FROM VERB & KARAKA SPECIFICATIONS TO SIMPLE SANSKRIT SENTENCES : An NLP Approach

Dissertation submitted to

Jawaharlal Nehru University

in partial fulfilment of the requirements

for the award of the degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE & TECHNOLOGY

by

Roshan Lal

SCHOOL OF COMPUTER & SYSTEMS SCIENCES

JAWAHARLAL NEHRU UNIVERSITY

NEW DELHI - 110067

INDIA

JANUARY, 1994

To

My Parents

CERTIFICATE

This is to certify that the dissertation titled **From Verb & Karaka Specifications to Simple Sanskrit Sentences: An NLP Approach** being submitted by me to Jawaharlal Nehru University, New Delhi, in partial requirements for the award of the degree of Master of Technology, is a record of the original work done by me under the supervision of **Prof. G.V. Singh, Professor, School of Computer and Systems Sciences,** Jawaharlal Nehru University, New Delhi, during Monsoon Semester, 1993.

The results reported in this dissertation have not been submitted in part or in full to any other university or institution for the award of any degree or diploma.

Roshan Lal 5/1/94

6-1-94

Prof. K. K. Bharadwaj
Dean,
School of Computer &
Systems Sciences,
Jawaharlal Nehru University,
New Delhi.

Prof. G. V. Singh
School of Computer &
Systems Sciences,
Jawaharlal Nehru University
New Delhi.

CONTENTS

CHAPTER 4

CHAPTER 5

ACKNOWLEDGEMENT

It is my pleasure to express my deep sense of gratitude to Professor G. V. Singh, SC&SS, JNU, for his gracious and valuable guidance. Without his guidance this work simply would not have happened. I feel privileged that I worked under his supervision.

My warmest thanks also to Professor K. K. Bharadwaj, Dean, SC&SS, JNU, for providing the excellent academic environment and facilities which enabled the successful completion of my project.

My sincerest thanks to my friend D.K. Lobiyal. His advice were valuable, his helpful attitude constant. Amit revealed mystries and tricks of Prolog. My heartful thanks to him.

Reeta Nath, Susmita, Ajay, Archana, Bhushan, Sumitra and other classmates were ever so friendly and ready to cooperate. My thanks to them. I thank my very own Sanjay for his friendliness, cooperation and being such a good company in those long hours in the Lab.

My thanks are due to Robin serving a good quality tea during those tiring hours which I spent in lab to accomplish this work.

I express my thanks to Department of Electronics for funding the machine on which the present work was carried out.

My special thanks to **Dr. K. Suryanarayan.** He explained me the intricacies of Paninian Grammar. Throughout the preparation of this work, I availed his expertise and valuable suggestions.

New Delhi
January, 1994.

ROSHAN LAL

CHAPTER I

NATURAL LANGUAGE PROCESSING

1.1 WHAT IS NLP?

If we look at the world as a whole in a broad perspective, the world may be thought of as made up of two entities namely **OBJECTS** and **ACTIONS**. Therefore, in our communication, we express objects and actions performed on these objects. Our communication, generally, takes place in the form of Natural Language (sentences). Therefore, there should be a mapping between these world constructs (objects and actions) and our language constructs.

In NL the basic unit of communication is a sentence which mainly consists of Noun Phrases (NPs) and Verb Phrases (VPs). The objects are denoted by NPs and the actions by VPs. To use computers for the processing of NL, we have to represent objects and actions in a way and form which is understandable to the computer. Conversely, the objects and action constructs represented in computer have to be mapped to NL constructs to generate NL units. The mapping of NL constructs to world constructs and the representation of the same for computers's understanding is called **Natural Language Understanding** and the related process of generating NL constructs from the representation of the intended idea, inside computer, is called **Natural Language Generation**. These two parts constitute what is now commonly called **Natural Language Processing**.

1.2 NATURAL LANGUAGE UNDERSTANDING

Computers require a great deal of precision, completeness, exactness in communication.

They don't have the linguistic flexibility exhibited by humans. We can and do express the same objects, actions or idea in sentences which are drastically different in their form. If we are thirsty we may say to a waiter (i) A glass of water please (ii) Give me some water (iii) May I have a glass of water ? To all sentences the waiter shall respond by giving us water. To be useful for NLP, the computer should respond in the same way (as waiter) to our intended idea, however we paraphrase our intention. And precisely this the goal of natural language understanding research.

We say a program *understands* a natural language if it behaves by taking a (predictably) correct or acceptable action in response to the input. For example, we say a child demonstrates understanding if it responds with the correct answer to a question. The action taken need not be an external response. It may simply be the creation of some internal data structures as would occur in learning some new facts. But in any case, the structures created should be meaningful and correctly interact with the world model representation held by the program.

1.3 NATURAL LANGUAGE GENERATION

It is sometimes claimed that language generation is the exact inverse of language understanding. While it is true the two processes have many differences, it is an oversimplification to claim that they are exact opposites [6].

The generation of natural language is as difficult as understanding it, inasmuch as a system must not only decide what to say, but how the utterance should be stated. A generation system must decide which form is better (active or passive), which words and structures best express the intent, and when to say what. To produce expressions that are natural and close to

humans require more than rules of syntax, semantics, and discourse. In general, it requires that a coherent plan be developed to carry out multiple goals. A great deal of sophistication goes into the simplest types of utterance when they are intended to convey different shades of meanings and emotions. A participant in dialogue must reason about a hearer's understanding and his or her knowledge and goals. During the dialogue, the system maintain proper focus and formulate expressions that either query, explain, direct, lead or just follow the conversation as appropriate.

1.4 ISSUES INVOLVED NL GENERATION

Generation systems accept a representation of some set of linguistic goals and have to produce a set of natural language sentences that realize those goals. The problem can be divided into two main areas:

1. selecting from the knowledge representation the particular semantic content of the sentences
2. transforming the semantic content into actual natural language sentences

The area of planning and plan-based representations of linguistic goals is relevant to the first problem, and the mapping between the final representation, the logical form, and the syntactic structure are relevant to the second[9].

Consider a situation where an agent Ram sold a car to another agent Shyam. The information available from a representation would, probably, include all the properties of Ram, all the properties of Shyam, whatever is known about the car Ram is selling, plus information about selling events in general, and this selling event in particular. How is a generation system

is to choose the information that is relevant and should be included in a description of what happened? The decision must be based in part of what the system thinks is necessary to identify the key objects involved (for example, the hearer may not know Shyam by name, so the system might describe Shyam in terms of his relationship to Ram, say as Ram's Boss), and what aspects of the situation are important to convey (for example, possibly Ram asked a price that is much more than the car is worth). This question, of course, cannot be decided independently of some description of what linguistic goals the speaker has (for example, may be the speaker wants to convince the hearer that Ram is a swindler, or maybe the speaker wants to show that Ram didn't like his car, and so on).

Once the content has been decided, it must be realized in a set of sentences. This requires significant knowledge about how different linguistic structures have different linguistic effects. Even when the content can be realized in a single sentence, the situation is very complex. For example the following sentences say nearly the same thing, and most understanding systems would abstract away the differences between them during syntactic and semantic processing.

1. The man in the top hat dropped the bamboo cane.
2. The man wearing the top hat dropped the bamboo cane.
3. The bamboo cane was dropped by the man who was wearing the top hat.
4. The man who dropped the bamboo cane was wearing a top hat.

All of these sentences have approximately the same meaning, and if one is true, then they all are true. They each have slightly different effects, however. Sentences 1 and 4, for example, use different information to identify the man (the man in the top hat versus the man who dropped

the bamboo cane) and thus would have different effects when processed by referential analysis. So generation program would have to know which one of these is the appropriate description. Thus a system should have sufficient information to determine the exact structural description of the given situation to give rise to the required effect, voice etc.

1.5 PROBLEM DESCRIPTION

Sentence generation and analysis are two main issues in Natural Language Processing. The generation of a sentence starts from the meaning the speaker intends to convey. Semantically, the verb(i.e. dhatu) may be considered as the center of a sentence, and other words are related to it either directly or indirectly. These relations are specified by karaka. Based on the meaning of the verb the possible karakas that go with it can be known. If these values are stored with the verb, the rest of generation process can easily taken care of.

Sanskrit lacks a strict syntax, and hence the order of the words cannot be easily predicted. The constituent words can occur in any manner, whatsoever. For example, the verb may occur as the first word, in certain constructions. Questions relating to concord, government and order need to be considered for sentence generation.

In the present dissertation, it is proposed to generate **Simple Sanskrit Sentences using Verb and Karaka specifications**. The target sentences shall be consisting of one *tiganta pada* (finite verbal form) and one or more *subanta padas*. The generation of a sentence is supported by two lexicons, one for noun-forms and other for verb-forms. The input to the system is *dhatu* and *pratipadikas*. The form of the dhatu to be generated depends on *lakara*, *purusha* and *vachan*. Similarly, the relation of the noun/s to the verb is governed by the case-markers, which in turn

come from *karaka* specifications.

This work is augmented and complemented by the related work, namely, 'Sentence analysis based on verb and karaka specifications'.

The present work does not consider adjectives, adverbs, participles, passive and complex structures. The sentences generated contain minimum of two words and a maximum of seven words.

CHAPTER II

PANINIAN GRAMMATICAL FRAMEWORK

Language is the most effective means of communication. It consists of innumerable sentences. The grammar of a language is a formal specification of allowable structures of that language. It provides a finite set of rules to describe infinite number of sentences.

Language processing through computers has given a new impetus for the study of various aspects of language, particularly, grammar. In recent times, traditional grammars have given way to computational grammars. The Paninian grammar, written some two thousand years back, is greatly appreciated for its computational approach in describing Sanskrit language. It is discussed in greater detail below :

2.1 PANINIAN GRAMMAR :

The Astadhyayi, written by Panini[7], is a grammar of the Sanskrit language as it was spoken by the Indo-Aryan tribe living on the Gangetic plain around the sixth centuary B.C.

The Astadhyayi is a grammar in the precise mathematical sense of the term as it provides a set of rules whereby well- formed Sanskrit sentences can be generated.

It is a tribute to the Astadhyayi that, despite the rapid development of various linguistic theories in the modern times, it has still remained a work of great linguistic interest. According to a renowned American linguist,' it is monument of human intelligence'.

The Paninian grammar is a descriptive grammar for Sanskrit language. It is a generative

grammar with computational approach in handling the linguistic data. In recent times, it has attracted the attention of computer scientists working in NLP.

Sanskrit being an inflectional language with a relatively free word order, lays emphasis on morphology i.e. the word formation, the Paninian grammar essentially generates words, of different types.

The major word formation types that the Paninian grammar generates are as follows :

1. *Subanta* formation (nominal inflection),

2. *Tiganta* formation (verbal conjugation),

3. *Kridanta* formation (primary derivatives),

4. *Tadhhita* formation (secondary derivatives),

5. *Samasa* formation (compound formation) and

6. *Stripratyant* formation (feminine base formation).

The aforesaid word-forms can be classified either as inflectional or derivational. *Subantas* and *tigantas* come under the first category, while the others fall under the second category. The inflectional morphology conceives words as role players in a sentence while the derivational morphology involves the formation of various lexical items.

The *subantas* involve declension of nominal bases, and deal with case markings. The *tigantas* deal with the conjugation of verb roots in different tenses and moods. The *kridantas* involve the derivation of a nominal base from a verb root. Apart from the nouns, the *krit* suffixes also generate participles, adjectives and so on. The *taddhitas* deal with the derivation of secondary noun forms. *Samasas* is a word-compounding process in which simple words whether nouns, adjectives, numerals, verbs or indeclinables are combined with one another to form

compounds. Feminine basis in Sanskrit are formed by adding *stripratyayas* to the nominal bases.

For the purposes of the present work, only *subanta* and *tiganta* are considered.

There is no sharp distinction between morphology and syntax in Sanskrit grammar. As mentioned earlier, the output of nominal and verbal morphology give us the syntactic units.

2.2 SANSKRIT SYNTAX :

Sentence is the minimal meaningful unit of a language. The division of a sentence into words and , of a word into base and suffixes is merely for the sake of grammatical analysis.

In Sanskrit , the verb plays a key role and is the semantic focus of a sentence. Thus, a finite verbal form in Sanskrit contains information regarding tense/mood, number, person, voice and so on. However, Sanskrit also allows purely nominal sentences.

The corner-stone of Sanskrit syntax is the notion of Karaka. Karaka is a conceptual notion that mediates mapping between grammatical relations and case-suffixes.A syntactic unit is called *'pada'* and Panini defines *pada* as :

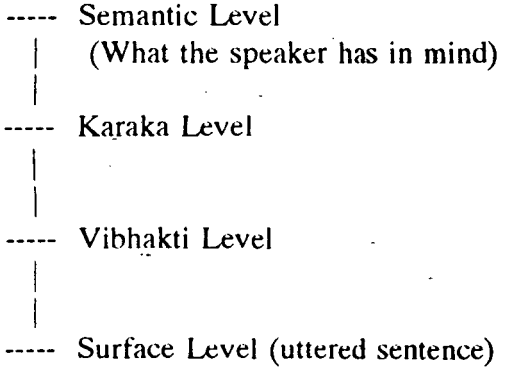
> *sup-ting antam padam* P.1.4.14.
>
> 'that which ends with a *sup* suffix (i.e. nominal suffix) or with a ting suffix (i.e. verbal suffix) is called pada'.

Thus, a Sanskrit sentence may be conceived as made up of inflected nouns and conjugated verbs.

2.3 KARAKA RELATIONS

The karakas are expressed in terms of a case-ending. While a strict one-to-one

correspondence between a *karaka* and a case-suffix may not be established, it may, in general, be stated that a particular case-suffix roughly corresponds to one karaka. A single case-suffix can represent more than one *karaka*, and conversely, one and the same *karaka* may be represented by more than one case suffix. There is no straight forward mapping from *vibhakti* to semantic relation between noun groups and verbs. The rules of grammar provide a mapping from *karaka* relations between the verb(s) and the nominals in a sentence. The *karaka* relations by themselves do not give the semantics. They specify relations which mediate between *vibhakti* of nominals and the verb form on one hand and semantic relations on the other. The various levels in the Paninian model are as shown below :



In all there are six types of karaka relations. What the *karaka* relations do specify with respect to a verb, are six or so relations of nominals to that particular verb. These provide for a mapping from *karaka* relations to semantic relations. Thus the *karaka*s provide the maximum necessary information relative to a verb. For example, the *karta karaka* may get mapped to agent for one verb, and experience for another, etc.

Karaka theory incorporates two other insights.

1. Each and every verb refers to an activity or event that can be further sub-divided into

a complex of activities. Each of the sub-activities has its own semantic relations and karaka relations.

Each of the sub-activities has its own semantic relations with associated objects. Thus, for each of the sub-activities there will be appropriate *karaka* relations.

Even though the verb representing a main activity may be used in a sentence, the *karaka* relations specified might correspond to a particular sub-activity. This will have obvious consequences for semantic relations.

2. *Karaka* relations also depend on the concept of *Vivaksha* refers to the speaker's viewpoint or attitude towards the activity. A sentence is not a statement of an objective activity but of the objective activity coloured by the speaker's viewpoint. (*Vivaksha* should not be confused with the speaker's intentions which come under pragmatics.) Usually *vivaksha* affects the choice of verb form, which in turn affects the *karaka* relations and *vibhakti.* For example, the objective activity of falling of the leaf of a tree may be expressed as

vrksat parnam patati (The leaf falls from the tree)

and also

vrksasya parnam patati (The leaf of tree falls)

Karta karaka holds between that nominal and a verb in a sentence, whose referent is swatantra or the most independent or autonomous out of all the *karaka* nominals that are expressed by the speaker. However, it is with respect to the activity implied by the verb.

Every verb in a sentence refers to an activity. Sometimes the verb can be ambiguous, in which case it may refer to several activities. When used in a sentence, it may refer to any one

of its possible activities. *Karta* of a verb in a sentence is one which is the *'ashraya'* or base of the activity.

There are six types of *karakas* in Sanskrit. These are *karta, karman, karana, sampradana, apadana* and *adhikarana.* Given below are the definitions of the same :

1. *KARTA* : The most independent person /thing in the performance of an action.
2. *KARMA* : The object most desired (to be obtained) by the agent of the action.
3. *KARANA* : The efficient means in the performance of an action.
4. *SAMPRADANA* : The recipient of the object of the action 'da' (to give).
5. *APADANA* : The static object in the action of separation.
6. *ADHIKARANA* : The substratum of the action performed.

We now give the rules in the Astadhyayi that relate these *karakas* with the case-suffixes (*vibhakti*). Only the most general rules are mentioned.

1. *pratipadikartha-linga-parimana-vacanamatre prathama* (P.2.3.46)

   Where the sense is that of the Crude form or where there is the additional sense of gender only, or measure only, the first case-affix is employed.

2. *karmani dvitiya* (P.2.3.2)

   When the object is not denoted by the termination of the verb, &c. i.e. the verb does not agree with it, the second case-affix is added to the word.

3. *kartrkaranayos tritiya* (P.2.3.18)

   In denoting the agent or the instrument the third case- affix is employed.

4. chaturthi sampradane (P.2.3.13)

   In denoting the *sampradana-karaka* the fourth affix or Dative is employed after the noun.

5. *apadane pancami* (P.2.3.28)

When the *Apadana-karaka* is denoted , the fifth case- affix is employed.

6. *saptamyaadhikarane ca* (P.2.3.36)

The seventh case-affix is employed when the sense is that of location.

2.4 SENTENCE FORMATION BASED ON KARAKA :

Example 1: Let us say we want to construct a sentence 'Rama reads a book'. Isolated words *ramah, pustakam, patathi* are connected with the words in the sense of agent *(Karta),* object *(Karaman)* and action *(Kriya).* In Ashatadyayi, derivation of words and their interrelation in a sentence starts from meaning. The lexicon *(dhatukosh)* contains a list of verbal roots and underived words *pratipadika.* The items are selected from the lexicon and the semantic relation between them is decided. Nominals are assigned gender and number, and the verb a reference of time.

If we choose the present tense *(varthamana kala)* verb gets *lat.* In our case *Ramah* is the agent, and book *(pustaka)* is the goal.

By 3.4.69, Lat expresses (1) agent or (2) goal or (3) the state (where the verb lacks goal). If we choose *Lat* to represent the agent, after transformations the final verb form is *patathi.* The goal can be expressed by 2.3.2 accusative case, since it is not expressed yet *(anabhihita),* (by 2.3.2) we get *pustakam (pustaka + am).* Verb ending already expressed agent (by 3.1.68 *karthari shap*), hence we cannot express it again. Now second or third case ending becomes applicable (by 2.3.2 and 2.3.18). But since we have to only express gender and number (by 2.3.46), the nominal stem gets the nominative case (su) and we form the word *(rama + su) ramah.*

agent

rama + su | *pata + thip*

pustaka + am

goal

Final form is : *rama(h) pustak(am) pata(thi).*

2.5 ROLE OF MORPHOLOGY :

We shall now deal with morphology of Sanskrit, which is of prime consideration in sentence generation.

2.5.1 Subanta

A *subanta* is made up of a nominal base and a nominal suffix. The nominal base is called *pratipadika.* There are 21 nominal suffixes in Sanskrit. There are 7 *vibhaktis* and 3 vachanas in Sanskrit. The 21 nominal suffixes are arranged in 7 x 3 paradigm. The suffixes are listed as :

vibhakti	*ekvachan*	*dvivachan*	*bahuvachan*
prathama	*su*	*au*	*jas*
dvitiya	*am*	*aut*	*sas*
tritiya	*taa*	*bhyam*	*bhis*
chaturthi	*ne*	*bhyam*	*bhyas*
panchami	*nasi*	*bhyam*	*bhyas*

shashti	*nas*	*os*	*aam*
saptami	*ni*	*os*	*sup*

2.5.2 Tiganta

This deals with the conjugation of verbs. Panini divides all the verb roots *(dhatu)* into ten *ganas.* They are *Bhavadi, Adadi, Divadi, Tudadi, Churadi, Tanadi, Rudhadi, Kriyadi, Kandvadi, Juhotiyadi.*

There are about 2000 *dhatus* in Sanskrit. The *bhavadi gana* is by far the largest one comprising of almost 40% of all the dhatus. The transformation rules acting on verbal roots are different for each gana and they are the same within a particular *gana.* The verbal root from a specific *gana* can then be a *parasmaipadi, atmanepadi* or *ubhaypadi.* Knowledge of *lakara* (which pertain to tense or mood of a verb) is required to specify a desired action[6]. Sanskrit has ten *lakaras* namely *lat, lit, lut etc.*. Like English, we have three numbers or purushas in Sanskrit namely *uttamapurusha* (first), *madhyamapurusha* (second) and *prathamapurusha* (third). And finally we need a *purusha* to form a *tiganta pada.*

CHAPTER III

SENTENCE GENERATION

3.1 SENTENCE REPRESENTATION

Panini doesn't formally define the term *vakya* (sentence), but the word is used in Astadhyayi. A sentence is a series of connected words. To express the connection between the words, Panini uses different non-technical terms such as joining (1.4.59 *yoga*), syntactically connected (2.1.1 *samaratha*), wish or desire (3.2.114 *sakanksha*) and so on.

In Sanskrit, the relation which one word bears to another in a sentence is determined by its grammatical form. No change occurs in the meaning of the sentence, how-so-ever the order of the words be changed. Therefore, the mere order of word is not of material importance, though a perfect arbitrariness in that respect is not allowable. But in English and other languages, deficient in inflection, order plays a vital role. If the order of words is changed, a corresponding change in meaning takes place. Sanskrit Syntax also takes into account the meaning and use of participles, the various tenses and moods, etc. The usual order of words in a sentence is, first the subject with its adjuncts, then the object with its adjuncts, followed by the adverbs and lastly the predicate.

In our system the sentence representation is done in the following manner.

Every verb essentially represent an action and nouns represent the substantives that play a role in bringing about this action. The noun-verb relation is specified in terms of karakas. The dhatu

lexicon contains the verbs along with the *karakas* it takes. Every verb takes some karakas invariably (which are called *anivarya karakas*) and some optionally (which are called *vaikalpika karakas*). Based on the karaka specifications the system takes the noun inputs. For example, if the verb takes *karma-karaka* a corresponding noun is expected. The system contains a noun lexicon wherein are stored its attributes.

Based on the input information the sentence is generated. As mentioned above, the present system is developed for generating simple sentences.

3.2 SEMANTIC BASIS OF SENTENCE GENERATION

The generation of a sentence starts with the meaning the speaker intends to convey . Depending on the meaning the speaker chooses the lexical elements and grammatical elements (suffixes etc). The semantics of the sentence is also conditioned by the subjective element, in terms of the speaker's view point of the world (called *vivaksa* in Sanskrit). Once these initial elements are selected the rules of grammar operate on them. The processing takes place, which includes morpho-phonemic changes and finally the correct sentences are generated. This, in brief, explains the semantic basis of entire grammatical operations.

As in the case of grammar, in our system too, the lexical elements are either nouns or verbs. The suffixes are *sup* suffixes (nominal suffixes) or *ting* suffixes (verbal suffixes). The primary, secondary or compound suffixes (*krit, tadhita and samasa.pratyayas*) are not considered here.

3.3 ROLE OF MORPHOLOGY IN SENTENCE GENERATION

In Sanskrit, the emphasis is on words rather than on sentences. A word can further be split

into two simpler elements, (1) a thing (i.e. *pratipadika*) or action (i.e. *dhatu*), and (2) the part containing information connected with thing or action (*karaka*). The word, whether it is a noun or a verb, that can be employed in the language is called *pada* (1.4.14). Ashtadhyayi provides for introducing affixes after bases. Nominal bases can also be derived by affixing suffixes called *krit*, *taddhita* or *samasa*. Similarly, verbs can be derived by affixing *san*, *kamyach* suffixes etc to the primary verbal roots.

In the previous chapter it was mentioned that the *karaka* relations (i.e. noun-verb relations) are represented by the *vibhaktis* (i.e. case endings). While a one-to-one mapping of the *karaka* to *vibhakti* is not possible, it may be stated that each *vibhakti* roughly corresponds to one *karaka*. In our system we have taken this approach for implementation.

Let us say we want to generate a sentence, Rama reads the book, *Ramah pustakam pathati*. The various words are generated by morphological process. This sentence comprises the words *ramah*, *pustakam* and *pathati*. We describe derivation of these forms below :

For the noun formation, let us take the word *rama* - a masculine name singular in number (singular, dual, plural). Now the crude base (*pratipadikam*) is rama (either derived from the verb *ram* or by the fact it is a meaningful string of sounds) by (1.2.45). To this base, we add affix *su* (from *sup* 4.1.2) to denote the singular nominative case. After this, these two elements go through transformations according to the rules applicable at every state; this process is summarized as follows:

3.3.1 Subanta (Noun Phrase)

The noun ***ramah*** is derived as follows :

By (1.2.45) Rama is a crude form or nominal base (*pratipadikam*). By (4.1.2 *pratyayah*, 4.1.3 *paraschcha*) the 21 case affixes *sup* come after the nominal base *Rama*. By (1.4.102) *supah*, the sets of three is called singular, dual and plural. By (1.3.2) *u* in *su* is an *ith*. With (1.3.9) (1.1.60) *u* is deleted. By (1.4.14) (*Rama* + *s*) is a *pada*. By (8.2.66) *s* at the end of the *pada* becomes *ru*. Now *u* in *ru* is ith (by 1.3.2) and deleted (by 1.3.9 *tasya lopah* and 1.1.60 and *adarshanam lopah*). By 1.4.110 and 8.3.15 it becomes *visarjaneeya*. Hence the form *Ramah*. The above process may be conceived as proceeding in the following steps. The numbers on the right side correspond to the rule numbers of Astadhyayi.

rama + su P.4.12. / P.4.1.3

rama + s P.1.3.2 / P.1.3.9

rama + ru P.1.4.14 / P.8.2.66

rama + r P.1.3.2 / P.1.3.9

rama + h P.1.4.100 / P.8.3.15

> ramah

Note: Only the important rules that go into the generative process is shown here. Several other rules such as interpretive *(paribhasha)* rules, evaluation of the rule hierarchies, are not shown here. But it is enough here to note that the process is deterministic and has all the primitive rules necessary.

Procedure for the word *pustakam* is similar to the example above, but suffix *am* (2nd case - accusative singular in number) is used.

pustaka + *sup* by 4.1.1, 3.1.1, 3.1.2 and 4.1.2. *pustaka* + *am* by 1.4.108 and 1.4.22 (accusative singular) by 7.1.24 substitute am in place of *su* and *am*. -> *pustakam* by 6.1.107.

3.3.2 Tiganta (Verb Phrase)

Outline of the formation of verb pathati is as follows: The lexicon connected with Ashtadyayi contains the list of verbal roots called dhatupata. The position of the root in the dhatupata and the control characters (such as *iths*, and anudatta svaritha accents on the vowels) associated with these roots provide the necessary information about the morphological characteristics. For the verbal roots selected, suffix L is introduced to convey time (present, past, future.. lat, lit..) and agent. The suffix L is replaced by verb endings or participle affixes, is divided into two groups called *parasmaipadam* and *atmanepada*. Each of these group contains nine endings to signify person (*prathama, madhyama, uttama*) and number (single, dual and plural). By the end of generative pocess, the verb will contain information about action, time, person, active or passive and so on. Now, let us form the verb pathati (meaning - reads). The root is pathA (root #318 in the dhatupata and will come under the first conjugation (*bhvadhi*) of total ten).

Here also only the important rules are listed.

pathA -> *path* (by 1.3.1 *bhuvadayo dhatavah*) it is a verbal root. Being upadesha (since it is in dhatupata list of Panini), by 1.3.2 (*Upadeshejanunasika ith*) the nasal A is indicator and deleted (by 1.3.9 *thasya lopah* and 1.1.60 *adarshanam lopah*).

The process of verbal conjugation is described below:

path + *lat* P.3.1.91 / P.3.2.123

path + *la* P.3.4.77

path + *tip* P.3.4.78

path + *ti* P.1.3.3 / P.1.3.9

path + *sap* + *ti* P.3.1.68

path + a + ti P.1.3.4 / P.1.3.9

> *path + a + ti*

> *pathati.*

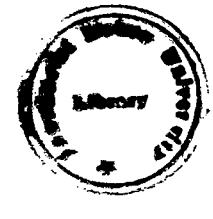
Thus with the subanta and tiganta procedures, the sentences are generated.

3.4 COMPUTATIONAL LEXICON

Knowledge can be represented using an appropriate data structure, which is referred to as lexicon. Lexicon is quite essential and necessary for Natural Language Processing (NLP) systems, i.e. parser, generator, translator, etc. It is because almost all the components of NLP systems refer to lexicon in order to accomplish their task. A simple lexicon for NLP systems may contain syntactic, semantic and contextual knowledge.

The lexicon may be defined as a collection of lexemes along with their related attributes. A lexeme is considered as one word taken along with its grammatical attributes, semantic attributes, conjugate list, computational words (derived words), relations, possible surface realization, pronunciation and meaning of the word.

In our system we maintain two lexicons, one for verbal roots and other for noun-bases.



CHAPTER IV

IMPLEMENTATION

Prolog has been chosen as the vehicle for implementation. It provides backtracking and pattern matching mechanisms which are very useful for Natural Language Processing Systems. Modular programming approach has been adopted in the design and implementation of the software which makes modifications and maintenance easier. The organization of the system is shown in fig 4.1.

The implementation comprises basically following major modules:

1. LEXICON
2. GENERATOR
3. MAIN MODULE

These modules are described in detail below.

4.1 LEXICON

A language is contained in its lexicon. The system operates on verb (*dhatu*) and nominal (*pratipadika*) data. Therefore, this module consists of **DHATUKOSH** (for verb) and **PRATPADIKAKOSH** (for nominals). Both are discussed below.

4.1.1. DHATUKOSH

This module has a verb database and procedures which operate on it. The database is used by almost all the procedures in the module. So the entry structure and an efficient retrieval

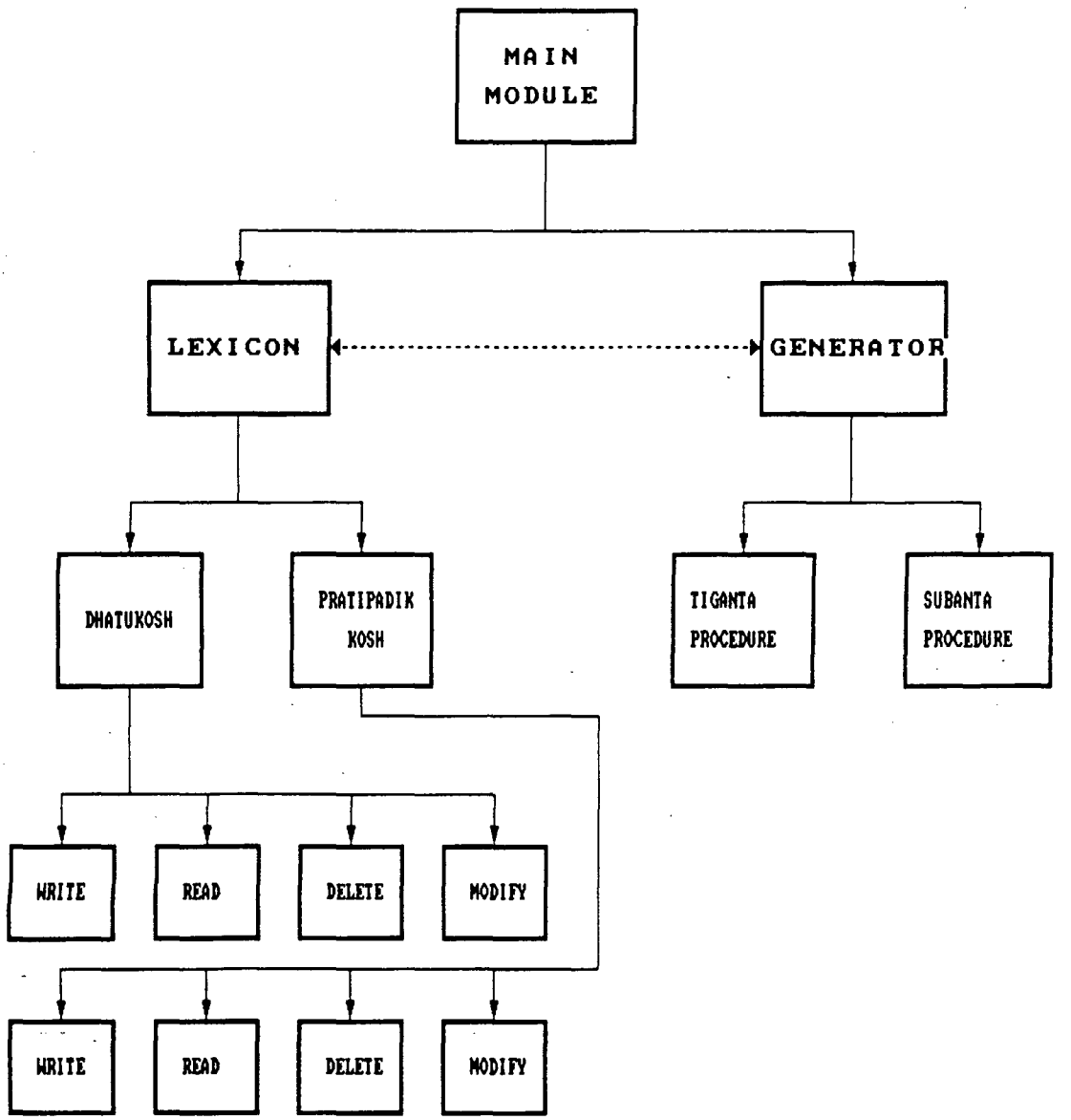


Fig. 4.1 : System Organization

mechanism is vital. An entry in the lexicon comprises a key and other attributes. The key is the basis of search in the database. To locate a particular entry, the key must be unique. The key in our database is dhatu (verbal root). The other attributes associated with dhatu entry are -

1. *Anyaroop* (i.e. the second form of the dhatu, if any)
2. *Lakara-samooha* (Tensegroup sarvadhatuka/ardhadhatuka)
3. *Gana* (each dhatu corresponds to one of the ten ganas)
4. *Pada* (parasmaipadi, atmainaepadi, ubhaypadi)
5. *Anivarya karaka soochi* (karakas the dhatu must require)
6. *Vaikalpika karaka soochi* (karakasoptionally required by the dhatu)

The module has two operational modes, (1) Maintenance mode and (2) Processing mode. The full strength of the module is available through maintenance mode. In processing mode only read and write operations are available. Below we describe the functioning of DHATUKOSH in maintenance mode.

When the module is run the MAIN MENU of the program appears on the monitor with four options available to the user (Fig A.1). They are WRITE, READ, DELETE and MODIFY. The user can select any option or optionally may come out of the system by pressing ESC key.

Each option is associated with procedure denoted by the option name. They are described below.

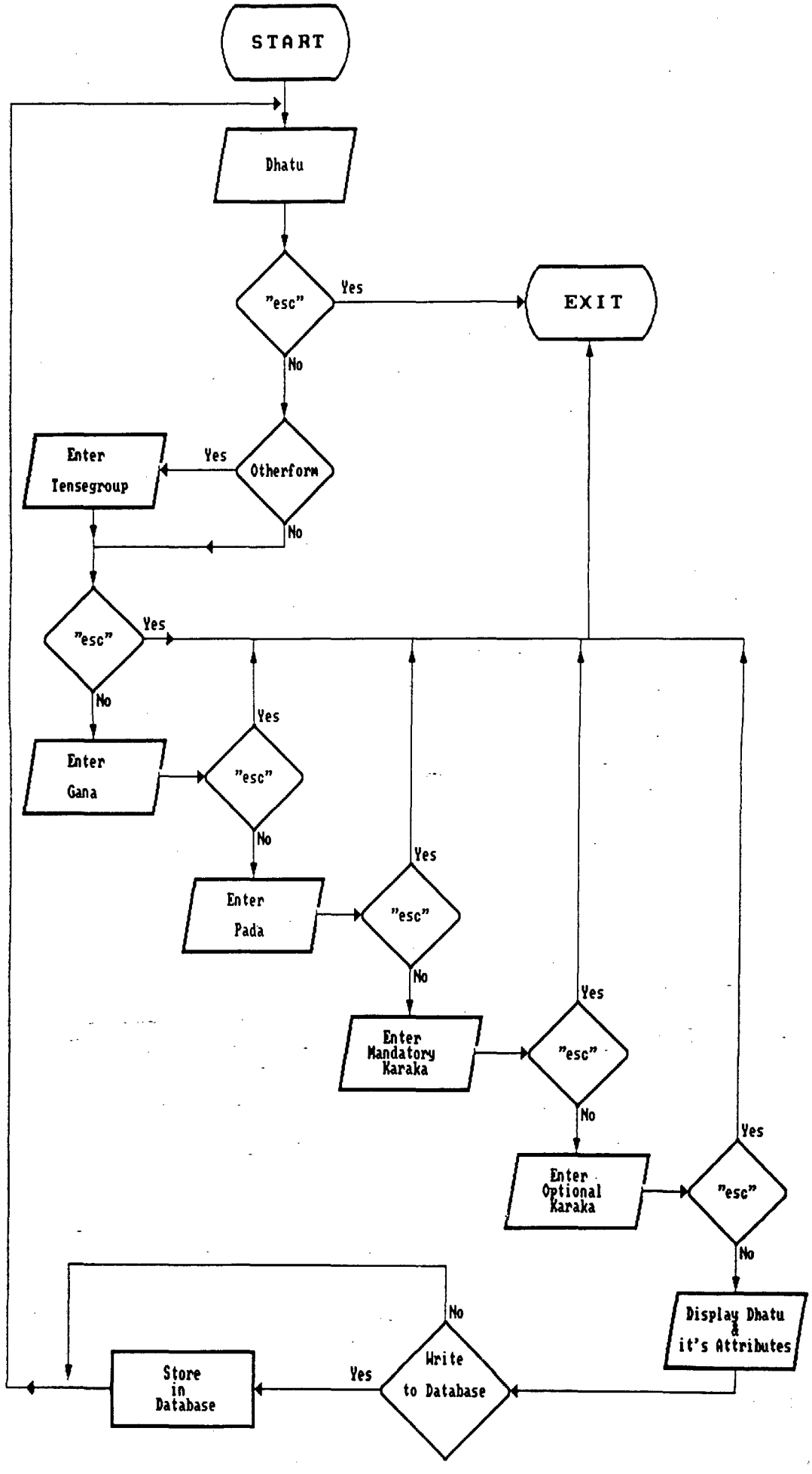
4.1.1.1 Write

The purpose of this procedure is to write new entries for the verb and its associated attributes in the verb database. If the lexicon does not already exist, it is created anew. When a user selects

the WRITE option from the MAIN MENU, this procedure gets activated. The operations involved in this procedure are shown in the flowchart Fig 4.2.1

First the user is prompted to input dhatu from the keyboard, as in Fig A.2.1. The user enters a dhatu . The dhatu is searched in database the verb database; if the dhatu is present in the database an appropriate message is displayed to the user (Fig A.2.2). If the dhatu is not present the user is asked to input anyaroop (the other form of the verb), if any (Fig A.4). If a non-null string is input for the anyaroop, the user is asked to select lakara-samooha (the tensegroup in which the verb has different form) from a pop-up submenu (Fig A.2.4). If the verb does not have an anyaroop, the user just presses Return key and the pop-up submenu does not appear. After this, the user is required to select a gana from the gana submenu (Fig.A.2.5). Having selected a gana, selection of pada is required. The user may select the desired pada from a pop-up menu (Fig A.2.6).

Now, the anivarya (mandatory) karakas the dhatu must require, have to be selected. The user may select a list of anivarya karakas from a list of karakas displayed on the monitor as in Fig A.2.7. The vaikalpik karakas now may be selected by the user (Fig A.2.8). It should be noted here, that those karakas which were marked to be anivarya karakas for the dhatu are not available in selection of vaikalpik karaka. Suppose, we take a hypothetical case of dhatu which takes all the six karakas invariably, then the fig A.2.8 doesn't appear at all. After selection of vaikalpik karakas, all the inputs necessary for a database entry are available and ready to be stored. Suppose, the dhatu entered to the system is already in the database (possibly with different attributes) then the user is given a second chance and a confirmation is asked from the user before overwriting the existing entry.


START
Dhatu
"esc"
Yes
EXIT
No
Enter Tensegroup
Yes
Otherform
No
"esc"
Yes
No
Enter Gana
Yes
"esc"
No
Enter Pada
Yes
"esc"
No
Enter Mandatory Karaka
Yes
"esc"
No
Enter Optional Karaka
Yes
"esc"
No
Display Dhatu & it's Attributes
No
Write to Database
Yes
Store in Database

If the user presses ESC during input/selection while in WRITE option, the current input is abandoned and the screen goes back to the Fig A.2.1 screen. Here it should be noted that no check is maintained for the correctness of input dhatu. It is assumed that the user knows the correct dhatu and its attributes. It is possible to enter nonsensical dhatu such as "7!agfgf9" in the database. Probably the user authorized to invoke this option will be a privileged one.

4.1.1.2 Read

When the user selects the READ option from the MAIN MENU, this procedure is invoked. The user is asked to enter a dhatu (Fig A.2.1). The input dhatu string is taken as key for search in the database. If a match is found for the dhatu string input, the entry is displayed (Fig A.3.1). On failure to find the input string in the database, an appropriate message is flashed on the screen (Fig A.3.2)

The sequence of operations possible through this procedure is shown in the flowchart Fig.4.2.2.

4.1.1.3 Delete

This procedure is called on the selection of DELETE option. The user is required to input the dhatu, for which the entry from the database is to be deleted (Fig A.2.1). If the dhatu input is found to exist in the database, the corresponding entry is deleted. For a nonexisting dhatu, the user gets an appropriate message (Fig A.3.2).

The flowchart for this procedure depicts its working as in fig 4.2.3.

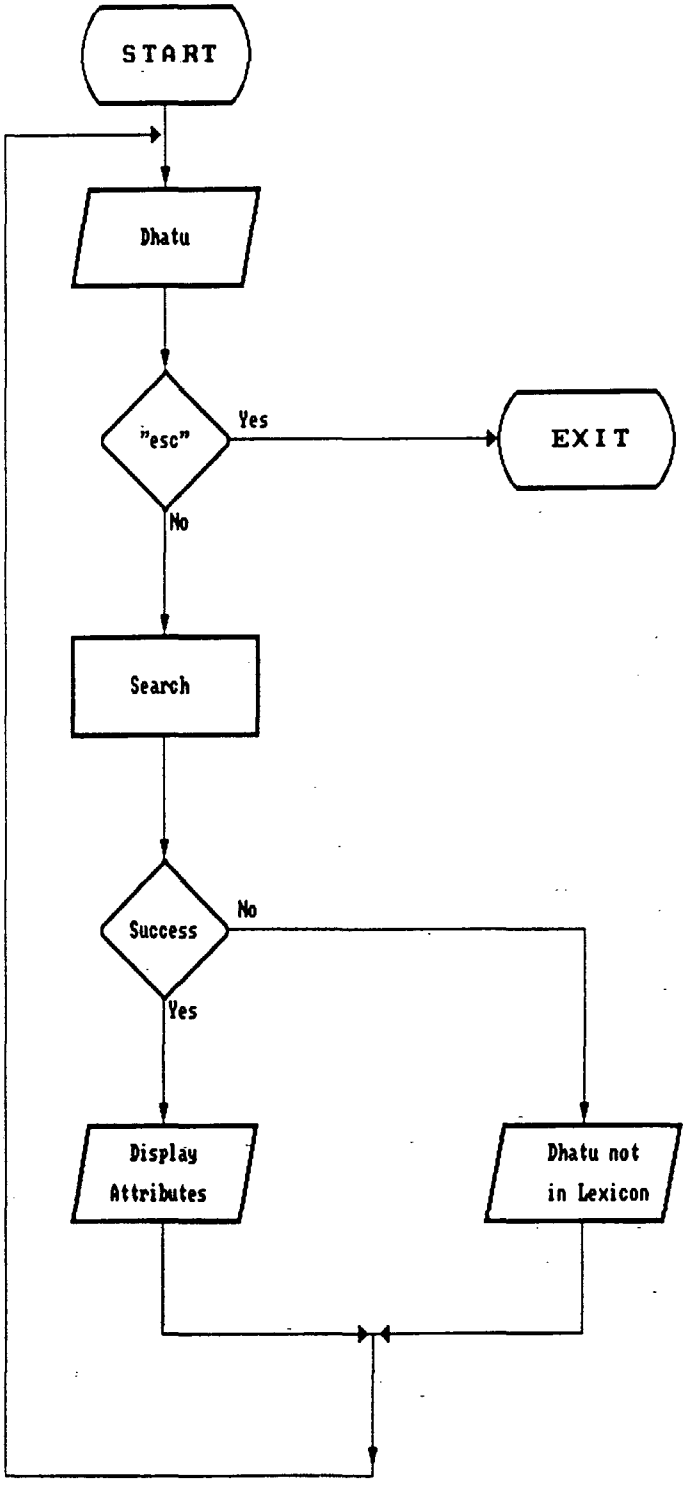


Fig. 4.2.2 : Flow chart for READ procedure

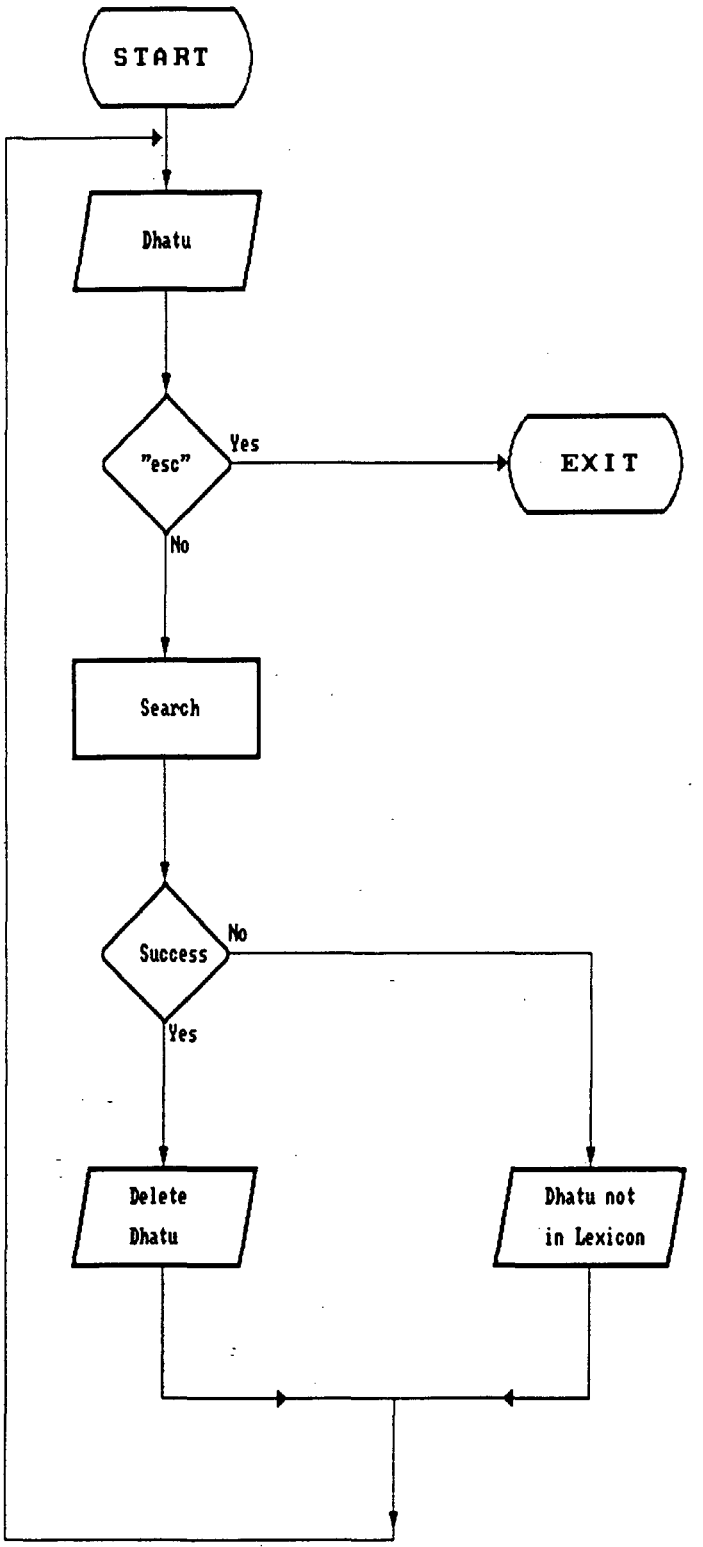


Fig. 4.2.3 : Flow chart for DELETE procedure

4.1.1.4 Modify

If the user wishes to change a few attributes of a dhatu in the database, he/she may select the MODIFY option from the MAINMENU. Changing all the attributes is equivalent to writing a new entry, WRITE option is a better choice for doing it. On selecting MODIFY, this procedure is invoked. The operations involved in this procedure are shown by a flowchart in Fig.4.2.4. The user inputs the dhatu, the entry of which is to be modified. The input string is search in the database. If the string is not found in the database, a message is displayed on the screen (Fig.A.3.2). Alternatively, if the input dhatu matches a key attribute in the database, all the attributes of the dhatu are retrieved and displayed. The user is asked to select attributes to be modified from the pop-up submenus. The selected attributes are then given new values, all through menus, and finally the entry gets modified as desired.

The control gets back to MAIN MENU on pressing ESC during the oprerations under this option.

4.1.2. PRATIPADIKAKOSH

This module has a nominal-database and procedures which operate on the database. The nominal-database is used by all the procedures in the module. The key in our nominal-database is pratipadika (nominal base). The other attributes associated with pratipadika entry are :

1. *Varga* (*sanjna, sarvnama, sankhyavachaka*)
2. *Linga* (*stri, puns, napunsaka*)

Like DHATUKOSAH, the module has two operational modes, (1) Maintenance mode and (2) Processing mode. The full strength of the module is available through maintenance mode. In

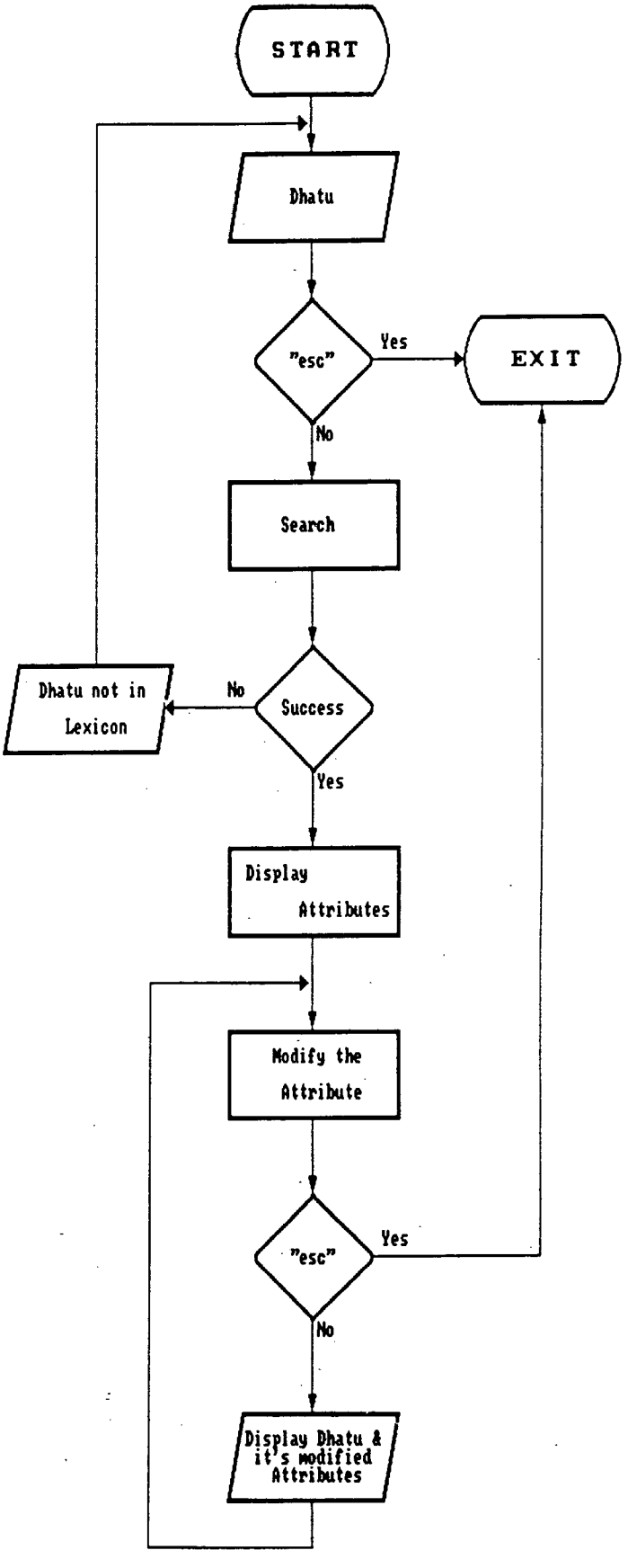


Fig. 4.2.4 : Flow chart for MODIFY procedure

processing mode only read and write operations are available. Below the functioning of PRATIPADIKAKOSH in maintenance mode is described.

When the module is run the MAIN MENU of the program appears on the monitor (Fig A.1). As is evident from the figure, four options are available to the user. They are WRITE, READ, DELETE and MODIFY. The user can select any option or may come out of the system by pressing ESC key.

With each option is associated a procedure of the same name. Functioning of these procedures is quite similar to that of the Read, Write, Delate and Modify procedures in DHATUKOSH module. One by one we describe these procedures below.

4.1.2.1 Write

The purpose of this procedure is to write new entries for the pratipadika and its associated attributes in the nominal- database. If the database file does not already exist, it is created anew. When a user selects the WRITE option from the MAIN MENU, this procedure gets activated. Flowchart in Fig 4.3.1 shows the working of this procedure.

First the user is prompted to input dhatu from keyboard. The user enters a pratipadika . The pratipadika is searched in the database. If the search is unsuccessful, a message is flashed on the screen. Otherwise, the user is required to select a varga from the varga submenu. After a varga is selected, a selection of linga, for the already input pratipadika is required.

Again, similar to Write Procedure of DHATUKOSH, it should be noted that no check is maintained for the correctness of input pratipadika. It is assumed that the user knows the correct pratipadika and its attributes.

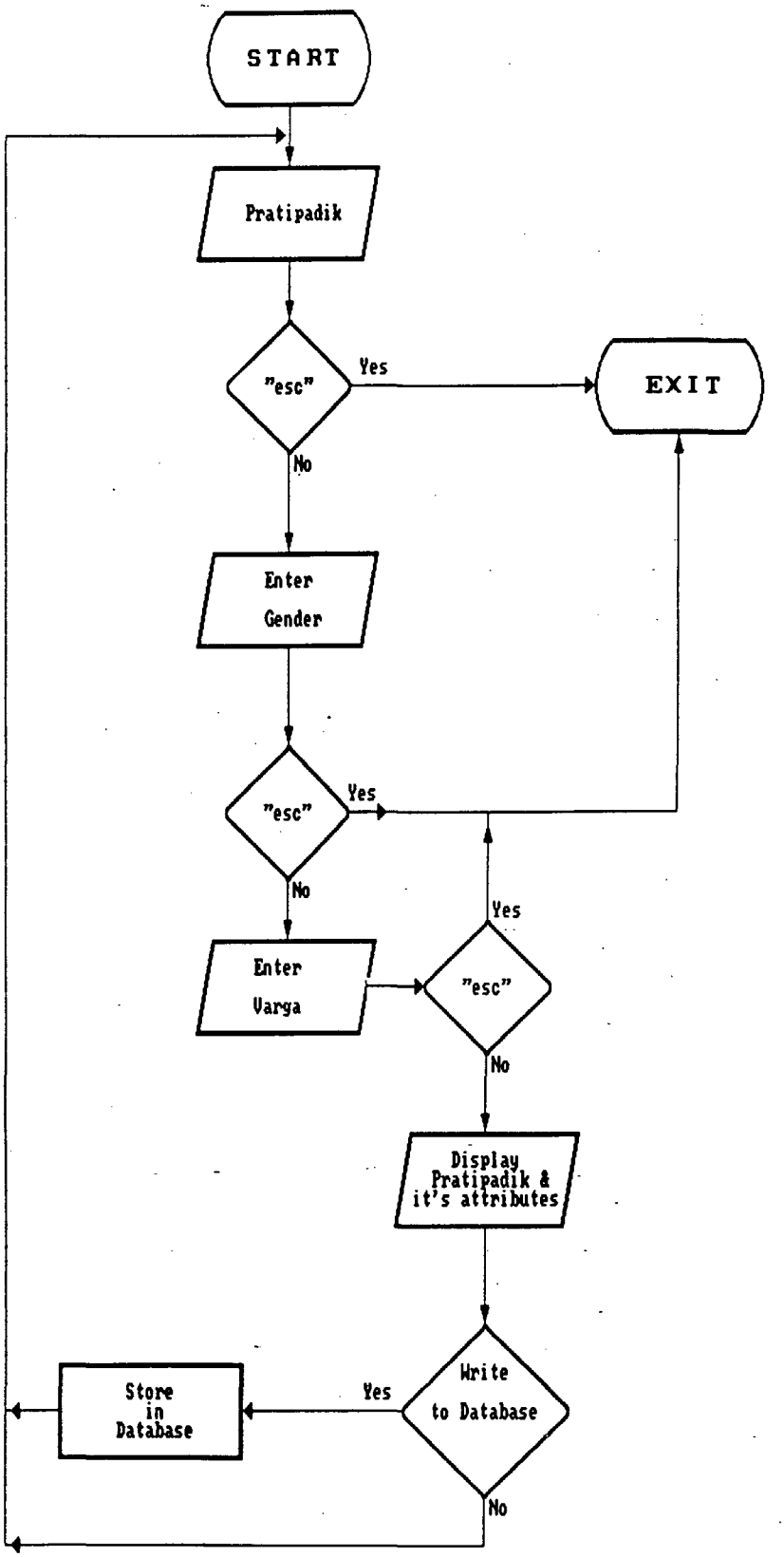


Fig. 4.3.1 : Flow Chart for WRITE procedure

4.1.2.2 Read

When the user selects the READ option from the MAIN MENU, the main module calls this procedure. The user is asked to enter string for a pratipadika. The input string is taken as key for search in the nominal-database. If a match is found for the pratipadika string input, the entry is displayed. On failure to find the input string in the database, an appropriate message is flashed on the screen.

The sequence of operations possible through this procedure is shown in the flowchart in Fig. 4.3.2.

4.1.2.3 Delete

If the user selects DELETE option, this procedure is called by the main module. The user is required to input the pratipadika, for which the entry from the databases is to be deleted. If the pratipadika input is found in the database, the corresponding entry is deleted. For a nonexisting pratipadika, the user gets an appropriate message from the system.

The flowchart for this procedure depicts its working as in fig 4.3.3.

4.1.2.4 Modify

The flow chart for functioning of this procedure is given in Fig. 4.3.4.

To change a few attributes of a pratipadika in the lexicon, the user may select the Modify option from the MAIN MENU. Changing all the attributes is equivalent to writing a new entry. On selecting Modify option this procedure is called by the main module. The user inputs the pratipadika, the entry of which is to be modified. The input string is searched in the database.

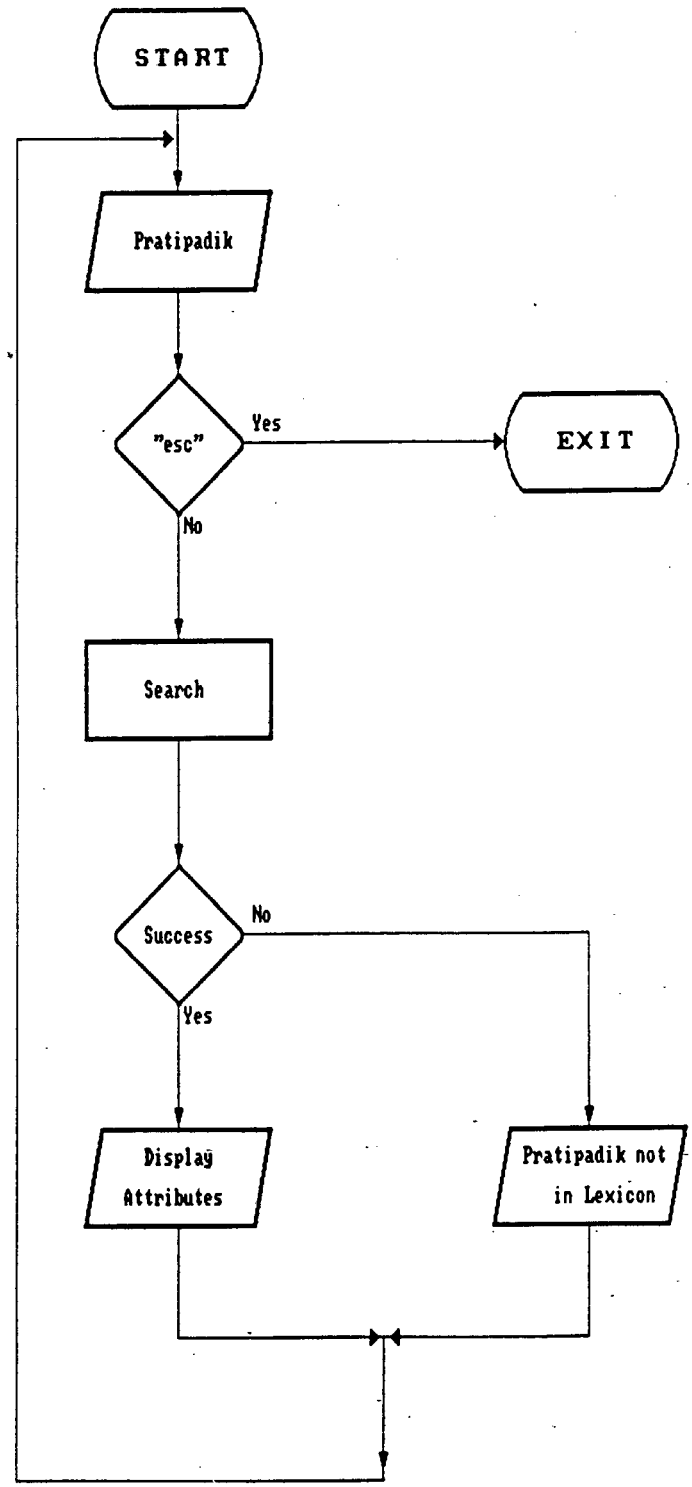


Fig. 4.3.2 : Flow chart for READ procedure

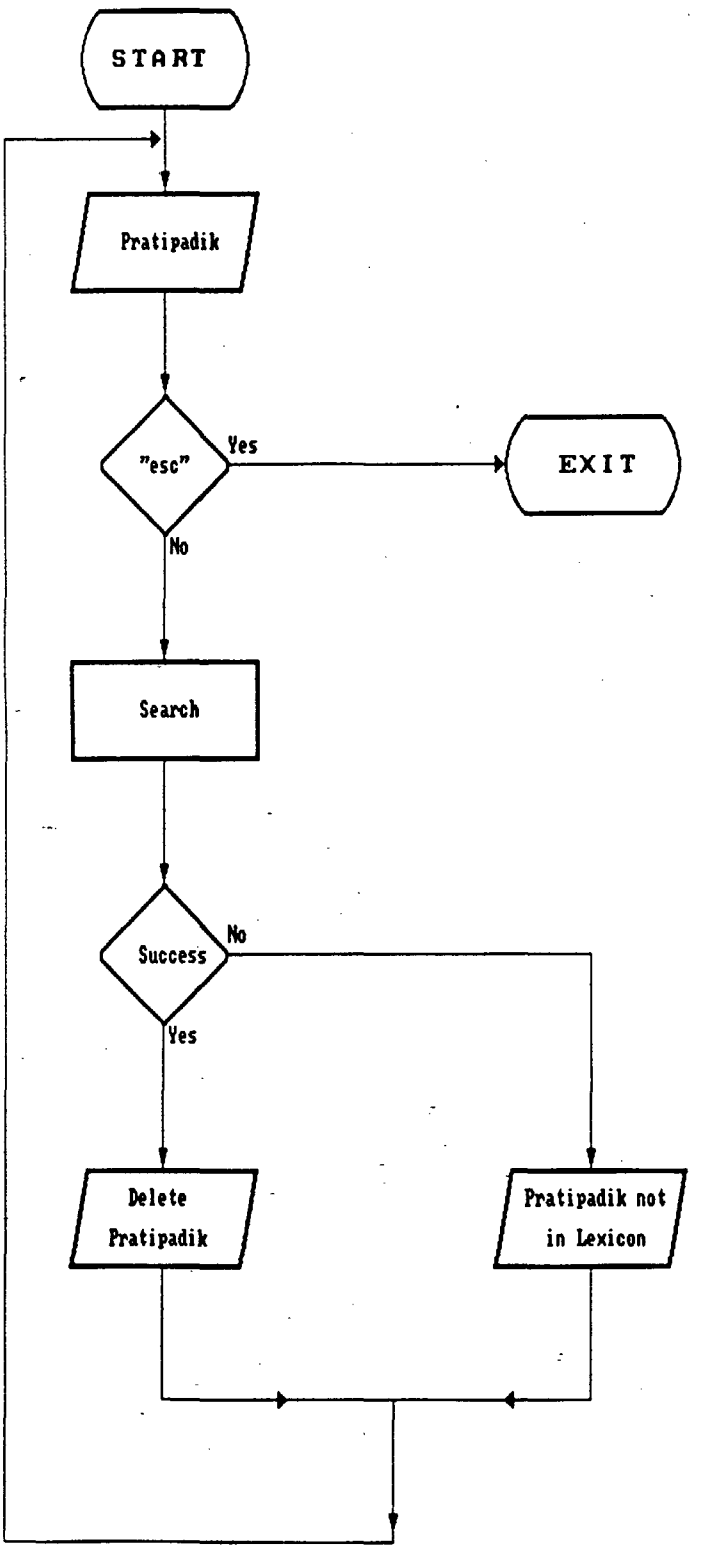


Fig. 4.3.3 : Flow chart for DELETE procedure

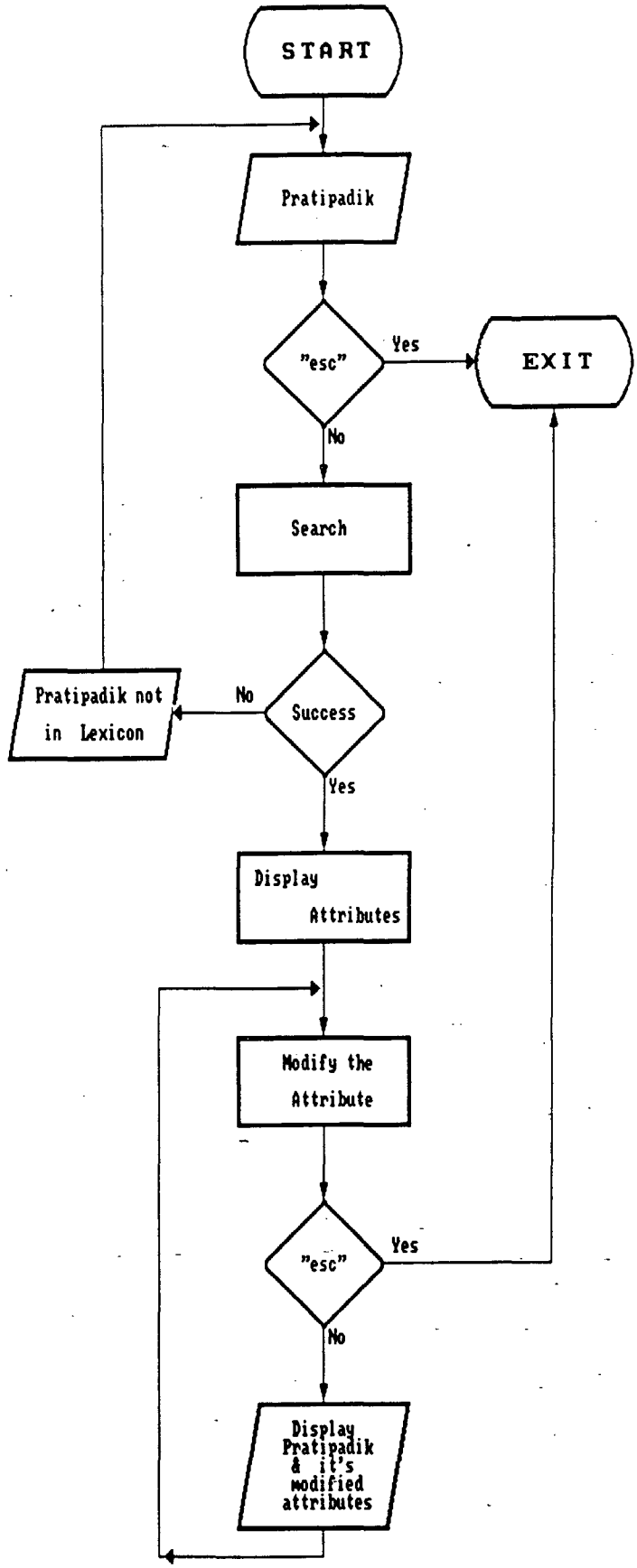


Fig. 4.3.4 : Flow chart for MODIFY procedure

If the string is not found in the database, a message is displayed to the user. Alternatively, if the input pratipadika matches a key attribute in the database, all the attributes of the pratipadika are retrieved and displayed). The user is asked to select attributes to be modified from the pop-up submenus. The selected attributes are then given new values, all through menus, and finally the entry gets modified as desired.

4.2 GENERATOR

The purpose of this module is to generate padas. This module takes its necessary inputs from the LEXICON module. The GENERATOR module has two procedures - SUBANTA (for generating subanta padas) and TIGANTA (for generating the tiganta pada) which are described below.

4.2.1 Subanta

The input parameters passed to this procedure are: pratipadika, varga, linga, vibhakti and vachan. The output of the procedure is a subanta pada.

A pada consists of two parts, a base part and a suffix part. There are a total of 21 suffixes. In the formation of final pada, the pratipadika and suffix undergo changes according to the rules of the grammar implemented in this module. These changes depend on the varga, linga and the pratipadik ending. This gives a changed base form and then the changed base form and suffix are then combined according to the sandhi rules implemented in SANDHI procedure to produce a finished subanta pada.

4.2.2 Tiganta

Dhatu, lakara, purusha and vachana values are passed as input parameters to this procedure from the main module. The output of the procedure is a finished tiganta pada.

A pada consists of two parts, a base part and a suffix part. There are a total of 18 suffixes. In the formation of final pada the dhatu and suffix undergo changes according to the rules of grammar coded in this module. These changes depend on the lakara, and gana. According to the gana of a dhatu an appropriate vikaran is selected from the lexicon. Combining of vikaran and again with dhatu brings changes in the base (dhatu) form. This changed base form and suffix are then combined according to sandhi rules to produce a finished tiganta pada.

4.3 MAIN MODULE

In the very beginning when the program is run, the main module checks for the existence of the verb-database and nominal- database files. If these files are not already existing they are created afresh, before proceeding further. The program then does initial set up of screen and the system is ready for the generation process.

The system prompts the user to give a dhatu (Fig A.4.1). The user enters the dhatu string through the keyboard (Fig A.4.2). The dhatu entered is searched in the verb database. The dhatu may either be already present in the verb database or it may not exist in there. This process is performed, essentially, by calling READ procedure of the DHATUKOSH module. In the first case, the dhatu entry is displayed on the screen. In second case, the WRITE procedure is called in its processing mode to insert the dhatu entry in the verb database. Next, the user is required to select a purusha from a pop-up submenu (Fig A.4.3). Lakara now has to be selected by the

user from a submenu (Fig A.4.4).

The system proceeds further for taking pratipadikas for each karaka associated with the dhatu (Fig A.4.5). The pratipadika is checked in the nominal database by the READ procedure of PRATIPADIKAKOSH in processing mode. If the pratipadika is present in the nominal-database, its attributes, linga and varga are retrieved. Otherwise, to insert the pratipadika entry in the nominal database, the WRITE procedure of PRATIPADIKAKOSH module in its processing mode is called. The user is then required to select vachan for the given pratipadika (Fig A.4.6). The system now asks pratipadikas for remaining karakas (Fig A.4.7).It may be mentioned here that while inputting pratipadika for anivarya (mandatory) karkas the user is forced to give non-null string. For vaikalpika karkas null-string is acceptable and for the null-string the user is not required to select a vachan. With this all the inputs for a sentence generation are over. Throughout the above process the user can come out to the main screen (Fig A.4.1) by pressing ESC.

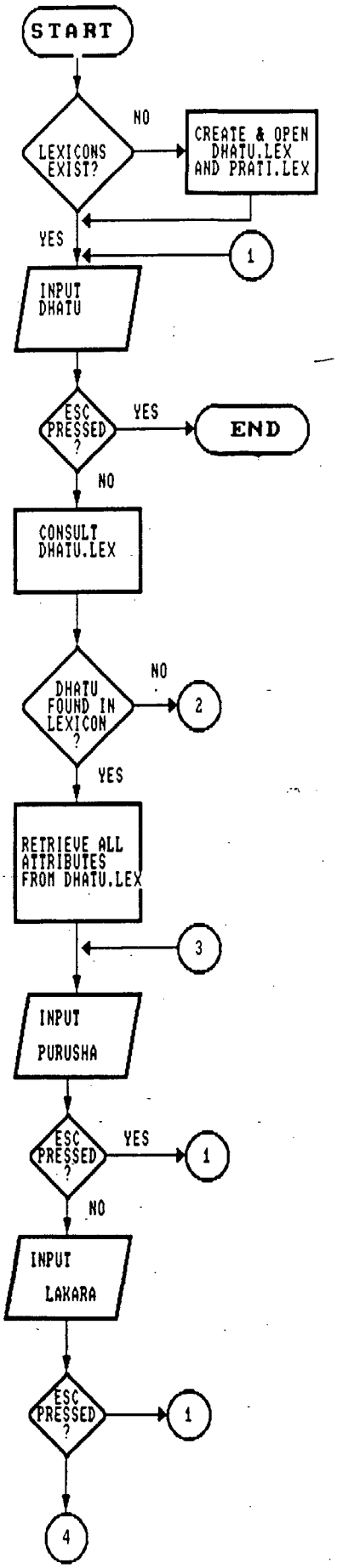
Before calling Subanta and Tiganta procedures, the main module finds the vachan for karta-karaka and this vachan shall be passed to the Tiganta procedure as one of the parameter. Another task which remains now is to find vibhaktis for each pratipadika. This vibhakti shall be needed in the formation of subanta pada for the given pratipadika.

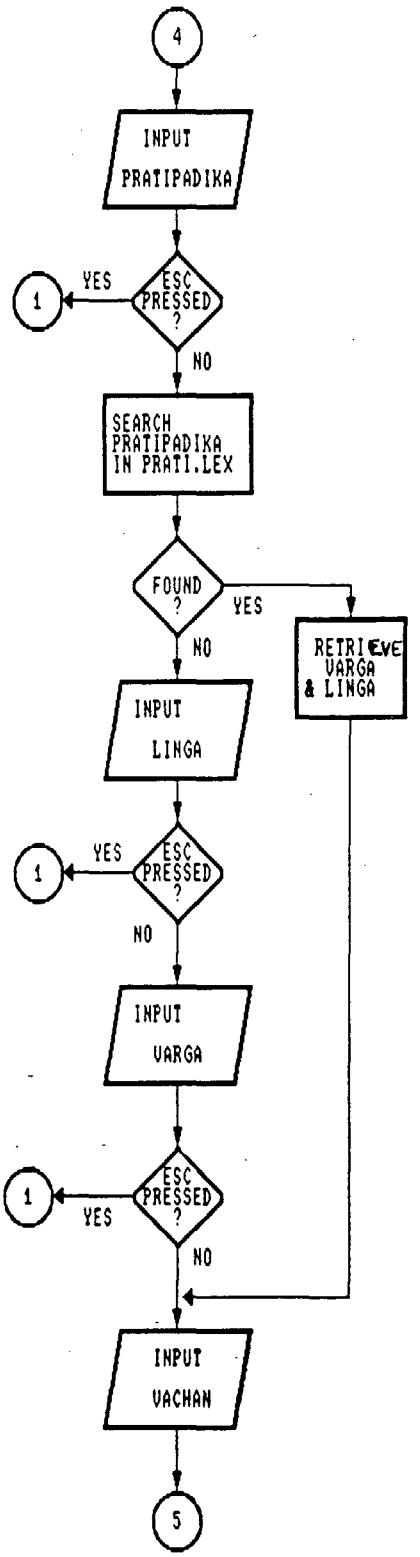
The generator module then calls Subanta procedure for each input pratipadika passing. With this, only thing left for a sentence is the tiganta pada.

The generator now calls Tiganta procedure. This module is only once for a sentence. The Tiganta gets dhatu, gana, pada, lakara, purusha and vachan as its input parameters. The Tiganta procedure gives a tiganta pada as its output.

Since our sentences are nothing but a string of one tiganta pada and one or more subanta pada, the sentence is ready and is displayed on the screen (Fig A.4.8).

A flowchart showing the process of sentence generations is given in Fig 4.4. The input specifications and the results of the system are given in Appendix.


START
LEXICONS EXIST?
NO
CREATE & OPEN DHATU.LEX AND PRATI.LEX
YES
1
INPUT DHATU
ESC PRESSED ?
YES
END
NO
CONSULT DHATU.LEX
DHATU FOUND IN LEXICON ?
NO
2
YES
RETRIEVE ALL ATTRIBUTES FROM DHATU.LEX
3
INPUT PURUSHA
ESC PRESSED ?
YES
1
NO
INPUT LAKARA
ESC PRESSED ?
1
4


4
INPUT
PRATIPADIKA
ESC
PRESSED
?
YES
1
NO
SEARCH
PRATIPADIKA
IN PRATI.LEX
FOUND
?
YES
NO
RETRIEVE
VARGA
& LINGA
INPUT
LINGA
ESC
PRESSED
?
YES
1
NO
INPUT
VARGA
ESC
PRESSED
?
1
YES
NO
INPUT
VACHAN
5

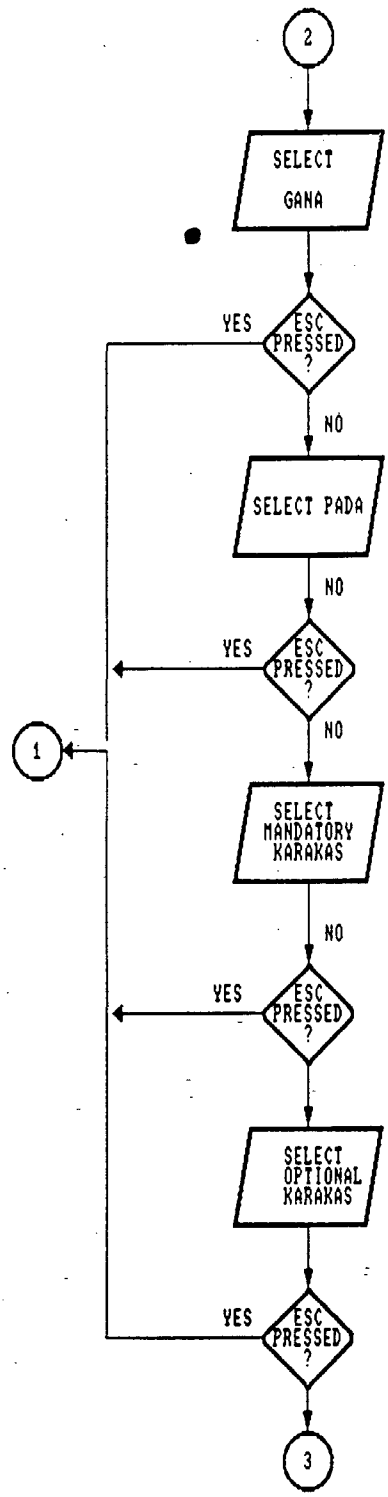

2
SELECT GANA
YES
ESC PRESSED ?
NO
SELECT PADA
NO
YES
ESC PRESSED ?
NO
1
SELECT MANDATORY KARAKAS
NO
YES
ESC PRESSED ?
SELECT OPTIONAL KARAKAS
YES
ESC PRESSED ?
3

Fig. 4.4 (cont..)

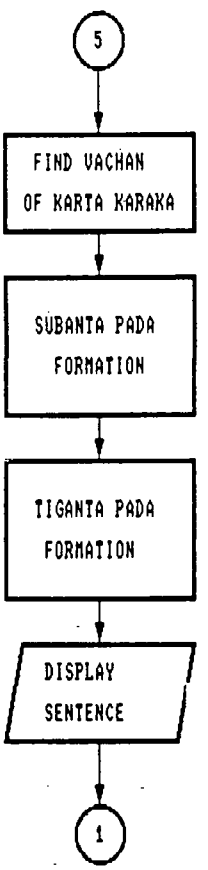

5
FIND VACHAN
OF KARTA KARAKA
SUBANTA PADA
FORMATION
TIGANTA PADA
FORMATION
DISPLAY
SENTENCE
1

Fig. 4.4 (cont..)

CHAPTER V

CONCLUSION

The present work is a modest attempt to develop a synthesizer to generate simple Sanskrit sentence based on Paninian Grammatical Framework. The system provides a user friendly environment and is menu driven. The implementation is done in Prolog language because of its in-built backtracking and pattern matching mechanism.

Though a vibhakti can denote more than one karaka and vice versa but because of the time limitation, only one-to-one mapping between karaka and vibhakti has been considered here. In addition to this, the present system does not account for sentences with adjectives, adverbs, participle and passive voice. No hierarchy of nouns is considered. Further, issues relating to semantics and pragmatics are needed to be considered.

Thus, there is enormous scope for future expansion. Once the system is completely developed it can be used in various areas of Natural Language Processing, such as teaching/learning, text understanding machine translation, database interfaces, text generation, etc.

The present work is augmented by the related work, namely, sentence analysis based on verb and karaka specifications. The system is developed in such a manner that it can be used for teaching/learning purposes.

Sample input specifications to the system and the output produced are given in appendix.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Allen, James. Natural Language Understanding.

Menlo Park, CA : Benjamin/Cummings, 1987.

[2] Harris, Mary Dee. Introduction to NLP.

Reston : Reston Publishing Company, 1985.

[3] Katre, Sumitra M. Astadhyayi of Panini.

Delhi : Motilal Banarsidas, 1989.

[4] Kiparsky, P. Some Theoretical Problems in Panini's Grammar.

Poona : Bhandarkar Oriental Research Institute, 1982.

[5] Mishkoff, Henry, C. Understanding Artificial Intelligence.

Delhi : BPB Publications, 1986.

[6] Patterson, Dan W. Introduction to Artificial Intelligence and Expert Systems.

New Delhi : Prentice-Hall of India, 1992.

[7] Rich, Elaine. Artificial Intelligence.

New York : McGraw Hill, 1983.

[8] Sager, Naomi. Natural Language Information Processing.

Reading, Mass : Addison-Wesley, 1981.

[9] Singh, G.V., et. al., The Word-Morphology of Sanskrit, Sinha, R.M.K. (ed.), Proceedings of Computer Processing of Asian Language (CPAL-2).

New Delhi : Tata McGraw-Hill, 1992.

[10] Syseca, A G. Prolog for NLP.

John Wiley & Sons, 1985.

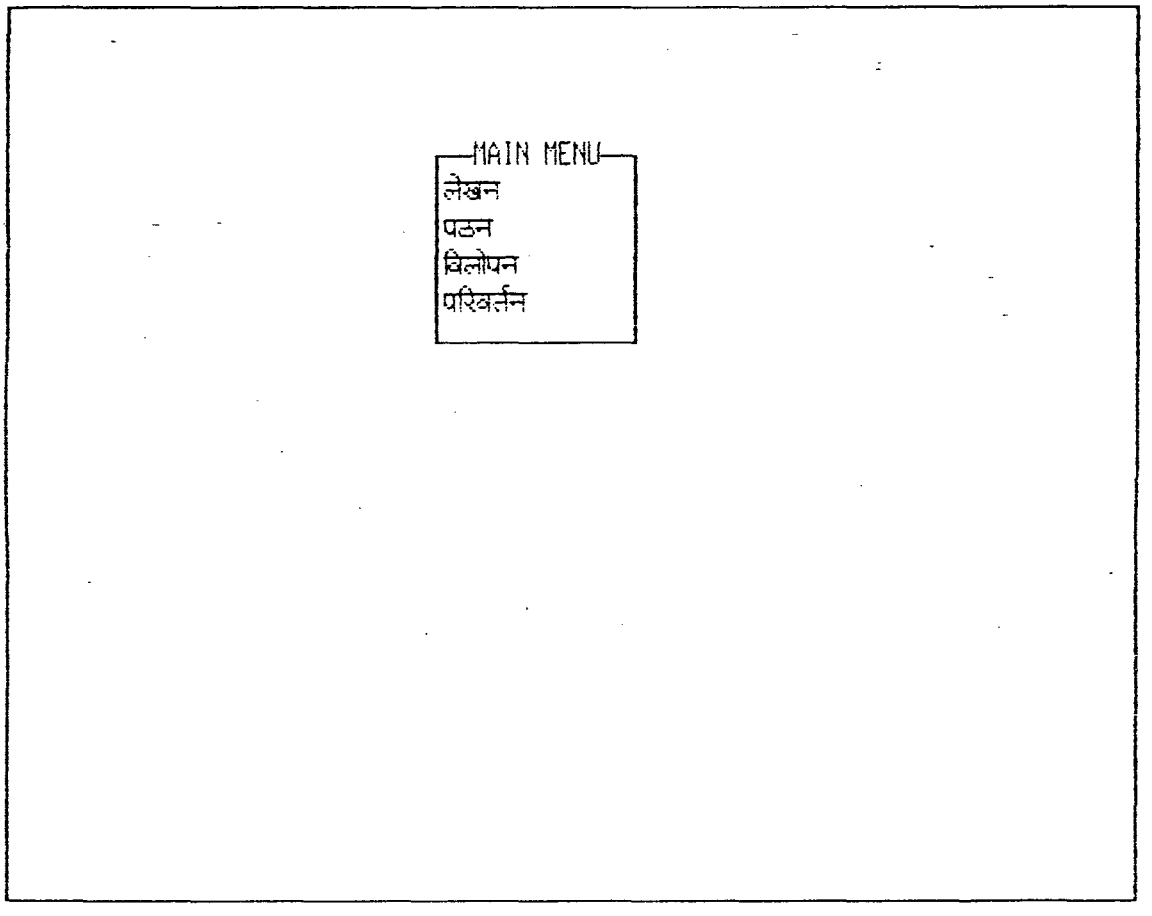
[11] Vasu, S C. The Siddhanta Kaumudi of Bhattoji Diksita.

Delhi : Motilal Banarsidas, 1982.

[12] Winograd, T. Language as a Cognitive Process, Volume I : Syntax

Reading, Massachusetts : Addison-Wesley, 1983.

APPENDIX


MAIN MENU
लेखन
पठन
विलोपन
परिवर्तन

धातु कोश

धातु दें :

निर्देश

निकास के लिए Esc दबाएँ

धातु क्लेश

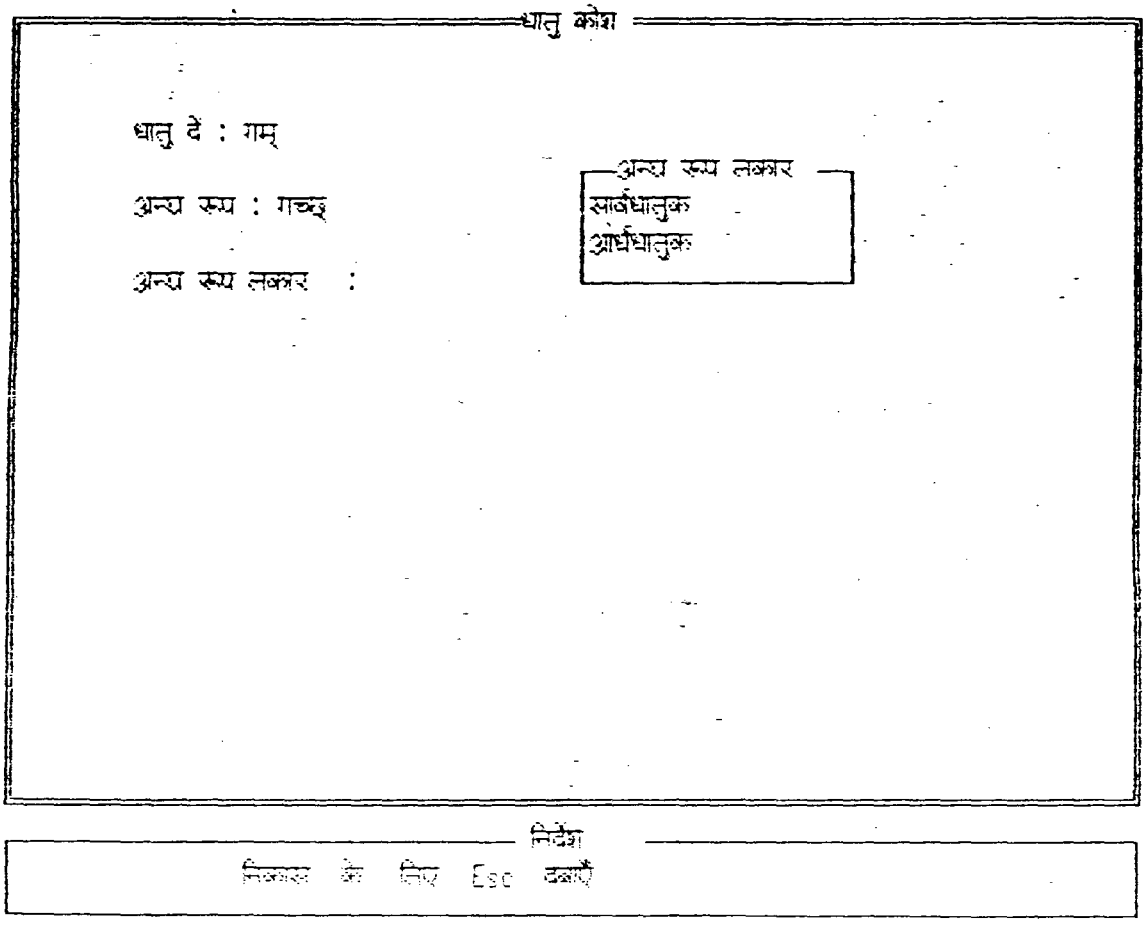
धातु दें : गम्

संदेश

गम् पहले से ही धातुक्लेश में है ।

निर्देश

निकास के लिए Esc दबाएँ


धातु कोश
धातु दे : गम्
अन्य रूप : गच्छ्
अन्य रूप लकार :
अन्य रूप लकार
सार्वधातुक
आर्धधातुक
निर्देश
निकलने के लिए Esc दबाएँ

धातु क्लेश

संदेश

धातुक्लेश में डालें ? हाँ/न के लिए F1/F2 कुंजी दबाएं

धातु दें : गम्

अन्य रूप : गच्छ्

अन्य रूप लकार : सार्वधातुक

गण : भ्वादि

पद : परस्मैपदी

अनिवार्य कारक : ["कर्ता"]

वैकल्पिक कारक : ["सम्प्रदान","अपादान","करण","कर्म","अधिकरण"]

निर्देश

निकास के लिए Esc दबाएँ

धातु क्लेश

धातु : गम्

धातु = गम्
अन्य रूप = गच्छ्
लकार = सार्वधातुक
गण = भ्वादि
पद = परस्मैपदी
अनिवार्य कारक = ["कर्ता"]
वैकल्पिक कारक = ["सम्प्रदान","अपादान","करण","कर्म","अधिकरण"]

निर्देश

निकास के लिए Esc दबाएँ

धातु कोश

धातु : वच्

संदेश

वच् धातुकोश मे विद्यमान नही है

निर्देश

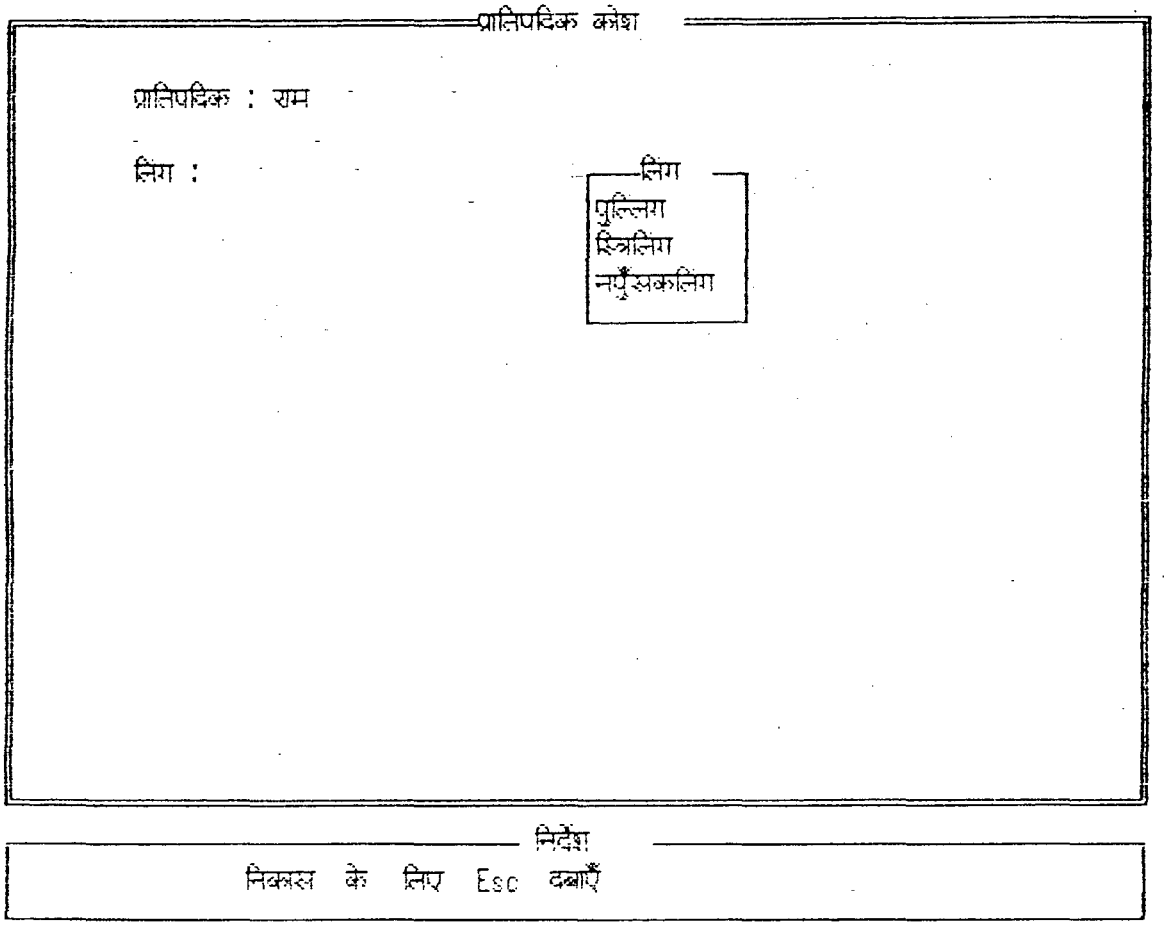
निकास के लिए Esc दबाएँ

प्रातिपदिक कोश

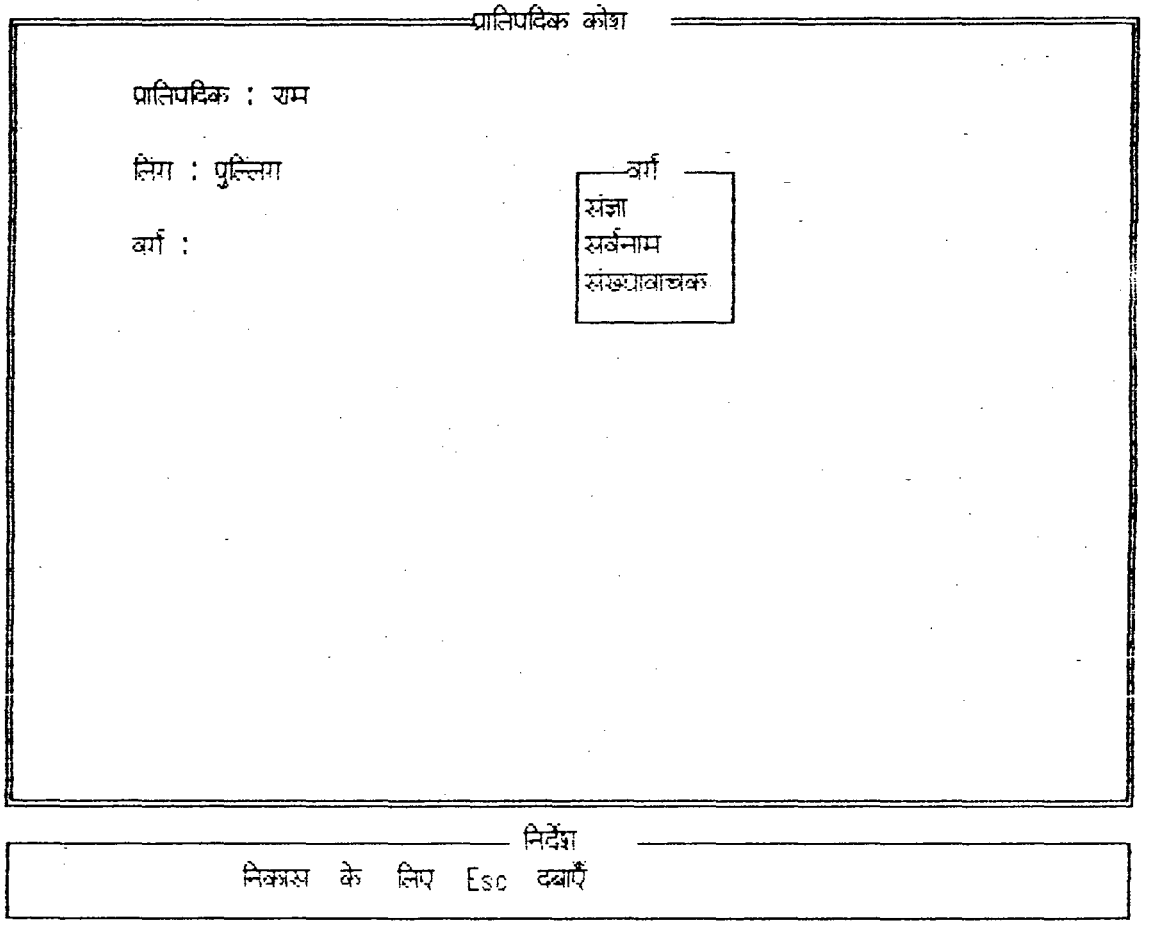
प्रातिपदिक :

निर्देश

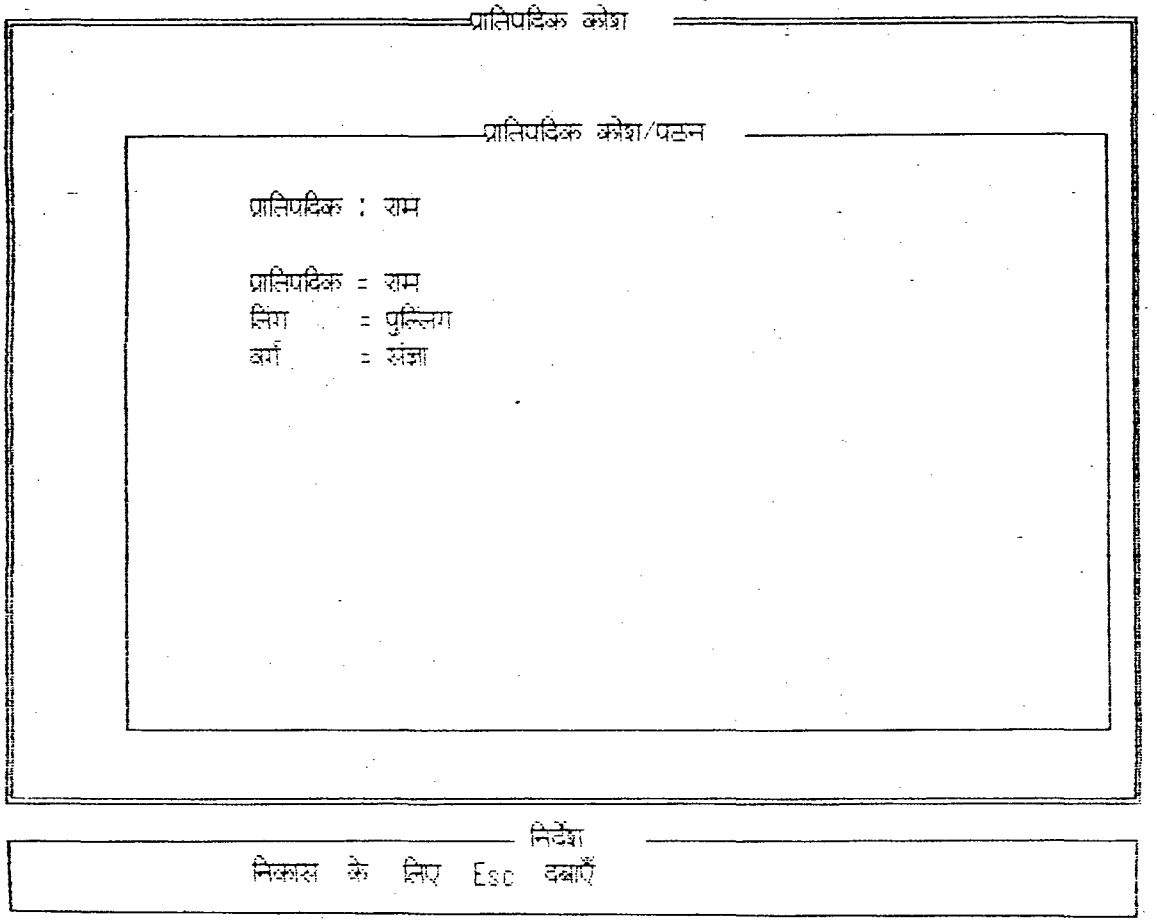
निकास के लिए Esc दबाएँ

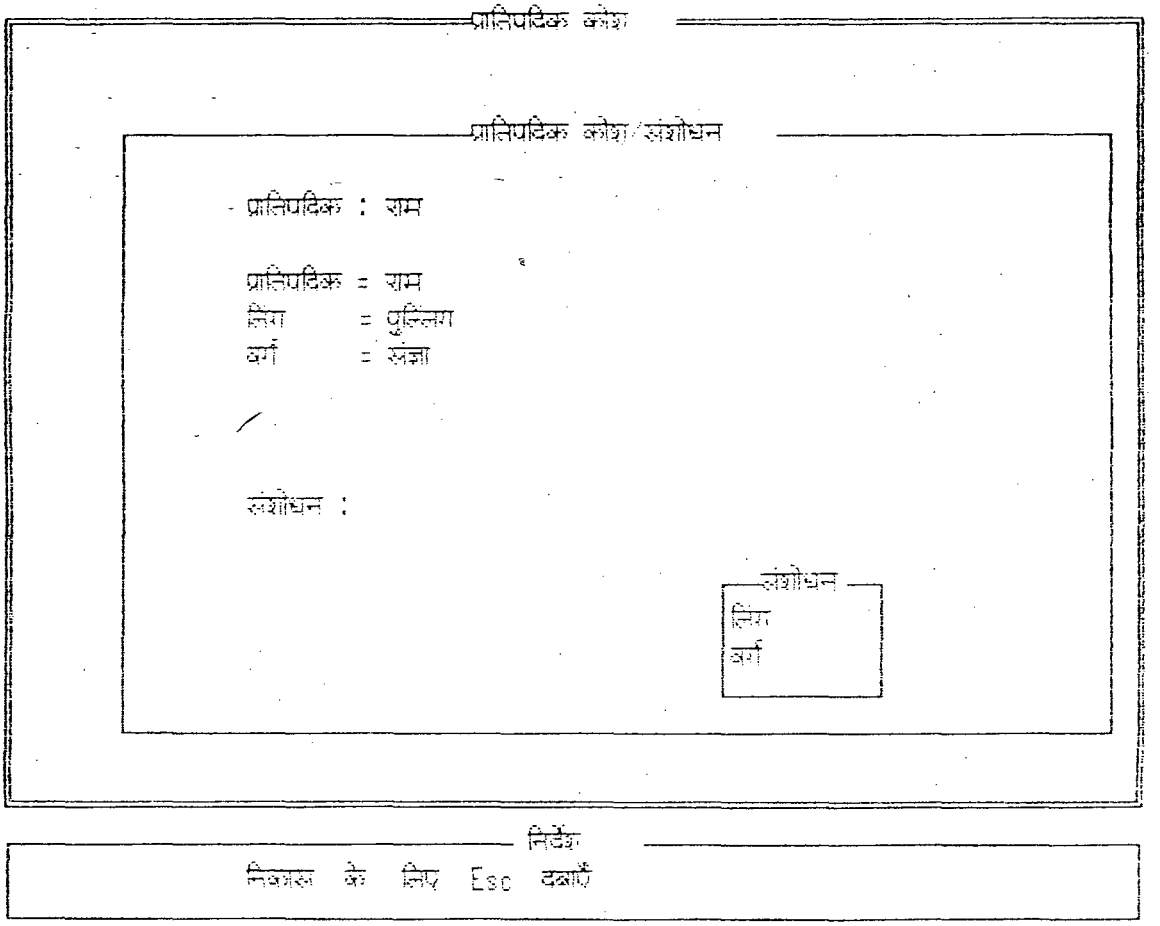

प्रातिपदिक क्लेश
प्रातिपदिक : राम
लिंग :
लिंग
पुल्लिंग
स्त्रीलिंग
नपुंसकलिंग
निर्देश
निकलने के लिए Esc दबाएँ

A.3.4


प्रातिपदिक कोश
प्रातिपदिक : राम
लिंग : पुल्लिंग
वर्ग :
वर्ग
संज्ञा
सर्वनाम
संख्यावाचक
निर्देश
निकास के लिए Esc दबाएँ

A·3·5


प्रातिपदिक कोश
प्रातिपदिक कोश/पठन
प्रातिपदिक : राम
प्रातिपदिक = राम
लिंग = पुल्लिंग
वर्ग = संज्ञा
निर्देश
निकास के लिए Esc दबाएँ

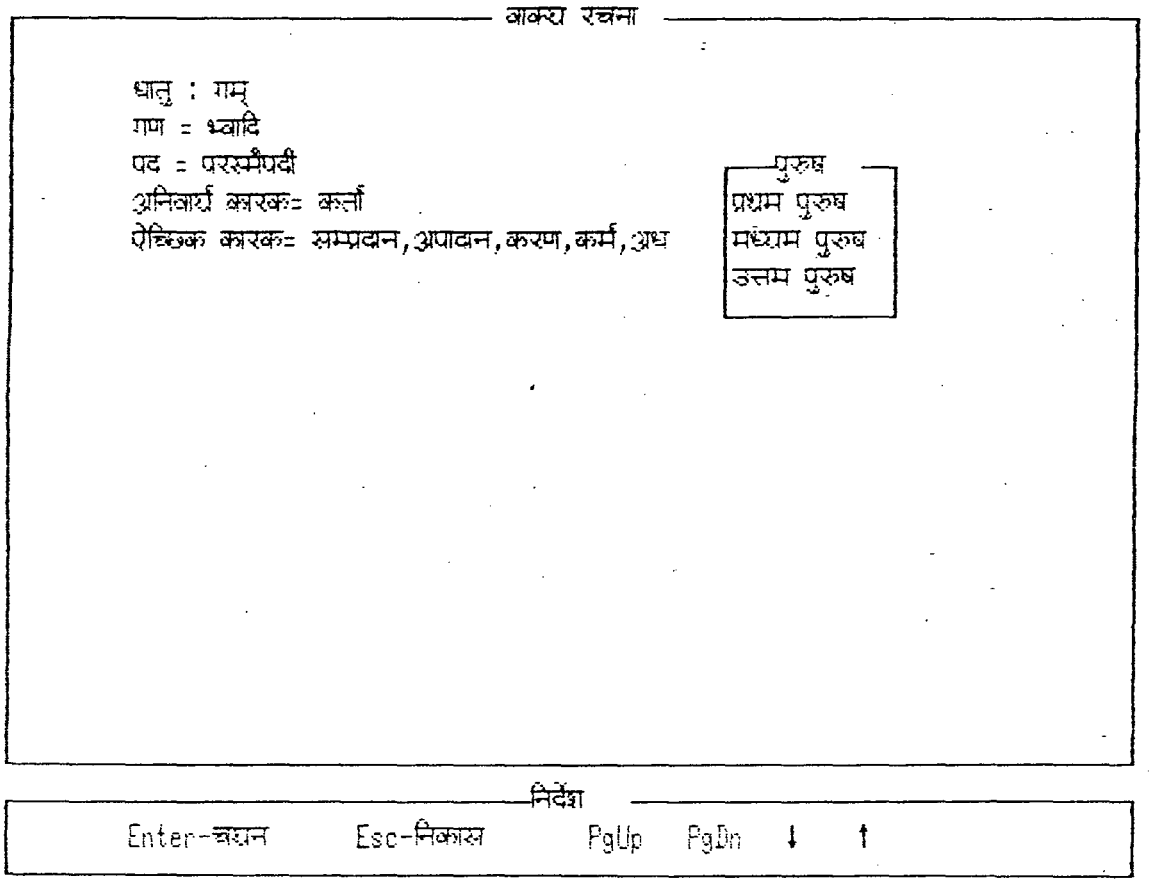

प्रातिपदिक कोश
प्रातिपदिक कोश/संशोधन
प्रातिपदिक : राम
प्रातिपदिक = राम
लिंग = पुल्लिंग
वर्ग = संज्ञा
संशोधन :
संशोधन
लिंग
वर्ग
निर्देश
निकलने के लिए Esc दबाएँ

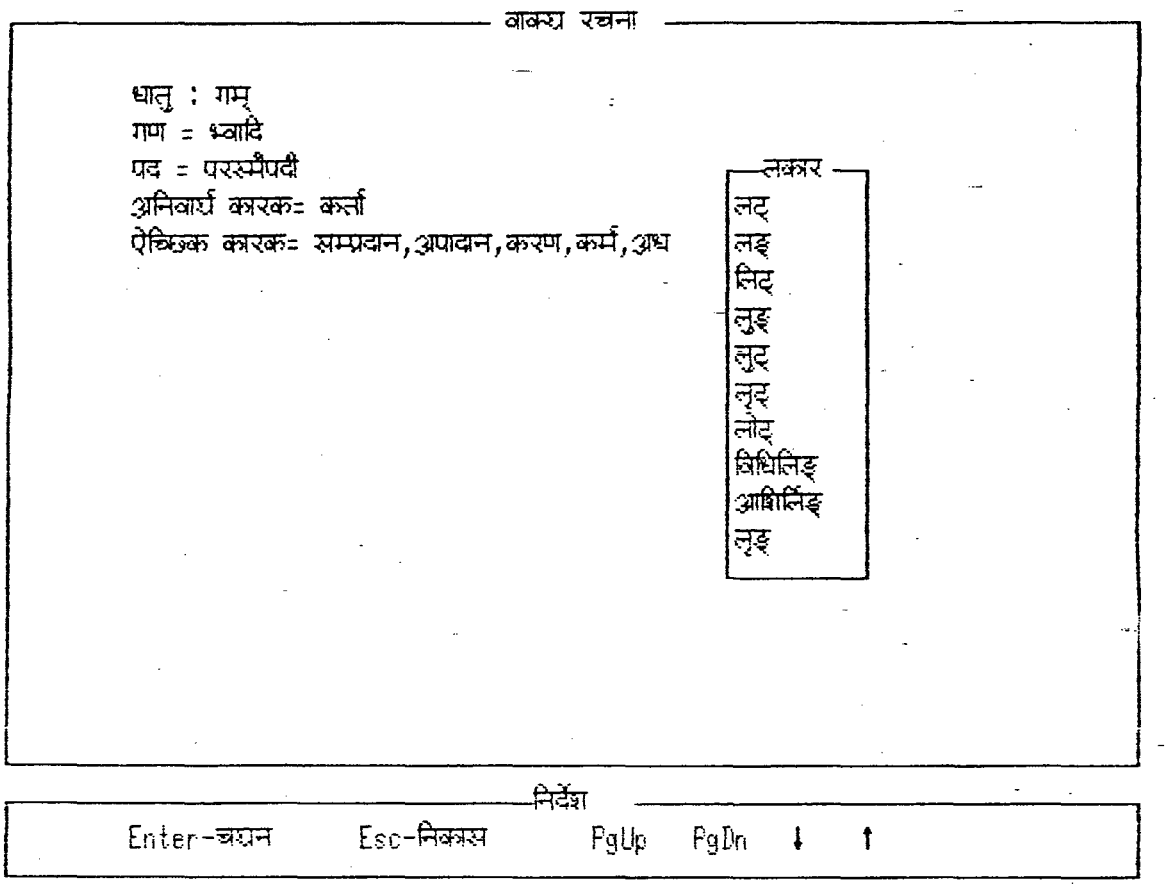
वाक्य रचना

धातु :

निर्देश

ESC-निकास


वाक्य रचना
धातु : गम्
गण = भ्वादि
पद = परस्मैपदी
अनिवार्य कारक= कर्ता
ऐच्छिक कारक= सम्प्रदान, अपादान, करण, कर्म, अध
पुरुष
प्रथम पुरुष
मध्यम पुरुष
उत्तम पुरुष
निर्देश
Enter-चयन
Esc-निकास
PgUp
PgDn
↓
↑

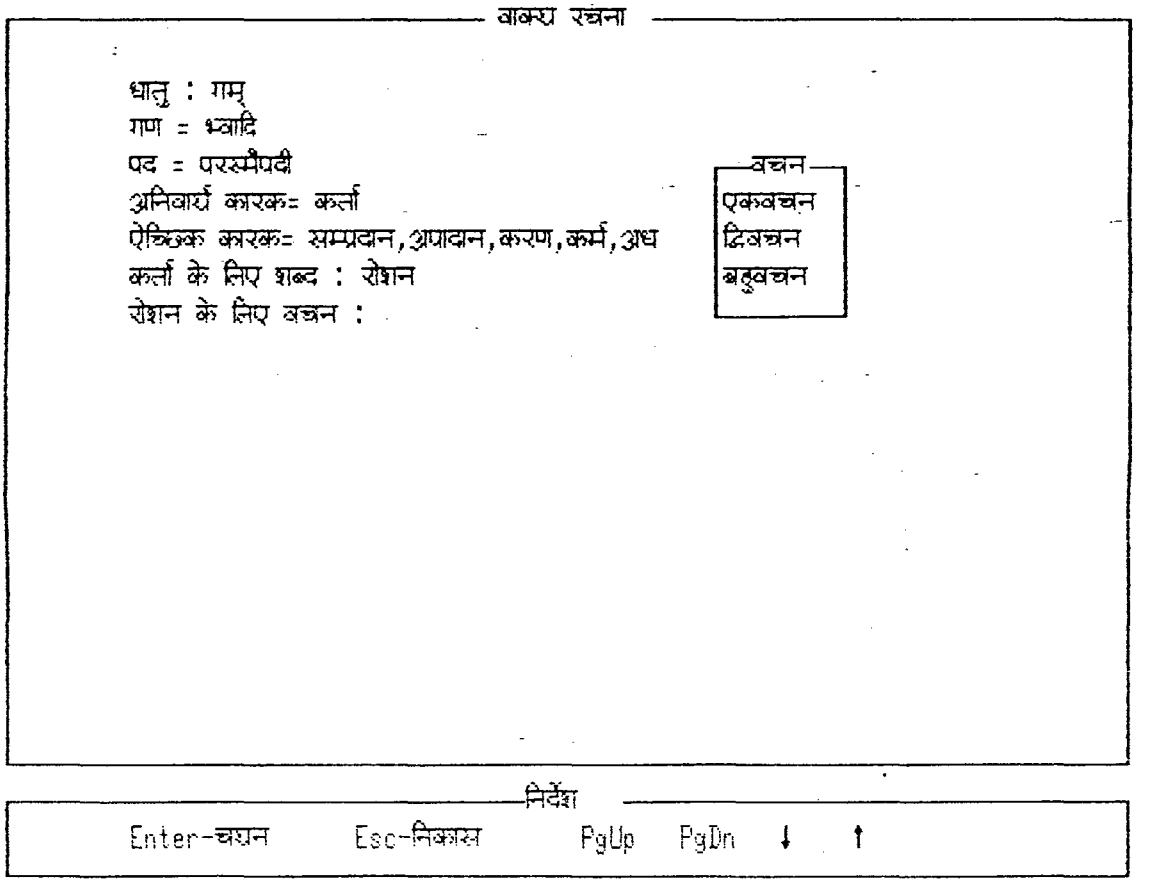

वाक्य रचना
धातु : गम्
गण = भ्वादि
पद = परस्मैपदी
अनिवार्य कारक= कर्ता
ऐच्छिक कारक= सम्प्रदान,अपादान,करण,कर्म,अध
लकार
लट्
लङ्
लिट्
लुङ्
लृट्
लृङ्
लोट्
विधिलिङ्
आशीर्लिङ्
लृङ्
निर्देश
Enter-चयन
Esc-निकास
PgUp
PgDn
↓
↑

वाक्य रचना

धातु : गम्
गण = भ्वादि
पद = परस्मैपदी
अनिवार्य कारक= कर्ता
ऐच्छिक कारक= सम्प्रदान,अपादान,करण,कर्म,अधिकरण
कर्ता के लिए शब्द :

निर्देश

Enter-चयन Esc-निकास PgUp PgDn ↓ ↑


वाक्य रचना
धातु : गम्
गण = भ्वादि
पद = परस्मैपदी
अनिवार्य कारक= कर्ता
ऐच्छिक कारक= सम्प्रदान,अपादान,करण,कर्म,अध
कर्ता के लिए शब्द : रोशन
रोशन के लिए वचन :
वचन
एकवचन
द्विवचन
बहुवचन

निर्देश
Enter-चयन
Esc-निकास
PgUp
PgDn
↓
↑

वाक्य रचना

धातु : गम्
गण = भ्वादि
पद = परस्मैपदी
अनिवार्य कारक= कर्ता
ऐच्छिक कारक= सम्प्रदान,अपादान,करण,कर्म,अध
कर्ता के लिए शब्द : रोशन
रोशन के लिए वचन : एकवचन

वचन
एकवचन
द्विवचन
बहुवचन

सम्प्रदान के लिए शब्द : अध्ययन
अध्ययन के लिए वचन : एकवचन

अपादान के लिए शब्द : ग्राम
ग्राम के लिए वचन : एकवचन

करण के लिए शब्द : शकट
शकट के लिए वचन : एकवचन

कर्म के लिए शब्द : नगर
नगर के लिए वचन :

निर्देश

Enter-चयन Esc-निकास PgUp PgDn ↓ ↑

वाक्य रचना

रोशनः अध्ययनाय ग्रामात् शकटेन नगरम् गच्छति।

निर्देश

Enter-चयन Esc-निकास PgUp PgDn ↓ ↑