule adds the affix ऐरक् to environment, and passes the environment to its parent class i.e. *A.4.1.92*. *A.4.1.92* does not have anything to execute of its own, so it simply passes environment to *A.4.1.83*, which checks if any affix is already attached, as the affix `ऐरक्' is attached in this case, no action is performed. The environment gets passed to parent node of each rule finally this terminates at top level rule *A.3.1.1*. Now consider the derivation of उपगोः अपत्यं. As with the case of चटकायाः अपत्यं, the path of the rules checked for eligibility of rule application remains the same till rule *A.4.1.92* is reached. Once *A.4.1.92* is reached it will notify all of its descendants as well. But since no rule will find its application, *A.4.1.92* will become the final node of the path here. It is added to candidate list, which while executing will simply call the parent rule *A.4.1.83*. *A.4.1.83* checks for presence of any new Taddhita affix in the environment. If the check evaluates to false, i.e. if no new Taddhita affix is found, *A.4.1.83* treats this as default case scenario and attaches अण्, the default case affix to environment. After each execution of a rule, the system checks for presence of any new assignment of `technical term' or else all rules in the top level are notified as discussed in Section ??.

The arthādhikāra rules inherit from its corresponding pratyayādhikāra rules, apart from the ones already mentioned in Section 3.2. When it comes to affixation, during the checking for eligibility of a rule to apply i.e at the checking phase, we need to traverse the pratyayādhikāra rule, before an arthādhikāra rule is reached. Though a pratyayādhikāra rule is visited during checking, no action is taken there. The pratyayādhikāra rule simply passes the environment to all arthādhikāra rules which are its direct descendants. In fact for a single affixation, all the pratyayādhikāra rules get notified from its parent rule, and those rules in turn notify all their direct descendants as there is not enough information to select a single pratyayādhikāra at during the checking phase. So in effect, the process of affixation for taddhita starts by checking for the right semantic condition i.e at the arthādhikāra rules as it is in the case of traditional system of derivation. Before that, the other rules either simply pass on the environment to their descendants or check for conditions that is necessary for the process to qualify as a case for affixation under taddhita. The effect of pratyayādhikāra rule comes during the execution phase i.e at the applying of affix phase and not on the checking phase. During the execution phase the pratyayādhikāra which is parent to the winner arthādhikāra rule acts as the final gate that makes sure that the environment has the valid taddhita affix added to the environment before it reaches its parent. The rule checks if any taddhita affix is introduced by virtue of special case rules, and if no such execution has taken place, then the default affix is added to the environment. The environment is then passed on to higher level rules that takes care of other

generic aspects about the environment.

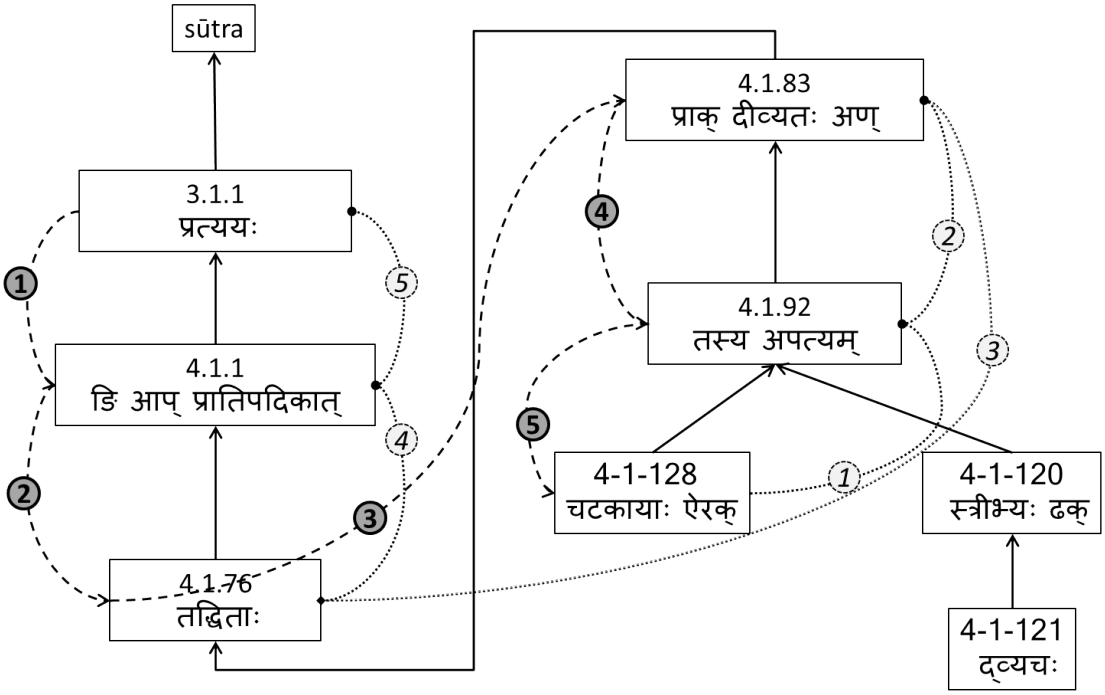

Figure 4.5: Affixation under patronymic relation for **चटका**

4.1.3 Śabdarūpa Representation

The environment is an object which contains methods for navigating through top level rules, list of applied rules and the śabdarūpa object as its attribute. Śabdarūpa object holds the data entities which take part in the derivation process. Entities can be stems, affixes, augments, characters or any of their properties. The most basic and atomic entity in the śabdarūpa object is another object called `śabda'. The śabdarūpa object also contains various instances of the class `śabda collection' , which contains a sequence of references to śabda objects along with a set of attributes that belongs to the collection. Śabdarūpa object is an instance of `śabda collection' class. Instances of `śabda collection' are used to represent various technical terms that may be attributed to environment or a part of it. Figure ?? shows how the śabdarūpa object is set up. Figure ?? shows the śabdarūpa object after the affixation of Taddhita affix **अण्** . As already seen in Section 4.1.2, the Taddhita affixation happened due to presence of technical term prātipadika. Now the technical term prātipadika is modeled as an attribute of the śabdarūpa object, and it is an object itself of the class `śabda collection'. It contains