Parsing Free Word Order Languages in the Paninian Framework

Akshar Bharati
Rajeev Sangal
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Kanpur 208016 India
Internet: sangal@iitk.ernet.in

Abstract

There is a need to develop a suitable computational grammar formalism for free word order languages for two reasons: First, a suitably designed formalism is likely to be more efficient. Second, such a formalism is also likely to be linguistically more elegant and satisfying. In this paper, we describe such a formalism, called the Paninian framework, that has been successfully applied to Indian languages.

This paper shows that the Paninian framework applied to modern Indian languages gives an elegant account of the relation between surface form (vibhakti) and semantic (karaka) roles. The mapping is elegant and compact. The same basic account also explains active-passives and complex sentences. This suggests that the solution is not just adhoc but has a deeper underlying unity.

A constraint based parser is described for the framework. The constraints problem reduces to bipartite graph matching problem because of the nature of constraints. Efficient solutions are known for these problems.

It is interesting to observe that such a parser (designed for free word order languages) compares well in asymptotic time complexity with the parser for context free grammars (CFGs) which are basically designed for positional languages.

1 Introduction

A majority of human languages including Indian and other languages have relatively free word order. In free word order languages, 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. Most existing computational grammars are based on context free grammars which are basically positional grammars. 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. [1] It uses the notion of karaka relations between verbs and nouns in a sentence. The notion of karaka relations is central to the Paninian model. The karaka relations are syntactico-semantic (or semantico-syntactic) relations between the verbals and other related constituents in a sentence. They by themselves do not give the semantics. Instead they specify relations which mediate between vibhakti of nominals and verb forms on one hand and semantic relations on the other (Kiparsky, 1982) (Cardona (1976), (1988)). See Fig. 1. Two of the important karakas are karta karaka and karma karaka. Frequently, the karta karaka maps to agent theta role, and the karma to theme or goal theta role. Here we will not argue for the linguistic significance of karaka relations and differences with theta relations, as that has been done elsewhere (Bharati et al. (1990) and (1992)). In summary, karta karaka is that participant in the action that is most independent. At times, it turns out to be the agent. But that need not be so. Thus, 'boy' and 'key' are respectively the karta karakas in the following sentences

[1] The Paninian framework was originally designed more than two millennia ago for writing a grammar of Sanskrit; it has been adapted by us to deal with modern Indian languages.

```
---  semantic level (what the speaker
 |                          has in mind)
---  karaka level
 |
---  vibhakti level
 |
---  surface level (uttered sentence)
```

Fig. 1: Levels in the Paninian model

The boy opened the lock.
The key opened the lock.

Note that in the first sentence, the karta (boy) maps to agent theta role, while in the second, karta (key) maps to instrument theta role.

As part of this framework, a mapping is specified between karaka relations and vibhakti (which covers collectively case endings, post-positional markers, etc.). This mapping between karakas and vibhakti depends on the verb and its tense aspect modality (TAM) label. The mapping is represented by two structures: default karaka charts and karaka chart transformations. The default karaka chart for a verb or a class of verbs gives the mapping for the TAM label tA_hE called basic. It specifies the vibhakti permitted for the applicable karaka relations for a verb when the verb has the basic TAM label. This basic TAM label roughly corresponds to present indefinite tense and is purely syntactic in nature. For other TAM labels there are karaka chart transformation rules. Thus, for a given verb with some TAM label, appropriate karaka chart can be obtained using its basic karaka chart and the transformation rule depending on its TAM label. [2]

In Hindi for instance, the basic TAM label is tA_hE (which roughly stands for the present indefinite). The default karaka chart for three of the karakas is given in Fig. 2. This explains the vibhaktis in sentences A.1 to A.2. In A.1 and A.2, 'Ram' is karta and 'Mohan' is karma, because of their vibhakti markers ϕ and ko, respectively. [3] (Note that 'rAma' is followed by ϕ or empty postposition, and 'mohana' by 'ko' postposition.)

```
A.1  rAma   mohana   ko   pItatA   hE.
     Ram    Mohan    -ko  beats    is
     (Ram beats Mohan.)

A.2  mohana ko    rAma  pItatA  hE.
     Mohan -ko    Ram   beats   is
     (Ram beats Mohan.)
```

[2]The transformation rules are a device to represent the karaka charts more compactly. However, as is obvious, they affect the karaka charts and not the parse structure. Therefore, they are different from transformational grammars. Formally, these rules can be eliminated by having separate karaka charts for each TAM label. But one would miss the liguistic generalization of relating the karaka charts based on TAM labels in a systematic manner.

[3]In the present examples karta and karma turn out to be agent and theme, respectively.

KARAKA	VIBHAKTI	PRESENCE
Karta	ϕ	mandatory
Karma	ko or ϕ	mandatory
Karana	se or dvArA	optional

Fig. 2: A default karaka Chart

TAM LABEL	TRANSFORMED VIBHAKTI FOR KARTA
yA	ne
nA_padA	ko
yA_gayA	se or dvArA (and karta is optional)

Fig. 3: Transformation rules

Fig. 3 gives some transformation rules for the default mapping for Hindi. It explains the vibhakti in sentences B.1 to B.4, where Ram is the karta but has different vibhaktis, ϕ, ne, ko, se, respectively. In each of the sentences, if we transform the karaka chart of Fig.2 by the transformation rules of Fig.3, we get the desired vibhakti for the karta Ram.

```
B.1  rAma   Pala   ko   KAtA   hE.
     Ram    fruit  -ko  eats   is
     (Ram eats the fruit.)

B.2  rAma   ne    Pala   KAyA.
     Ram    -ne   fruit  ate
     (Ram ate the fruit.)

B.3  rAma   ko    Pala   KAnA   padA.
     Ram    -ko   fruit  eat    had to
     (Ram had to eat the fruit.)

B.4  rAma   se   Pala   nahI   KAyA   gayA
     Ram    -se  fruit  not    eat    could
     (Ram could not eat the fruit.)
```

In general, the transformations affect not only the vibhakti of karta but also that of other karakas. They also 'delete' karaka roles at times, that is, the 'deleted' karaka roles must not occur in the sentence.

The Paninian framework is similar to the broad class of case based grammars. What distinguishes the Paninian framework is the use of karaka relations rather than theta roles, and the neat dependence of the karaka vibhakti mapping on TAMs

and the transformation rules, in case of Indian languages. The same principle also solves the problem of karaka assignment for complex sentences (Discussed later in Sec. 3.)

2 Constraint Based Parsing

The Paninian theory outlined above can be used for building a parser. First stage of the parser takes care of morphology. For each word in the input sentence, a dictionary or a lexicon is looked up, and associated grammatical information is retrieved. In the next stage local word grouping takes place, in which based on local information certain words are grouped together yielding noun groups and verb groups. These are the word groups at the vibhakti level (i.e., typically each word group is a noun or verb with its vibhakti, TAM label, etc.). These involve grouping post-positional markers with nouns, auxiliaries with main verbs etc. Rules for local word grouping are given by finite state machines. Finally, the karaka relations among the elements are identified in the last stage called the *core parser*.

Morphological analyzer and local word grouper have been described elsewhere (Bharati et al., 1991). 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 karaka charts and transformation rules. The second task requires lakshan charts for nouns and verbs (explained at the end of the section).

A data structure corresponding to karaka chart stores information about karaka-vibhakti mapping including optionality of karakas. Initially, the default karaka chart is loaded into it for a given verb group in the sentence. Transformations are performed based on the TAM label. There is a separate data structure for the karaka chart for each verb group in the sentence being processed. Each row is called a *karaka restriction* in a karaka chart.

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. When testing a noun group against a karaka restriction of a verb group, vibhakti information is checked, and if found satisfactory, the noun group becomes a candidate for the karaka of the verb group.

The above can be shown in the form of a constraint graph. Nodes of the graph are the word groups and there is an arc labeled by a karaka from a verb group to a noun group, 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 shows that it takes a sentential or verbal karaka.) 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.

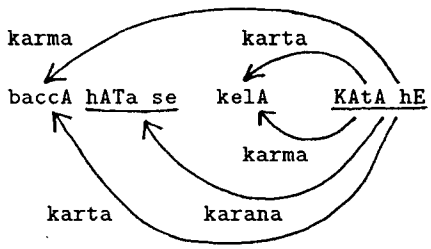

Fig. 4: Constraint graph

As an example, consider a sentence containing the verb KA (eat):

```
baccA  hATa se   kelA   KAtA hE.
child hand -se  banana  eats
(The child eats the banana with his hand.)
```

Its word groups are marked and KA (eat) has the same karaka chart as in Fig. 2. Its constraint graph is shown in Fig. 4.

A parse is a sub-graph of the constraint graph satisfying the following conditions:

1. For each of the mandatory karakas in a karaka chart for each demand group, there should be exactly one out-going edge from the demand group labeled by the karaka.
2. For each of the optional karakas in a karaka chart for each demand group, there should be at most one outgoing edge from the demand group labeled by the karaka.
3. There should be exactly one incoming arc into each source group.

If several sub-graphs of a constraint graph satisfy 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, and is probably ill-formed.

There are similarities with dependency grammars here because such constraint graphs are also produced by dependency grammars (Covington, 1990) (Kashket, 1986).

It differs from them in two ways. First, the Paninian framework uses the linguistic insight regarding karaka relations to identify relations between constituents in a sentence. Second, the constraints are sufficiently restricted that they reduce to well known bipartite graph matching problems for which efficient solutions are known. We discuss the latter aspect next.

If karaka charts contain only mandatory karakas, the constraint solver can be reduced to finding a matching in a bipartite graph. [4] Here is what needs to be done for a given sentence. (Perraju, 1992). For every source word group create a node belonging to a set U; for every karaka in the karaka chart of every verb group, create a node belonging to set V; and for every edge in the constraint graph, create an edge in E from a node in V to a node in U as follows: if there is an edge labeled in karaka k in the constraint graph from a demand node d to a source node s, create an edge in E in the bipartite graph from the node corresponding to (d, k) in V to the node corresponding to s in U. The original problem of finding a solution parse in the constraint graph now reduces to finding a complete matching in the bipartite graph {U,V,E} that covers all the nodes in U and V. [5] It has several known efficient algorithms. The time complexity of augmenting path algorithm is $O\left(min\left(|V|, |U|\right).|E|\right)$ which in the worst case is $O(n^3)$ where n is the number of word groups in the sentence being parsed. (See Papadimitrou et al. (1982), Ahuja et al. (1993).) The fastest known algorithm has asymptotic complexity of $O\left(|V|^{1/2}.|E|\right)$ and is based on max flow problem (Hopcroft and Karp (1973)).

If we permit optional karakas, the problem still has an efficient solution. It now reduces to finding a matching which has the maximal weight in the weighted matching problem. To perform the reduction, we need to form a weighted bipartite graph. We first form a bipartite graph exactly as before. Next the edges are weighted by assigning a weight of 1 if the edge is from a node in V representing a mandatory karaka and 0 if optional karaka. The problem now is to find the largest maximal matching (or assignment) that has the maximum weight (called the *maximum bipartite matching* problem or *assignment* problem). The resulting matching represents a valid parse if the matching covers all nodes in U and covers those nodes in V that are for mandatory karakas. (The maximal weight condition ensures that all edges from nodes in V representing mandatory karakas are selected first, if possible.) This problem has a known solution by the Hungarian method of time complexity $O(n^3)$ arithmetic operations (Kuhn, 1955).

Note that in the above theory we have made the following assumptions: (a) Each word group is uniquely identifiable before the core parser executes, (b) Each demand word has only one karaka chart, and (c) There are no ambiguities between source word and demand word. Empirical data for Indian languages shows that, conditions (a) and (b) hold. Condition (c), however, does not always hold for certain Indian languages, as shown by a corpus. Even though there are many exceptions for this condition, they still produce only a *small* number of such ambiguities or clashes. Therefore, for each possible demand group and source group clash, a new constraint graph can be produced and solved, leaving the polynomial time complexity unchanged.

The core parser also disambiguates 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. Similarly for the nouns. It should be noted (without discussion) that (a) disambiguation of senses is done only after karaka assignment is over, and (b) only those senses are disambiguated which are necessary for translation

The key point here is that since sense disambiguation is done separately after the karaka assignment is over it leads to an efficient system. If this were not done the parsing problem would be NP-complete (as shown by Barton et al. (1987) if agreement and sense ambiguity interact, they make the problem NP-complete).

3 Active-Passives and Complex Sentences

This theory captures the linguistic intuition that in free word order languages, vibhakti (case endings or post-positions etc.) plays a key role in determining karaka roles. To show that the above, though neat, is not just an adhoc mechanism that explains the isolated phenomena of semantic roles mapping to vibhaktis, we discuss two other phenomena: active-passive and control.

No separate theory is needed to explain active-passives. Active and passive turn out to be special cases of certain TAM labels, namely those used to mark active and passive. Again consider for example in Hindi.

[4] We are indebted to Somnath Biswas for suggesting the connection.

[5] A matching in a bipartite graph {U,V,E}is a subgraph with a subset of E such that no two edges are adjacent. A complete matching is also a largest maximal matching (Deo, 1974).

F.1 rAma mohana ko pItatA hE. (active)
Ram Mohan -ko beat pres.
(Ram beats Mohan.)

F.2 rAma dvArA mohana ko pItA gayA.(passv.)
Ram by Mohan -ko beaten was
(Mohan was beaten by Ram.)

Verb in F.2 has TAM label as yA_gayA. Consequently, the vibhakti 'dvArA' for karta (Ram) follows from the transformation already given earlier in Fig. 3.

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 Hindi sentences G.1, G.2 and G.3.

In G.1, Ram is the karta of both the verbs: KA (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 Pala (fruit) both are shared by the two verbs kAta (cut) and KA (eat). In G.3, the karta 'usa' (he) is shared between the two verbs, and 'cAkU' (knife) the karma karaka of 'le' (take) is the karana (instrumental) karaka of 'kAta' (cut).

G.1 rAma Pala KAkara mohana ko bulAtA hE.
Ram fruit having-eaten Mohan -ko calls
(Having eaten fruit, Ram calls Mohan.)

G.2 rAma ne Pala kAtakara KAyA.
Ram ne fruit having-cut ate
(Ram ate having cut the fruit.)

G.3 Pala kAtane ke liye usane cAkU liyA.
fruit to-cut for he-ne knife took
(To cut fruit, he took a knife.)

The observation that the matrix verb, i.e., main verb rather than the intermediate verb controls the vibhakti of the shared nominal is true in the above sentences, as explained below. 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 identifies shared karakas.

The first part is in terms of the karaka vibhakti mapping described earlier. Because the intermediate verbs have their own TAM labels, they are handled by exactly the same mechanism. For example, kara is the TAM label [6] of the intermediate verb groups in G.1 and G.2 (KA (eat) in G.1 and kAta (cut) in G.2), and nA [7] is the TAM label

[6] 'kara' TAM label roughly means 'having completed the activity'. But note that TAM labels are purely syntactic, hence the meaning is not required by the system.

[7] This is the verbal noun.

TAM LABEL	TRANSFORMATION
kara	Karta must not be present. Karma is optional.
nA	Karta and karma are optional.
tA_huA	Karta must not be present. Karma is optional.

Fig. 5: More transformation rules

of the intermediate verb (kAta (cut)) in G.3. As usual, these TAM labels have transformation rules that operate and modify the default karaka chart. In particular, the transformation rules for the two TAM labels (kara and nA) are given in Fig. 5. The transformation rule with kara in Fig. 5 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 KA (eat) in G.1 and kAta (cut) in G.2 do not have (independent) karta karaka present in the sentence. Ram is the karta of the main verb. Pala (fruit) is the karma of the intermediate verb (KA) in G.1 but not in G.2 (kAta). In the latter, Pala is the karma of the main verb. All these are accommodated by the above transformation rule for 'kara'. The tree structures produced are shown in Fig. 6 (ignore dotted lines for now) where a child node of a parent expresses a karaka relation or a verb-verb relation.

In the second part, there are rules for obtaining the shared karakas. Karta of the intermediate verb KA in G.1 can be obtained by a sharing rule of the kind given by S1.

Rule S1: Karta of a verb with TAM label 'kara' is the same as the karta of the verb it modifies [8].

The sharing rule(s) are applied after the tentative karaka assignment (using karaka to vibhakti mapping) is over. The shared karakas are shown by dotted lines in Fig. 6.

4 Conclusion and future work

In summary, this paper makes several contributions:

- It shows that the Paninian framework applied to modern Indian languages gives an elegant account of the relation between vibhakti and karaka roles. The mapping is elegant and compact.

[8] The modified verb in the present sentences is the main verb.

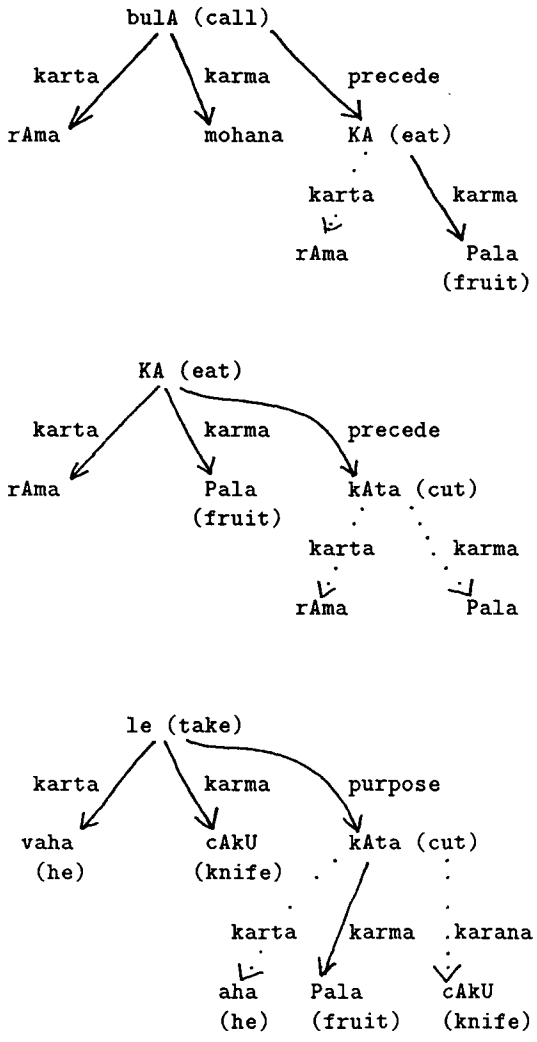

Fig. 6: Modifier-modified relations for sentences G.1, G.2 and G.3,respectively. (Shared karakas shown by dotted lines.)

- The same basic account also explains active-passives and complex sentences in these languages. This suggest that the solution is not just adhoc but has a deeper underlying unity.
- It shows how a constraint based parser can be built using the framework. The constraints problem reduces to bipartite graph matching problem because of the nature of constraints. Efficient solutions are known for these problems.

  It is interesting to observe that such a parser (designed for free word order languages) compares well in asymptotic time complexity with the parser for context free grammars (CFGs) which are basically designed for positional languages.

A parser for Indian languages based on the Paninian theory is operational as part of a machine translation system.

As part of our future work, we plan to apply this framework to other free word order languages (i.e., other than the Indian languages). This theory can also be attempted on positional languages such as English. What is needed is the concept of generalized vibhakti in which position of a word gets incoporated in vibhakti. Thus, for a pure free word order language, the generalized vibhakti contains pre- or post-positional markers, whereas for a pure positional language it contains position information of a word (group). Clearly, for most natural languages, generalized vibhakti would contain information pertaining to both markers and position.

Acknowledgement

Vineet Chaitanya is the principal source of ideas in this paper, who really should be a co-author. We gratefully acknowledge the help received from K.V. Ramakrishnamacharyulu of Rashtriya Sanskrit Sansthan, Tirupati in development of the theory. For complexity results, we acknowledge the contributions of B. Perraju, Somnath Biswas and Ravindra K. Ahuja.

Support for this and related work comes from the following agencies of Government of India: Ministry of Human Resource Development, Department of Electronics, and Department of Science and Technology.

References

Ahuja, R.K., Thomas L. Magnanti, and James B. Orlin, Network Flows: Theory, Algorithms

and Applications, Prentice-Hall, 1993 (forthcoming).

Barton, G. Edward, Robert C. Berwick, and Eric S. Ristad, Computational Complexity and Natural Language, MIT Press, Cambridge, MA, 1987.

Bharati, Akshar, Vineet Chaitanya, and Rajeev Sangal, A Computational Grammar for Indian Languages Processing, Journal of Indian Linguistics, IL-51. (Also available as TRCS-90-96, Dept. of CSE, IIT Kanpur, 1990.)

Bharati, Akshar, Vineet Chaitanya, and Rajeev Sangal, Local Word Grouping and Its Relevance to Indian Languages, in Frontiers in Knowledge Based Computing (KBCS90), V.P. Bhatkar and K.M. Rege (eds.), Narosa Publishing House, New Delhi, 1991, pp. 277-296.

Bharati, Akshar, Vineet Chaitanya, and Rajeev Sangal, LFG, GB, and Paninian Frameworks: An NLP Viewpoint, Part of NLP tutorial for CPAL-2: UNESCO 2nd Regional Workshop on Computer Processing of Asian Languages, 12-16 March 1992, I.I.T. Kanpur. (Also available as TRCS-92-140, Dept. of CSE, IIT Kanpur.)

Cardona, George, Panini: A Survey of Research, Mouton, Hague-Paris, 1976.

Cardona, George, Panini: His Work and Its Tradition (Vol. 1: Background and Introduction), Motilal Banarsidas, Delhi, 1988.

Covington, Michael A., Parsing Discontinuous Constituents in Dependency Grammar (Technical Correspondence), Computational Linguistics, 16,4 (Dec. 1990), p.234.

Deo, Narsingh, Graph Theory, Prentice-Hall, 1974.

Hopcroft, J.E. and R.M. Karp, "A $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs," J. SIAM Comp. 2 (1973), pp.225-231.

Kashket, Michael B., Parsing a free-word-order language: Warlpiri, Proc. of 24th Annual Meeting of ACL, pp. 60-66.

Kiparsky, P., Some Theoretical Problems in Panini's Grammar, Bhandarkar Oriental Research Institute, Poona, India, 1982.

Kuhn, H.W. "The Hungarian Method for the Assignment Problem", Naval Research Logistics Quarterly, 2 (1955), pp.83-97.

Papadimitrou, Christos H., and K. Steiglitz, Combinatorial Optimization, Prentice-Hall, Englewood Cliffs, 1982.

Perraju, Bendapudi V.S., Algorithmic Aspects of Natural Language Parsing using Paninian Framework, M.Tech. thesis, Dept. of Computer Science and Engineering, I.I.T. Kanpur, Dec. 1992.