Sādhanā, Vol. 19, Part 1, February 1994, pp. 113–127. © Printed in India.

Paninian framework and its application to *Anusaraka*

AKSHAR BHARATI, VINEET CHAITANYA and RAJEEV SANGAL[1]

Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur 208 016, India
[1]E-mail: sangal@iitk.ernet.in

Abstract. The *Paninian** framework proposes *karakas* as semantico-syntactic relations that play a crucial role in mediating between surface form and meaning. The framework accounts for theta-role assignment, active passive, and control in a uniform manner. It has been successfully used in building an extremely fast prototype machine-translation system between two Indian languages. The constraint parser and the generator are designed with information theoretic considerations. Paninian framework is particularly suited to free word order languages. As most human languages are relatively word-order free, the Paninian framework should be explored as a serious contender for such languages. Based on the Paninian theory, the concept of language accessor or *anusaraka* has emerged, which has the potential to overcome the language barrier in India.

Keywords. *Paninian* grammar; *karaka*; *anusaraka*; machine translation; language accessor.

1. Introduction

How is it that natural language is used by speakers to convey information to the hearers? How is it that on hearing an utterance, the hearers are able to get intended information? These are the questions that have intrigued *Paninians*. The goal of the Paninian approach is to construct a theory of human natural language communication that answers these questions. Grammar, a part of such a theory of communication, is a system of rules that establishes a relation between what the speaker decides to say and his utterance, and similarly, what the hearer hears and the meaning he extracts[1].

The main problem that the Paninian approach addresses is how to extract *karaka* relations (which are syntactico-semantic relations) from a sentence. As it is inspired

*In this paper non-English words occur many times and hence are italicized only at first mention.

[1]There is a difference between the goals of the Paninian approach on one hand and generative enterprise on the other. The latter is interested in identifying the innate universal grammar in the mind of every human child by which it so effortlessly and without explicit tutoring acquires language that it is exposed to.

by an inflectionally rich language, it emphasizes the roles of case endings and markers such as post-positions (or prepositions). Position or word order is brought into consideration only when necessary.

In the next section, we pose two problems in Hindi, a 'free' word order language: (1) how to identify semantic relations (or karaka role) from *vibhakti* markers, and (2) how to identify word senses. In § 3, we show how the Paninian theory accounts for the problems. Section 4 discusses computational aspects.

A majority of human languages including Indian and other languages have relatively free word order. In free word order languages, the order of words contains only secondary information such as emphasis etc. Primary information relating to 'gross' meaning (e.g., one that includes semantic relationships) is contained elsewhere. In contrast to these languages, most existing natural language processing systems are based on context free grammars which are basically positional grammars. These are designed for languages in which the position of a constituent in a sentence contains key information. It is important to develop a suitable computational grammar formalism for free word order languages for two reasons:

(1) a suitably designed formalism will be more efficient because it will be able to make use of primary sources of information directly;
(2) such a formalism is also likely to be linguistically more elegant and satisfying. Since it will be able to relate to primary sources of information, the grammar is likely to be more economical and easier to write.

In this paper, we describe such a formalism, called the Paninian framework, that has been successfully applied to Indian languages. The parser and the generator have been designed for the framework based on information theoretic considerations. These are part of a small prototype machine-translation system from Hindi to Telugu. The resulting system is extremely fast, because the grammar makes direct use of vibhakti, the source of primary information in Indian languages, and the parser makes efficient use of grammar while parsing.

2. Indian languages: Some salient features

2.1 *Free word order and vibhakti*

Indian languages have relatively free word order. Many of the constituents of a simple sentence can occur in any order without affecting the gross meaning of the sentence; what is affected is perhaps the emphasis etc. For instance, noun groups in a sentence can come in any order without affecting the theta relationships (or semantic relationship between the verb and the noun groups). Since position or order of occurrence of a noun group does not contain information about the theta roles in a simple sentence, a question can be asked regarding what carries this information. The answer seems to be that post-position markers after nouns (in North Indian languages) or surface case endings of nouns (in South Indian languages) or a mixture of the two at times, play a key role in specifying semantic relationships. We will collectively refer to the post position markers and surface case endings of nouns as vibhakti. Consider the following sentences in Hindi (Bharati *et al* 1990a, 1991a).

A.1 *ra:m mo:han ko: piṭta: hai*
Ram Mohan *-ko:* beat is.
'Ram beats Mohan'.

A.2 *mo:han ko: ra:m piṭta: hai*
Mohan-*ko:* Ram beat is.
'Ram beats Mohan'.

A.3 *mo:han ra:m ko: piṭta: hai*
Mohan Ram -*ko:* beat is.
'Mohan beats Ram'.

A.4 *ra:m ko: mo:han piṭta: hai*
Ram -*ko:* Mohan beat is.
'Mohan beats Ram'.

In A.1 and A.2, Mohan has the same vibhakti (i.e., *parsarg* or post position '*ko:*') and semantic relation with beating. Even though the position of '*ra:m*' and '*mo:han ko:*' are interchanged in A.2, it does not alter the respective semantic relations of Ram and Mohan with the verb. A.3 and A.4 show that semantic relation of Ram is interchanged with that of Mohan by interchanging their vibhakti. So the vibhaktis are crucial in determining the semantic roles. The relative position of the nominal does not seem to be very important for determining semantic relations.

However things are not always straightforward and the following need to be accounted for: A different vibhakti can be used for the same semantic relation with a given verb in a different sentence. For example, in the Hindi sentence B.1 to B.4, although Ram has the same semantic relation with eat (namely, the agent of eat), a different vibhakti is used each time (nil, *ne:*, *ko:*, *se:*, respectively).

B.1 *ra:m pʰal ko: kʰa:ta: hai:*
Ram fruit -*ko:* eats is.
'Ram eats the fruit'.

B.2 *ra:m ne: pʰal kʰa:ya:*
Ram -*ne:* fruit ate.
'Ram ate the fruit'.

B.3 *ra:m ko: pʰal kʰa:na: pada:*
Ram -*ko:* fruit eat had to.
'Ram had to eat the fruit'.

B.4 *ra:m se: pʰal nahi: kʰa:ya: gaya:*
Ram -*se:* fruit not eat could.
'Ram could not eat the fruit'.

2.2 *Head and vibhakti*

Vibhakti for nouns has already been defined earlier. A noun group is a unit containing a noun (or a pronoun, proper name etc.), its vibhakti and possibly the same objectives. The noun is the head and it occurs close to the lexical items that express its vibhakti. In case, the vibhakti is expressed by means of surface case ending, it is incorporated in the head word by morphological process; and in case, it is expressed by means of a post-position marker or parsarg and normally occurs immediately after the head. Sometimes a particle can intervene between the head word and the parsarg, but the parsarg is within a bounded distance (usually just one or two words away) from the head. This property has an important implication for parsing strategy: the vibhakti of a nominal can be identified first even before identifying the noun group (for example, before identifying adjectives or other modifiers).

Vibhakti for verbs can be defined similar to that for the nouns. A head verb may be followed by auxiliary verbs (which may remain as separate words or may combine with the head verb). Such information is collectively called vibhakti for the verb. The vibhakti for verb gives information about tense, aspect and modality (TAM), and is, therefore, also called the TAM label. TAM labels are purely syntactic determined from the verb form and the auxiliary verbs[2].

A verb group consists of the head verb, its vibhakti, and possibly some particles for emphasis, negation etc. Again, the head verb occurs close to the lexical items that express its vibhakti.

3. Paninian theory

Paninian grammar is particularly suited to free word order languages. It makes use of vibhakti information for mapping to semantic relations, and uses position information only secondarily. As the Indian languages have (relatively) free word order and vibhakti, they are eminently suited for description by the Paninian grammar. The Paninian framework was originally designed more than two millenia ago for writing a grammar for Sanskrit; it has been adapted by us to deal with modern Indian languages.

3.1 *Karaka relations*

Example sentences in Hindi from the previous section (in A and B) indicate that there is no straightforward mapping from vibhakti to semantic relations between noun groups and verbs. The key to arriving at an answer is to identify intermediate relations.

The notion of *karaka* (pronounced '*ka:rak*') relation is central to the model. These are semantico-syntactic relations between the verb(s) and the nominals in a sentence. The computational grammar specifies a mapping from the vibhaktis of nominals and the verb(s) in a sentence to karaka relations between them. Similarly, other rules of grammar provide a mapping from karaka relations to semantic relations between the verb(s) and the nominals. Thus, the karaka relations by themselves do not give the semantics. They specify relations which mediate between vibhakti of nominals and verb forms on the one hand and semantic relations on the other (Kiparsky 1982). Figure 1 shows the relationship pictorially.

The *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

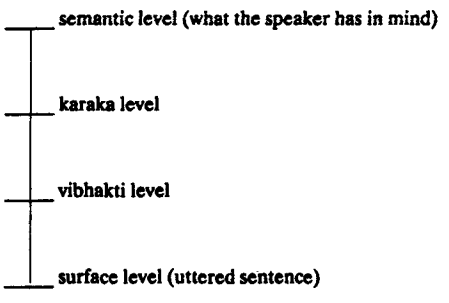

Figure 1. Levels in the Paninian model.

[2]A suitable mapping, though complex, can take us from TAM labels to TAMs.

Karaka	Vibhakti	Presence
Karta	Ø	mandatory
Karma	*ko:* or Ø	mandatory

Figure 2. Default mapping for some karakas.

nominals that are expressed by the speaker. However, it is with respect to the activity implied by the verb.

3.2 *Karaka to vibhakti mapping*

We are now ready to discuss the mapping from karaka level to vibhakti level. This will allow us to map a representation of a sentence at the karaka level to a representation at the vibhakti level (or the inverse). We will not have anything further to say, in this paper, on the mapping from semantic level to karaka level.

The most important insight regarding the karaka–vibhakti mapping is that it depends on the TAM label of the verb. There is a default mapping from karakas to vibhaktis of nominals in a sentence. The default mapping holds if the verb in question (whose karaka relations are given) has a particular TAM label called basic. If the verb has some other TAM label, the default is transformed depending on the TAM label[3].

In Hindi for instance, the basic TAM label is *ta:_hai* (which roughly stands for the present indefinite). The default mapping for two of the karakas is given in figure 2. This explains the vibhaktis in sentences A.1 to A.4. As Ram is the agent in A.1 and A.2, and Mohan in A.3 and A.4 and the agent is the most independent for the action 'beat', it is expressed by means of the karta karaka; and the remaining nominal by *karma* karaka. For the karta karaka, 0 vibhakti is used with *ta:_ hai* TAM label in A.1 to A.4 as explained in figure 2.

Figure 3 gives some transformation rules for the default mapping for Hindi. It explains the vibhakti in sentences B.1 to B.4 (assuming that Ram is the karta and *p^{h}al* 'fruit' is the karma). As explained by figure 3, karta takes *'ne:'* vibhakti in B.2 because of TAM label *'ya:'* (in the main verb group *k^{h}a:ya:*), *'ko:'* in B.3 because of TAM label *na:_pada:* (in *k^{h}a:na: pada:*), and *'se:'* in B.4 because of TAM label *ya:_gaya:* (in *k^{h}a:ya gaya:*).

3.3 *Control*

A major support for the theory comes from complex sentences, that is sentences containing more than one verb group. We first introduce the problem and then describe how the theory provides an answer. Consider the following sentences in Hindi.

G.1 *ra:m p^{h}al k^{h}a:kar mo:han ko: bula:ta: hai:*
Ram fruit having-eaten Mohan *-ko:* calls is
'Having eaten fruit, Ram calls Mohan'

TAM label	transformed vibhakti for karta
ya:	*ne:*
na:_pada:	*ko:*
ya:_gaya:	*se:* or *dwa:ra:* (and karta is optional)

Figure 3. Transformation rules.

[3] What karaka relations are permissible for a verb, obviously, depend on the particular verb. Not all verbs will take all possible karaka relations.

TAM label	transformation
kar	Karta must not be present
	Karma is optional
na:	Karta and karma are optional
ta:_hua:	Karta and karma are optional

Figure 4. More transformation rules.

G.2 *ra:m ne: pʰal ka:ṭkar kʰa:ya:*
Ram *ne:* fruit having-cut ate
'Ram ate having cut the fruit'

G.3 *pʰal ka:tne: ke: liye: usne: ca:ku: liya:*
fruit to-cut *-ke:-liye:* he:*-ne:* knife took
'To cut fruit, he took a knife'

In G.1, Ram is the karta of both the verbs: *kʰa:* 'eat' and *bula:* 'call'. However, it occurs only once. The problem is to identify which verb will control its vibhakti. In G.2, karta Ram and the karma *pʰal* 'fruit' both are shared by the two verbs *ka:t* 'cut' and *kʰa:* 'eat'. In G.3, the karta *usne:* 'he' is shared between the two verbs, and *'ca:ku:'* 'knife' the karma karaka of *le:* 'take' is the karana (instrumental) karaka of *'ka:t'* 'cut'.

The observation that the matrix verb rather than the intermediate verb controls the vibhakti of the shared nominal is true in the above sentences. The theory we will outline to elaborate on this theme will have two parts. The first part gives the karaka to vibhakti mapping as usual, the second part of the theory (not described in this paper) identifies shared karakas. See Bharati *et al* (1994) for details.

The intermediate verbs have their TAM labels just like other verbs. For example, *kara* is the TAM label of *kʰa:* 'eat' in G.1, and the *na:* is the TAM label of *ka:t:* 'cut' in G.3. As usual, these TAM labels have transformation rules that operate and modify the default karaka to vibhakti mapping. In particular, the suggested transformation rules for the two labels are given in figure 4. The transformation rule with kara in figure 4 says that karta of the verb with TAM label kara must not be present in the sentence and the karma is optionally present.

By these rules, the intermediate verb *kʰa:* 'eat' in G.1 does not have (independent) karta karaka present in the sentence. Ram is the karta of the matrix or the main verb *bula:* 'call'. *pʰal* 'fruit' is the karma of *kʰa:*. All these are accommodated by the above transformation rule for 'kara'. The karta of *kʰa:* in G.1 and similarly in other cases, can be obtained by sharing rules, which are not described here.

4. Information-theoretic parser

The Paninian theory outlined above can be used for building a parser. (See Bharati *et al* (1990b) for a discussion on other approaches to parsing.) Parsing is the reverse of generation, where given a sentence a suitable semantic structure is to be assigned to it. If we build a parser based on the Paninian theory, we have to obtain a representation of a given sentence at the vibhakti level, using which we must obtain a representation at the karaka level, and finally, a representation at the semantic (or mental) level. (Again, in this paper we will not talk about the semantic level, and will focus on mapping from the vibhakti level to the karaka level.)

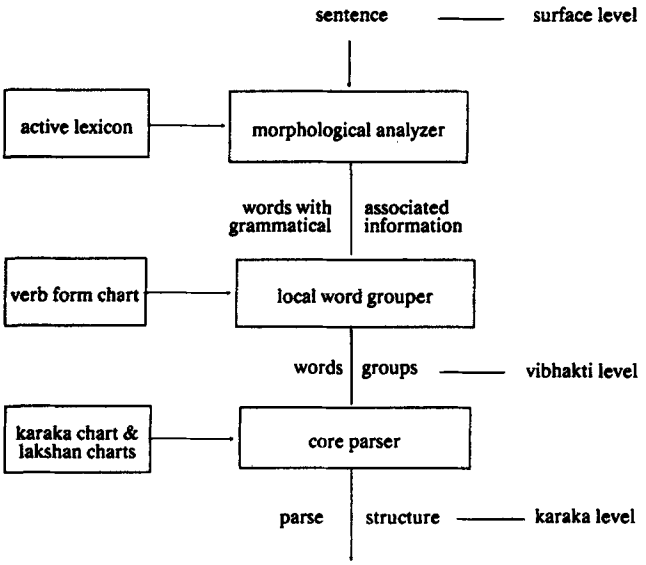

Figure 5. Structure of the parser.

It turns out that the Paninian theory is extremely suitable from the computational viewpoint. It can be used in a natural manner for structuring a parser which is extremely efficient.

It is fairly obvious that one part of the parser must take care of morphology. For each word in the input sentence, a dictionary or a lexicon needs to be looked up, and associated grammatical information needs to be retrieved. The words have to be grouped together yielding nominals, verbals etc. Finally, the karaka relations among the elements have to be identified. This is shown in figure 5.

4.1 *Information-theoretic approach*

The parser is based on information-theoretic considerations where at each stage of processing, just the right amount of information is extracted. At the morphological analysis stage, information available from word forms is obtained. For example, information about gender, number, person is obtained (wherever possible) from the nouns. On the other hand, for verbs in Hindi, gender, number, person (*gnp*) and part of TAM is obtained from the words. In case of Telugu verbs the complete TAM label, besides *gnp* is obtained as well.

In the local word grouping stage, words combine into groups (noun groups and verb groups) based on local information. The groups are formed with minimal computational effort (finite-state machine model) as only local information is used. On the other hand, local word grouping brings together just the right information (vibhakti for nouns, TAM labels for verbs) that is needed for the next stage of processing (that is, for karaka assignment). It does not attempt to distinguish all the fine shades of meaning, for example, temporal structure or modality. For one thing, such information cannot easily be determined at this stage of processing, Secondly, it is not needed for the next stage.

As mentioned earlier, the output of the local word grouper roughly corresponds to the vibhakti level. However, in those few cases where there is ambiguity in identifying the local word groups, which cannot be resolved at that level, the decision

is postponed. For example, in Hindi, a word that can be both a noun and an adjective, causes ambiguity in forming a local word group with its succeeding noun. The choice can only be made later during karaka assignment using karaka charts. As a result, in our information-theoretic parser, such a choice is delayed to the point when it can be made. Similar is the case with a noun that is followed by a marked *ka:*, *ke:* or *ki:* and succeeded by a noun.

After the local word grouping stage, there is karaka assignment and lexical disambiguation stage. This is done at this stage because the necessary information (vibhakti) for doing the above is available. Other phenomena such as quantifiers and anaphora are not handled because information for resolving them is not available.

This approach is consistent with the Indian grammatical analysis where meaning is extracted in several layers with increasing precision.

4.2 *Core parser*

Morphological analyser and local word grouper shown in figure 5, have been described elsewhere (Bhanumathi 1989; Bharati *et al* 1991b). Here we discuss the core parser.

Given the local word groups in a sentence, the task of the core parser is two fold:
(1) to identify karaka relations among word groups, and
(2) to identify senses of words.

The first task requires knowledge of karaka-vibhakti mapping, optionality of karakas, and transformation rules. The second task requires *lakshan* charts for nouns and verbs, to be discussed later.

A structure called karaka chart stores information about karaka-vibhakti mapping and optionality of karakas. Initially, the default mapping is loaded into it for a given verb group in the sentence. Transformations are performed using the TAM label. There is a separate karaka chart for each verb group in the sentence being processed. Information about semantic types of fillers of karaka roles is also available. But such information is limited to that necessary for removing ambiguity, if any, in karaka assignment. In other words, for a given verb, when karaka-vibhakti mapping is not sufficient for producing an unambiguous parse, semantic types are included. The semantic types so included have the sole-purpose of karaka diambiguation. This keeps the number of semantic types under control, and serves as a guiding philosophy for what semantic types to include. Figure 6 shows the starting semantic type hierarchy which is sufficient for a major part of language. An example karaka chart for *kʰa:* 'eat' is given in figure 7. It shows for each of the karakas its necessity (mandatory, desirable, or optional), vibhakti, and semantic type. These are called karaka restrictions in a karaka chart.

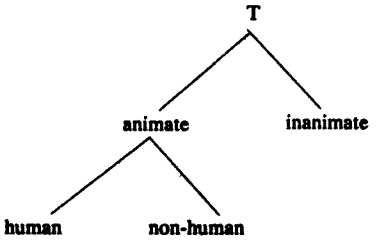

Figure 6. Semantic type hierarchy.

karaka	necessity	vibhakti	semantic type
karta	m (mandatory)	Ø	animate
karma	m	Ø - or - *ko:*	--

Figure 7. Karaka chart for *kʰa:*.

For a given sentence after the word groups have been formed, karaka charts for the verb groups are created and each of the noun groups is tested against the karaka restrictions in each karaka chart (provided the noun group is to the left of the verb group whose karaka chart is being tested). When testing a noun group against a karaka restriction of a verb group, vibhakti information and semantic type are checked, and if found satisfactory, the noun group becomes a candidate for the karaka of the verb group. This can be shown in the form of a constraint graph. Nodes of the graph are the word groups and there is an arc from a verb group to a noun group labelled by a karaka, if the noun group satisfies the karaka restriction in the karaka chart of the verb group. (There is an arc from one verb group to another, if the karaka chart of the former has a karaka restriction with semantic type as action). The verb groups are called demand groups as they make demands about their karakas, and the noun groups are called source groups because they satisfy demands. (A verb group can be a source group as well when it satisfies the demand of another verb group. This however does not affect its status as a demand group as well.)

As an example, consider a sentence containing the verb *kʰa:* 'eat' with its word groups marked:

bacca: ke:le: ko: kʰa:ta: hai:
child banana *-ko:* eats
'The child eats the banana'.

Its constraint graph is shown in figure 8. It also happens to be the solution graph. Consider another sentence where the constraint graph is different from the solution graph.

bacca: ke:la: kʰa:ta: hai:
child banana eats
'The child eats a banana'.

Here, both the nouns qualify, to be karta and karma. In such a situation, the parser produces both parses; however, the first parse is one in which the animate entity, namely *bacca:* (child), is the karta.

4.2a *Constraints*: A parse is a sub-graph of the constraint graph containing all the nodes and satisfying the following conditions (Bharati & Sangal 1990):

(1) for each of the mandatory karakas in karaka chart for each demand group, there should be exactly one outgoing edge from the demand group labelled by the karaka;
(2) for each of the desirable or optional karakas in a karaka chart for each demand group, there should be at most one outgoing edge from the demand group labelled by the karaka;

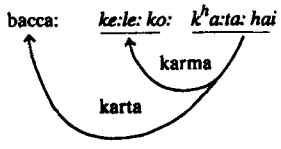

Figure 8. Constraint graph and solution graph.

(3) there should be exactly one incoming arc into each source group.

If there are several sub-graphs for a constraint graph satisfying the above conditions, it means that there are multiple parses and the sentence is ambiguous. If no sub-graph satisfies the above constraints, the sentence does not have a parse.

4.2b *Constraint parser*: Currently, a parse is obtained from the constraint graph using integer programming. A constraint graph is converted into an integer programming problem by introducing a variable x_{ijk} for an arc from node i to j labelled by karaka k in the constraint graph such that for every arc there is a variable. The variables take their values as 0 or 1. A parse is an assignment of 1 to those variables whose corresponding arcs are in the parse sub-graph, and 0 to those that are not. Equality and inequality constraints in integer programming problem can be obtained from the conditions listed earlier, as follows, respectively:

(1) for each demand group i, for each of its mandatory karakas k, the following equalities must hold:

$$\sum_j x_{ijk} = 1;$$

(2) for each demand group i, for each of its optional or desirable karakas k, the following inequalities must hold:

$$\sum_j x_{ijk} < 1;$$

(3) for each of the source groups j, the following equalities must hold:

$$\sum_{ik} x_{ijk} = 1.$$

The cost function to be minimized may be given as the sum of all the variables, in which case it does not show any preference for any of the parses.

It is possible, however, to put in preferences by suitably varying the cost function. If we use such a cost function, we will get a parse that has a minimum cost with respect to the cost function. The integer programming system can be so setup that we can ask for the next solution, in which case, we will get another parse with the same or higher cost. This can be repeated to obtain all the possible parses.

Some reasonable preferences which can be incorporated in the cost function are as follows: All else being equal,

(1) karta has the following preferences in descending order: human, non-human, inanimate (animacy preference);
(2) a source group is close to the demand group with which it has a relationship (closeness preference);
(3) karta occurs before karma in a sentence (leftness preference)[4].

[4]To the above list can be added a host of conditions dealing with anaphora, movement, garden-path sentences etc.

4.2c *Lakshan charts for sense disambiguation*: The second major task to be accomplished by the core parser is disambiguation of word senses. This requires the preparation of *lakshan* charts (or discrimination nets) for nouns and verbs.

A lakshan chart for a verb allows us to identify the sense of the verb in a sentence given its parse. Lakshan charts make use of the karakas of the verb in the sentence, for determining the verb sense. For example, in the sentence H.1 and H.2:

H.1 *kisa:n kʰe:t ko: jo:ʈta hai:*
farmer land *-ko:* ploughs
"The farmer ploughs the land"

H.2 *kisa:n ga:di: ko: jo:ʈta: hai:*
farmer cart *-ko:* attaches
"The farmer attaches the cart"

H.3 *kisa:n kʰe:t ko: ka:ʈta: hai:*
farmer crops *-ko:* harvests
"The farmer harvests the crops"

The verb *'jo:ʈta: hai:'* is used in two different senses: plough or attach. Lakshan chart for *'jo:ʈ'* would allow us to select the appropriate sense of *jo:ʈ* by testing whether its karma is land or cart etc. Again, it is designed from an information theoretic point of view. The available information is used in an economical and efficient manner in deducing the right amount of new information.

A verb lakshan chart for a verb is prepared by linguists and language experts by looking up different senses of the verb with the help of conventional dictionaries. The chart builder must select features carefully that would allow verb sense to be obtained.

Noun lakshan charts help disambiguate senses of nouns in a sentence. They make use of the parse structure (i.e., karaka relations) and the verb sense. For example, in sentences H.1 and H.3, the sense of *kʰe:t* is land and crop, respectively. However, depending on the verb of which it is a karma the appropriate sense in selected.

Preparation of lakshan charts is a laborious process but is essential for fully-automatic high-quality machine translation. Linguistic theories are generally silent on the issue of word sense disambiguation.

5. An application: *Anusaraka* – a translation aid

The above theory can be used for building translation systems among languages. For translating from a source language to a target language, we need a parser for the source language and a generator for the target language (see figure 9). This is the interlingua approach. The choice of intermediate representation is important. For a machine translation (MT) system based on the Paninian theory, the intermediate representation is in terms of verb and noun concepts (i.e. word senses) and karaka relations. For Indian languages, the quality of translation is fairly good using karaka relations. Further analysis can improve the quality of translation.

Preparation of language data particularly karaka charts and lakshan charts is a major task in building an MT system. It requires several man-years of effort per language. It turns out that one can talk about the concept of *anusaraka*.

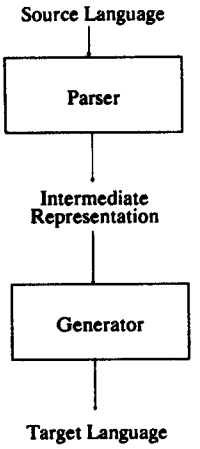

Figure 9. A machine translation system.

5.1 *What is language accessor*

Anusaraka[5] or Language Accessor (LA) allows a reader who knows one language to access or follow text in another language. It produces output which is comprehensible to the reader, without worrying about it grammatically. LA is the result of a new way of looking at the problem: Instead of considering the problem as that of translation, it views the problem to be that of accessing information in another language with possibly some effort by the reader. Such a system for Indian languages, for example, does "simple" analysis (typically upto vibhakti level) of the source language sentences, and then generates sentences in the target language. The generated sentence need not be grammatical, however; it must be such that the reader can follow the meaning of the source sentence or text with little or some effort. Because of the relatively simple analysis done by the system, it is extremely robust towards improper input: ungrammatical sentences, sentence fragments, texts with ellipses, and other real life texts[6]. See Narayana (1994) for technical details.

5.2 *Why LA*

LA addresses several problems at once.

(1) It makes a practical system available today without waiting for several years. Based on present dictionaries and other linguistic resources, it provides access to languages: a person can follow the meaning of a text in another language with some effort.
(2) The work done in building LA will not go waste after high quality fully automatic machine translation (MT) systems are built. In fact, this work forms an essential

[5] *Anusaarak* in Hindi literally means "follower". It is different from "*anuvaadak*" or translator. Anusaarak allows a Hindi reader, for example, to read Hindi following Kannada or "Kannada *anusaar* Hindi".

[6] It can be argued that there is no single "high quality" translation of a document. Quality or rather suitability of a translation depends on the purpose and the audience. For example, good translation of a document would be different for children and adults. Availability of Machine Translation or Language Accessor software and personal computers makes it possible to tailor the translator to the reader.

part of the future systems. Since the LA model is a part of the overall model for MT system, it will continue to develop and improve incrementally. It will not be a "dead-end" system.

(3) It will function as an important tool for the linguist. The LA system will assist in the development of computational lexicons because it will throw up problem cases for comparative analysis.

Thus, LA delivers something practical today; the work to be done for building it, needs to be done in any case for MT; and its availability will help in the remaining work towards development of MT systems of the future.

LA brings about several changes in goals and design criteria when compared with conventional MT systems. Grammaticality reduces in importance, its place is taken by comprehensibility. It emphasizes robustness. (The system should never fail on any imput!) It also assumes the availability of a personal computer to the person reading the output (unlike MT systems which assume that the translator uses the system to produce output for end users). It emphasizes the notion that multiple translations might be produced depending on the audience etc.

While LA and conventional MT systems might seem different, they are complementary. A practical system would also have an MT system besides the LA system in the background. Whenever possible MT would be tried, and in case of failure of the MT system to analyse the input etc., LA would be used to provide an answer. In this sense a practical LA system would keep improving as computational lexicons are prepared and MT systems become available.

5.3 *Applications*

LA can be seen as a core technology which can be put to varied uses. Like the internal combustion engine which can draw water, or move a car, or fly an aeroplane depending on how it used, LA can also be used for different applications. It can assist a translator in the task of translation, provide direct access to books and printed matter to a reader in another language, provide access to documents in a multilingual setting within an organization such as the government, tourist assistance office etc.

LA can also be viewed as a telescope for the linguist. Just as the telescope allows the astronomer to see farther and more clearly, LA allows the linguist to observe phenomena in different languages at first-hand, rapidly and easily. It allows generation of larger masses of data and rapid testing of new theories. Sometimes users are able to get the flavour of the original sentence by looking at its "literal" translation. LA can provide such a flavour.

5.4 *Some long term impact on education*

Several long term impacts are obvious from the applications outlined above. The biggest impact would be the breaking of language barriers using LA. Here we discuss the possible impact on education and languages.

LA research is sharply focussed on those aspects of languages (in particular, of Indian languages) by which essential communication is possible. Thus, it tries to use those features of a set of languages which are readily transferable (with minimal analysis). It also identifies features which do not transfer well and either avoids using them or provides help when they occur. This research can lead to the identification

of a core which is common across Indian languages. This core can be taught in schools, which will allow people to pick any Indian language with ease. When LA systems proliferate, they will encourage learning of this core (as well as of the differences between particular languages that an individual user uses). It may turn out that with the widespread learning of the core, LA promotes development of languages closer to each other.

5.5 *Tasks in developing LA*

Development of LA requires joint work by linguists and language experts, and computer scientists. The former two will work on preparation of suitable electronic dictionaries for LA (based on existing dictionaries meant for human beings), creating mappings between tense aspect modality labels, vibhaktis etc. Work will also be needed in lexical semantics: analysing relationships among different senses of word and how they arise, and in discovering universals about them. Also required is a comparative study of language constructions and phenomena, and how they map onto each other across languages.

Computer scientists will have to work on the basic LA software and suitable interfaces. The task of the basic software will be to analyse and generate text. It will include a morph package (morphological analyser and generator), local word grouper and splitter, routines for accessing dictionaries, resolving lexical category and word sense clashes etc. There is need for coming up with an appropriate architecture for the system. Suitable interfaces are needed for two categories of people: for the users of LA for accessing a document in another language, and for the linguists who wish to use it as a tool for doing comparative analysis of languages or to look at phenomena in other languages.

Finally, there are certain infrastructural needs. First, for LA to be useful, documents need to be available in electronic form. Today, large numbers of documents, including many books, and perhaps all newspapers in Indian languages are composed by the computer. As a result, the text is already available in machine readable form. There is need to create a system for its systematic distribution. Second, there is need to create computer facilities in libraries for example, on which LA software can be run. The community of users can contribute (and get gainful employment) by post-editing the output generated by the LA system, and producing translations for use by others. These can then also see the print media.

6. Conclusion and future work

We have described the Paninian framework as adapted to modern Indian languages. It turns out to be elegant as well as natural and efficient. It is elegant because it is able to handle diverse phenomena like karaka assignment, active-passive, and control in a unified manner. It is efficient because the constraint parser which arises very naturally using the framework is extremely fast.

Finally, we talk about the concept of anusaraka. It makes possible the building of translation aids extremely fast, at least for Indian languages. Currently, a 30,000 word anusaarak from Kannada to Hindi has been built and tested in our laboratory.

We gratefully acknowledge the help received from Dr K V Ramakrishnamacharyulu of Rashtriya Sanskrit Vidyapeetham, Tirupati in development of the theory. Several people have worked on the implementation of the core parser: Sivasubramanian, B Srinivas, P V Ravisankar on the earlier version, and Jayvant Anantpur, Vasudev Verma, Amba Kulkarni on the current version. Mr V N Narayana has been working on the Kannada–Hindi anusaraka.

References

Bhanumati B 1989 An approach to machine translation among Indian languages, Tech. Rep. TRCS-89-90, Dept. Comput. Sci Eng., Indian Institute of Technology, Kanpur

Bharati A, Chaitanya V, Sangal R 1990a A computational framework for Indian languages. Course Notes for Intensive Course in NLP, Vol. 1, Tech. Rep. TRCS-90-100, Dept. Comput. Sci. Eng., Indian Institute of Technology, Kanpur

Bharati A, Chaitanya V, Sangal R 1991a A computational grammar for Indian languages processing. *J. Indian Linguistics* 52(1–4): 91–103

Bharati A, Chaitanya V, Sangal R 1991b Local word grouping and its relevance to Indian languages. In *Frontiers in knowledge based computing (KBCS90)* (eds) V P Bhatkar, K M Rege (New Delhi: Narosa) pp. 277–296

Bharati A, Chaitanya V, Sangal R 1994 *Natural language processing: A Paninian perspective* (New Delhi: Prentice Hall of India) (to be published)

Bharati A, Sangal R 1990 A karaka based approach to parsing of Indian languages. *Proc. Int. Conf. Comput. Languages – COLING 90* (Helsinki: Assoc. Comput. Linguistics)

Bharati A, Sangal R, Chaitanya V 1990b Natural language processing, complexity theory and logic. In *Foundations of software technology and theoretical computer science 10. Lecture Notes in Computer Science* – 472 (eds) K V Nori, C E Veni Madhavan (Berlin: Springer Verlag)

Kiparsky P 1982 *Some theoretical problems in Panini's grammar* (Poona: Bhandakar Oriental Research Institute)

Narayana V N 1994 *Anusaraka: A device to overcome the language barrier*, Ph D thesis, Dept. Comput. Sci. Eng., Indian Institute of Technology, Kanpur (submitted)