Oriya Nominal Forms: A Finite State Processing

Kalyanamalini Sahoo
Resource Center for Indian Language Technology Solutions
Department of Management Studies
Indian Institute of Science
Bangalore - 560 012 INDIA
+91-80-293 3267
kalyani@mgmt.iisc.ernet.in

Abstract— This paper discusses the morphological processing of nouns in Oriya in a deterministic Finite State Automaton (FSA). It proposes a model for designing a morphological analyzer for Oriya nouns, which can provide lexical, morphological and syntactic information for each lexical unit in the analyzed nominal form. It draws out a finite-state machine that accepts valid sequences of morphemes in a nominal form and rejects invalid ones.

1. INTRODUCTION

Morphological analysis of words is a basic tool for automatic language processing, and indispensable when dealing with agglutinative languages like Oriya. In this context, some applications, like spelling correction, do not need more than segmentation of each word into its different component morphemes along with their morphological information. However, there are other applications such as lemmatization, tagging, phrase recognition, and determination of clause boundaries, which need an additional *morphosyntactic parsing* of the whole word. This work proposes a model for designing a morphological analyzer for Oriya nouns, which can provide lexical, morphological and syntactic information for each lexical unit in the analyzed nominal form. It draws out a finite-state machine that accepts valid sequences of morphemes in a nominal form and rejects invalid ones. We can use the FSA to solve the problem of morphological recognition,; determining whether an input string of morphemes makes up a legitimate Oriya word or not. We do this by taking the morphotactic FSAs and plugging in each nominal form into the FSA. We do this via two-level morphology (TLM). TLM represents a word as a correspondence between a lexical level, which represents a simple concatenation of morphemes making up a word, and the surface level, which represents the actual spelling of the final word. Morphological parsing is implemented by building mapping rules that map morpheme sequences like *pilaa-maane* 'children' on the surface level into morpheme and features sequences like child + plural at the lexical level.

The paper is organized as follows. Section 2 gives a brief description of Oriya nominal forms. Section 3 describes the architecture for morphological processing, specifies the phenomena covered by the analyzer, explains its design criteria, and presents the processing details. Finally, the paper ends with some concluding remarks.

2. ORIYA NOMINAL FORMS

Oriya is a syntactically head-final and morphologically agglutinative language. The subject NP agrees with the verb in person, number and honorificity. Oriya has natural gender (as opposed to grammatical gender), that is, gender is dictated by semantic rather than formal characteristics of words. Gender is not specified in the Agr features, nor it affects other grammatical categories like pronoun or verb. A number of morphemes carrying different grammatical functions get affixed to the nominal root to make a nominal form. The major affixing categories that cluster around the noun are: numeral, classifier, quantifier, number marker, negation marker, qualitative affirmative marker, Case marker and postpositions etc. Except for negation and Qual Aff markers, Oriya nominal forms typically contain a sequence of morphemes followed by a noun root. E.g.

(1) pilaa-dui-Taa-jaaka-nku
child-two-Cl-Quan-Case
'To those two children'

(2) apa-karma-maana
NEG-deed-pl.
'Bad deeds.'

[Cl=classifier, Quan=quantifier, NEG=negation marker, Pl=plural marker]

2.1 Pronouns

In Oriya all the pronouns can be used with reference to nouns of all genders. Like nouns, the pronouns can have Case features too. Pronouns are marked for person, number, honorificity and animacy. Pronouns can be ±definite. Reciprocal pronouns like *paraspara* 'each other', Reflexive pronouns like *se nije* 'he himself', Demonstrative pronouns like *ehaa/ehi* 'this', *eguDika/eguDaaka* 'these', *se/sehi* 'that' and *seguDik /seguDaak* 'those', Relative pronouns like *je* 'whoever', *jaahaaku* 'whomsoever', Correlative pronouns like *jie* 'who(ever)'----*se* 'he'/ 'she', Interrogative pronouns like *kie* 'who', *kaahaaku* 'whom', Distributive pronouns like *pratyeka* 'each'/ 'every', Universal pronouns

0-7803-7651-X/03/$17.00 © 2003 IEEE

like *samaste* 'all', Existential pronouns like *jaNe* 'one person', *goTe* 'a'/ 'one', Compound pronouns like *kehi jaNe / kie jaNe* 'somebody', Personal pronouns like *mun* 'I', *tu* 'you', *tume* 'you', *aapaNa* 'you', *se* 's/he' are found in Oriya. The sub-types of pronouns used in Oriya can be shown as the following type-hierarchy [1]:

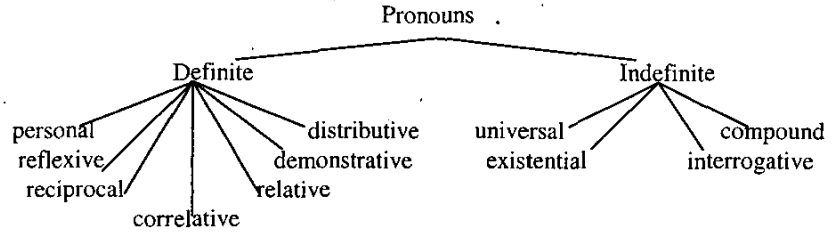

Figure 1. Types of pronouns in Oriya

Reduplicated pronominal forms like *kichhi kichhi* /something something/ 'certain amount' are also used in Oriya. Inflections are usually suffixed to the stem.

2.2 Number, Numeral and Quantifier

Oriya distinguishes between the number marker and numerals. Number is realized either as a numeral or as a nominal inflection, but not as both; e.g.

(3) a. paancha-Taa- pilaa
five Cl child
'Five children.'

b. pilaa-maane
child Pl
'Children.'

(4) a. paancha bhaai
five brother
'Five brothers.'

b. *paancha-Taa pilaa-maane
five Cl child pl
'Five children.'

Oriya has singular and plural numbers. Usually, singularity is indicated by the bare stem, or by the numeral *eka, -e, goTaae, goTie, goTe, -ka*, etc.

Plurality is marked by:
--the suffix *-e* in the nominal, e.g. *loke* 'people'
--the suffix *maane* or *maana*, e.g. *pilaa-maane* 'children'
--prefixing or suffixing *sabu* 'all', *samasta* 'all', e.g. *sabu-pilaa* /all child/ 'all the children', *aame-sabu* 'we all'
--the bound morphemes denoting entirety, multitude, such as *jaaka*, *taka* (in the case of uncountable nouns) and *guDaa*. E.g. *khira-taka* 'the whole amount of milk', *masaa-guDaa* 'the mosquitoes'

The plural nominal form is declined for Case and other grammatical functions.

However, numeral occurs in a position immediately preceding or following the root noun, while plural markers can occur immediately following the nominal root as well as at the final position of the nominal stem following the classifier. Number markers and quantifiers are in complementary distribution, hence, occupy the same positional slot in the nominal form.

2.3 Case

The classification of the major arguments in the sentences, i.e. 'Case roles', involves identifying the manner of their involvement in the state/event/action. Like other Indo-Aryan languages, Oriya follows the Case system of Sanskrit. The Cases are named and defined after the Sanskrit grammar. There are eight Cases found in Oriya: Nominative, Accusative, Instrumental, Dative, Ablative, Genitive, Locative, and Vocative. Nouns and pronouns are marked for Case indicating some grammatical function. However, pronouns lack the vocative Case. The Case morphemes are realized as follows:

Table 1. The Case morphemes in Oriya

Case	Singular	Plural/[+Hon] sg
Nominative	-	-e
Accusative	*ku*	*nku, maananku*
Instrumental	*re, dwaaraa, dei*	*re, dwaaraa, dei*
Dative	*ku*	*nku, maananku*
Ablative	*ru, Thaaru*	*maanankaThaaru*
Genitive	*ra*	*nkara, maanankara*
Locative	*re, Thaare*	*maanankaThaare*
Vocative	*he, bho*	-

The NOM sg morpheme is not morphologically realized. The morpheme *-ku* is generally used in the ACC with reference to definiteness in direct objects (DO), while an indefinite object NP may not have the Case morpheme realized. Note that although the ACC marker and the DAT marker are homophonous, and both of them have been derived from the objective Case, they have different grammatical functions. Usually, the ACC *-ku* occurs with a DO, and in certain cases with locational adverbials; while the DAT *-ku* is used with the Indirect Object (IO) in ditransitive constructions. Thus, they are in complementary distribution. The DAT *-ku* is never optional, while the ACC

-ku is very often dropped in the case of indefinite NPs. Besides being used with an object, the DAT *-ku* can occur with the EXPERIENCER argument in subject position. In conclusion, we can say that the ACC *-ku* is a marker of definiteness in direct objects, while the DAT *-ku* can appear on subject as well as on IO. So, we distinguish the ACC *-ku* from the DAT-*ku*.

The vocative Case markers like *he, bho* which are used for addressing a person with honour, e.g. *he jagannaatha / bho jagannaathu* 'hello Jagannaatha' are rare in colloquial speech. Usually, the interjection *aahe* [+honorific] or [+formal] and *aare* [-honorific] are used in addressing men, while *aago* [+honorific] or [+formal], *aalo* [-honorific] or [+affectionate] is usual in addressing women. The initial vowel usually drops when the interjection follows the noun, e.g. *raama-re* 'hello Rama', *mili-lo* 'hello Mili'. The nominative sg. and pl. forms can also be used (usually with some change in the intonations) for addressing a person.

Case morphemes occur at the final position of the nominal form.

2.4 Postpositional words

Tripathy [2] has pointed out that Oriya has almost discarded the synthetic or organic inflexion of Sanskrit grammar, and has adopted the new and simpler device of expressing the Case relations of Sanskrit grammar (and of Prakrit grammar) by taking recourse to postpositions.

The following postpositional words are suffixed to the noun to express different Case relations: *pare* 'after', *kari* 'by', *nimite* 'for, *parjyante* 'up to', *paain* 'for', *prati* 'to', 'against', *byatita* 'without', *binaa* 'without', *boli* 'literally speaking', *bhitare* 'in', 'inside', *laagi* 'for', *sahite* 'with'. Case-endings, postpositions, or postpositional words are added only to the substantives and not to the adjectives that they are modified by. They are suffixed both in singular and in plural. They occur at the final position of the nominal stem. Case markers and postpositional words are mutually exclusive and occur in the same positional slot in a nominal form.

2.5 Qual Aff markers & Qual NEG markers

Qualitative affirmative markers like *su-, sat-, sad-* and qualitative NEG markers like *apa-, ku-, duh-, a-, bad-* are used. They are usually prefixed to the noun root.E.g. *su-guNa* 'good quality', *sad-byabahaara* 'good behaviour', *a-sundara* 'ugly' *bad-abhyaasa* 'bad habit', *apa-karma* 'bad deed'. Affirmative markers and NEG markers are mutually exclusive. Only one item can be attached to the noun Root at a time.

3.THE ARCHITECTURE FOR MORPHOLOGICAL PROCESSING

As we discussed above, there is a rich structure in these morphological sequences, and in this paper we will model it by using a deterministic finite-state automaton. Such a morphological analyzer has to consider three main aspects (Ritchie et al.)[3], [4]:

(5) i) Morphographemics (also called morphophonology):
This term covers orthographic variations that occur when linking morphemes.
ii) Morphotactics:
Specification of which morphemes can or cannot combine with each other to form valid words.
iii) Feature-combination:
Specification of how these morphemes can be grouped and how their morphosyntactic features can be combined.

As a consequence of the rich morphology of Oriya, we control morphotactic phenomena, as much as possible, in the morphological segmentation phase. Alternatively, a model with minimal morphotactic treatment (Ritchie *et al*) would produce too many possible analyses after segmentation, which should be rejected in a second phase. The morphological analyzer created by (Ritchie *et al*) does not adopt finite state mechanisms to control morphotactic phenomena. Their two-level implementation incorporates a straightforward morphotactics, reducing the number of sublexicons to the indispensable (prefixes, lemmas and suffixes). This approximation would be highly inefficient for agglutinative languages like Oriya, as it would create many nonsensical interpretations that should be rejected by the system.

3. 1 A Deterministic Finite State Automaton

Since we cannot list every word in the language, computational lexicons are structured as a list of stems and affixes with a representation of the morphotactics. One way to model morphotactics is the finite-state automaton. We use a deterministic FSA to solve the problem of morphological recognition. It will determine whether an input string of morphemes makes up a legitimate Oriya noun or not. Such identification of sequences have a number of practical applications like spell checker, machine translation etc.

A (deterministic) Finite State Automaton (FSA) is a device that receives a string of symbols as input, reads the string one symbol at a time from left to right, and after reading the last symbol halts and indicates either acceptance or rejection of the input. The automaton performs computation by reacting on a class of inputs (on strings or sequences of symbols). The concept of a *state* is the central notion of an automaton. A state of an automaton is analogous to the arrangement of bits in the memory banks and registers of an actual computer. Here, we consider a state as a characteristic of an automaton which changes during the course of a computation and which serves to determine the relationship between inputs and outputs. For our automaton, the *memory* consists simply of the states themselves. The

computation of an FSA is directed by a 'program', which is a finite state of instructions for changing from state to state as the automaton reads input symbols. Given an input, the computation begins in a designated state, the *initial state*. After reading the input, the automaton either accepts or rejects it after some finite amount of computation. Thus, an FSA can be visualized as composed of

i) a control box, which at any point in the computation can be in one of the allowed internal states

ii) a reading head, which scans a single symbol of the input.

In a more formal way, as in [5]-[6],

A (deterministic) finite-state automaton is a quintuple (Q, $\sum$, q_0, F, δ) where

- Q is a finite set of N states $q_0, q_1, \ldots, q_n$
- $\sum$ is a finite input alphabet of symbols
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$, the set of final states
- $\delta(q,i)$ is the transition function or transition matrix between states. Given a state $q \in Q$ and an input symbol $i \in \sum$, $\delta(q,i)$ returns a new state $q' \in Q$. δ is thus a relation from $Q \times \sum$ to Q.

Thus, if the automaton is in a state $q \in Q$ and the symbol read from the input is a, then d (q,a) uniquely determines the state to which the automaton passes. This property entails high run-time efficiency, since the time it takes to recognize a string is linearly proportional to its length.

3.2 The FSA for Oriya

This section discusses how the FSA can be conceived as applying to Oriya nominal forms.

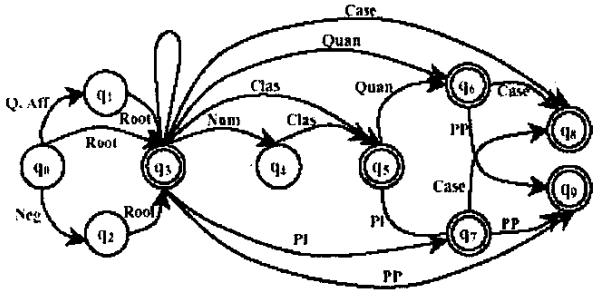

Fig. 2 The FSA for nominal forms in Oriya

The automaton is represented as a directed graph: a finite set of vertices (nodes), together with a set of directed links between pairs of vertices called arcs. Each node corresponds to a state. States are represented as circles with name tags in them. Arcs are represented by arrows going from one state to another state. The final states are represented by two concentric circles.

The machine starts at the initial state, runs through a sequence of states by computing a morpheme in each transition. If it matches the symbol on an arc leaving the current state, then it crosses that arc, and moves to the next state, and thus, advances one symbol in the input. Each state through which the speaker passes represents the grammatical restrictions that limit the choice of the next morpheme. Such a process gets iterated until the machine reaches the final state, successfully recognizing all the morphemes in the input string. But if the machine gets some input that does not match an arc, then it gets stuck there and never gets to the final state. This is considered as the FSA/machine rejecting or failing to accept an input. The path moves from the initial point on the left to the final point on the right, proceeding in the direction of arrows. Once the arrow moves one step, there is no backward movement (Of course, recursion of an item can be shown by using closed loops). The resulting FSA is deterministic in the sense that given an input symbol and a current state, a unique next state is determined.

The FSA starts at the initial state (q_0). From the initial state it can choose the Root state directly or Root via the Qual Aff state or Root via the Neg state, depending on whether it is qualitatively an affirmative or negative noun. From the Root state, it has various options to move to the next state: it can move to the Num state, classifier final state, quantifier final state, plural final state, Case final state or PP final state, out of which except for the Num state all the other states are final states. From the Num nonfinal state, it can go to Clas final state. From the Clas final state, it can choose either Quan state or Pl. state, which are final states too. From the Quan state as well as from Pl. state it can choose either Case final state or PP final state. As both the states are final states and no other input is available, the machine stops there, it does not traverse further. Reduplicated pronominal stems can be processed by using closed loop at q_3.

To test how a nominal form in the language be processed by this machine, we can take a concrete example like (6). This negative nominal form can be processed as follows.

(6) apa-karma-maana
NEG-root-pl.
'bad deeds'

It has 4 states. State 0 is the initial state and state 3 is the final state. It also has 3 transitions.

$Q = \{q_0, q_1, q_2, q_3\}$

$\sum = \{apa, karma, maana\}$

q_0= the initial state

$F = \{q_3\}$

$\delta(q,i)$ can be defined by the transition table as follows:

Table 2. The State Transition Table for the FSA for *apa-karma-maana*

	Input		
State	apa	karma	maana
0	1	Ø	Ø
1	Ø	2	Ø
2	Ø	Ø	3
3:	Ø	Ø	Ø

In the transition table, state3 is marked with a colon to indicate that it is a final state. Ø indicates an illegal or missing transition. It can be read as follows: "if we are in state 0 and we see the input *apa*, we must go to the state 1. If we are in state 0 and we see the input *karma* or *maana*, we fail."

4. CONCLUSION

An efficient finite-state morphological analyser has been described. We specify the co-occurrence restrictions of the morphemes in a nominal form and use the FSA to solve the problem of morphological recognition; determining whether an input string of morphemes makes up a legitimate Oriya noun or not. Such identification of sequences have a number of practical applications like spell checker, machine translation, etc. The morphological analyzer will help us to build a computational lexicon structured as a list of stems and affixes with a representation of the morphotactics and also can be used for designing a morphosyntactic analysis for each word in unrestricted Oriya texts. The design of the deterministic FSA we propose is new for Oriya, as far as we know. We think that our design could be interesting for the treatment of other agglutinative languages too.

REFERENCES

[1] Sahoo, Kalyanamalini, *Oriya Verb Morphology and Complex Verb Constructions.* Ph.D dissertation. Norwegian University of Science and Technology, Trondheim, Norway, 2001.

[2] Tripathy, Kunjabihari, *The Evolution of Oriya Language & Script.* P 137. Cuttack: Utkal University, 1962.

[3] Ritchie, Pullman, Black & Russel, *Computational morphology: Practical Mechanisms for the English Lexicon.* ACL-MIT Press Series in Natural Language Processing, 1992.

[4] Sproat, R., *Morphology and Computation.* ACL-MIT Press Series in Natural Language Processing, 1992.

[5] Jurafsky, D & J.H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* New Jersy: Prentice Hall, 2000.

[6] Roche, Emmanuel & Yves Schabes (eds.). *Finite State Language Processing.* The MIT Press, 1997.