A Network and Process Model for Morphological Analysis/Generation[1]

K. Narayana Murthy,
Department of Computer and Information Sciences,
University of Hyderabad, Hyderabad, INDIA
email: knmcs@uohyd.ernet.in

ABSTRACT

In this paper we present morphology of Kannada, one of the four major literary languages of the Dravidian family, through a computational model of morphological analysis/generation which can be called the 'network and process' model. Kannada is a diglossic language - the formal or the literary variety differs significantly from the spoken variety. Here we restrict ourselves to the standard dialect of the literary variety.

Kannada is mainly an agglutinating language of the suffixing type. A complete word form may consist of several suffixes joined according to well defined rules of saMdhi. In order to completely specify the morphological processes in a language such as Kannada, we need to look at several things. We need to know 1) the various affixes, 2) the relationships between the affixes and the grammatical categories/features and hence the meaning, 3) particular combinations of affixes that are permitted in the language, and finally, 4) the saMdhi processes that take place during affixation. In the particular model presented here, the first three aspects are encoded into a monolithic representation called the 'network' and the last aspect is handled by a set of ordered 'processes' of saMdhi. Hence the name 'network and process' model.

There are several advantages to this separation of issues into the two modules. Firstly, a network is an ideal representation to depict constraints on combinations and ordering of affixes. It is also an economical representation since common subparts can be collapsed. Secondly, the network itself can be used bidirectionally and hence we can have a single representation for both analysis and generation. Changes made to the network will be reflected automatically and immediately in both the analyzer and the generator. A network representation is conceptually as well as computationally very simple and efficient. On the other

[1] The work reported in this paper was supported in part by the Government of Karnataka under a consultancy project entitled "Machine Aided Translation from English to Kannada"

hand, the saMdhi processes are relatively more complex and they are sensitive to the order in which the processes are carried out. The variations across languages also seem to be more pronounced. Exceptions and idiosyncrasies are all bundled up in these processes. It is therefore a good strategy to separate out the saMdhi processes from the rest of the issues.

In this paper we give examples of the networks and the saMdhi processes for Kannada and show how these can be used for morphological analysis and generation. We also describe in brief the implemented system called MORPH. MORPH incorporates an analyser as well as a generator. The generator can generate specified word forms or an entire paradigm. This stands in sharp contrast to other computational models of morphology where one starts by listing out paradigms by hand. The system is menu based and the menus are generated dynamically. Hence any changes made to the network get reflected in the menus automatically. The dictionary includes more than 10,000 Kannada roots. While the internal representations are in Roman, user can give inputs and see the outputs either in Roman or in Indian scripts through GIST.

Currently, MORPH is being applied for Machine Aided Translation from English to Kannada. Experiments have also been made with several other languages including Telugu, Tamil and Bengali. MORPH seems to hold promise as a general tool suitable for a variety of languages.

1. Introduction:

This paper deals with the morphology of Kannada, one of the four major literary languages of the Dravidian family, spoken mainly in the state of Karnataka, South India. We present a particular computational model which we call the 'network and process' model. We start with a description of the model, present examples and then discuss the merits of the particular computational model.

Kannada is a diglossic language - the formal or the literary variety differs significantly from the spoken or the colloquial variety. For example, the first person singular form of the verb *'tinnu'* (eat) in the non-past tense is *'tinnutte:ne'* in the literary variety and *'tinnutti:ni'* (which gets further simplified to *'tiMti:ni'*) in the spoken variety. Here we restrict ourselves to the literary variety.

Kannada has a very rich and complex range of regional, social and stylistic variation. There are three major regional varieties

- the "Mysore" dialect, the "Mangalore" dialect, and the "Dharwar" dialect (Upadhyaya, 1976). However, finer regional distinctions are possible and Rajapurohit (1982) has given a more elaborate analysis with at least 7 dialectal regions. In this paper we show examples from the "Mysore" variety.

Kannada is mainly an agglutinating language of the suffixing type. Nouns are marked for number and case and verbs are marked, in most cases, for agreement with the subject in number, gender and person. This makes Kannada a relatively free word order language. In this paper we take examples from the inflectional morphology of Kannada verbs and nouns. We restrict ourselves to suffixation and exclude prefixation, external saMdhi and compounds. An implemented system called 'MORPH' is also briefly discussed. MORPH can be used both for analysis and generation.

2. The Network and Process Model:

We now present a computational model for morphological analysis and generation, called the Network and Process model. In this model, morphology is divided into two distinct but related components respectively called the network and the process. The network component includes three aspects:

1. the various affixes that take part in the morphological processes in the language

2. the associations between the affixes and the grammatical features (and hence meaning)

3. constraints on the selection of affixes in various combinations

The other major aspect of how exactly the affixes combine with the roots/stems, is dealt with in a separate component called the process component. As we shall see soon, this division into the two components offers certain unique advantages over other possible approaches.

2.1 The Network:

Consider the structure a finite verb in Kannada. In its simplest form, a finite verb form includes the root, an aspect suffix, a tense suffix and a gender-number-person suffix, taken in that order. For example

ma:Du	*+*	*O*	*+*	*utt*	*+*	*a:ne*	*= ma:Dutta:ne*
Root: (do)		*Aspect:O*		*Tense: non_past*		*GNP: m,sl,p3*	*((he) does)*

It may be noted that the gender, number and person are all encoded into a single atomic suffix. Further, while Kannada has three persons, three genders and two numbers, not all combinations have distinct suffixes. Thus the suffix *'a:re'* indicates third person masculine or feminine plural - it is partly neutral to gender. Similarly, in first and second persons, there are no gender distinctions. Also, the gender-number-person suffixes show variations across the three tenses - there are three separate sets of suffixes, one for each tense. For example, the n-sl-p3 suffix is *'ide'* in the past tense, *'ade'* in the non-past, and *'udu'* in the future/habitual. Thus the selection of a particular gender-number-person suffix is conditioned by the selection of the tense suffix and vice versa. Likewise, there are constraints on the selection of auxiliary verbs in the non-finite forms. For example, aspectual auxiliaries like *biDu* (lit. leave), *no:Du* (lit. see), koDu (lit. give) and *ha:ku* (lit. put) occur only after a past verbal participle and other aspectual auxiliaries such as *a:gu* (lit. become) and *toDagu* (lit. start) occur only in an infinitival context. Of course, such variations are not idiosyncratic to Kannada, they do exist in many other languages too. The network component provides a simple and efficient scheme for incorporating all such requirements. Figure 1 gives a sample of the network for the inflectional morphology of Kannada verbs and nouns.

A network consists of a set of states interconnected by labelled arcs. The states are given labels only for convenience of reference. The states in figure 1 are represented by numbers. The arc labels are the affixes. Each affix also carries the associated grammatical feature bundles. There is a well defined start state and one or more well defined terminal states. To generate a complete word form from a given root, we start at the start state and move through a sequence of states until we reach one of the terminal states. At each state transition, we attach the affix on the corresponding arc label. If the affix attachment is done according to appropriate saMdhi rules as specified in the process component, we get the complete word form. Some examples are given below to illustrate this process of generation:

no:Du + 0 + utt + a:ne = no:Dutta:ne
Root: (see) Aspect:0 Tense: non-past GNP:m,sl,p3 ((he) sees)

tinnu + i + ha:ku + i
Root: (eat) past_verbal_part. aspectual aux. past_verbal_part.

+ iru + id + anu = tiMduha:kiddanu
Aspect: Perf. Tense: past GNP:m,sl,p3 ((he) had eaten)

For analysis, we start from a terminal state and look for suffixes that match the labels of arcs leading to those states. By a series of affix stripping steps, we get to the root which can then be checked against the lexicon. Computationally, it is much more efficient to work from right to left for analysis since the number of suffixes is much smaller than the number of roots. In more technical terms, the branching factor is much smaller if we work from right to left than the other way round. Further, unlike some of the other models, we work directly at the level of affixes and stems/roots, not at the level of individual letters. This adds to the efficiency of our model.

Formally, the network component is an extension of the well known concept of Finite Automata (Hopcroft J.E., Ullman J.D., 1979, Roche E., Schabes Y. (Eds), 1997). The major extension is in terms of the incorporation of a separate process component that combines the affixes with the root/stem according rules of saMdhi formation rather than simply concatenating the strings. The network used here is a non-deterministic finite automaton. It is well known that for every non-deterministic finite automaton, there is an equivalent deterministic finite automaton (Hopcroft J.E., Ullman J.D., 1979). Recognition/generation algorithms for deterministic finite automata are of linear time complexity and hence about the fastest ever possible.

The network component is itself inherently bi-directional and can be used for both analysis and generation. Changes made to the network are automatically and immediately reflected in both the analyzer and the generator, saving the burden of having to modify the two separately, each time checking for consistency.

Note that it is impossible to select, say, the past tense and a GNP suffix from the non-past set. Selectional constraints are naturally and elegantly incorporated in a network. Some languages involve additional constraints based on such things as syllabic pattern, consonant/vowel ending etc. It is possible to incorporate such constraints into the network itself, so that affixes are always selected appropriately. An arc can then be taken if and only if the conditions specified therein, if any, are satisfied.

Networks are efficient in representation too. Common subparts can be collapsed. Loops can also be represented easily. For example, from state 10 in figure 1 we can go back to state 0. Thus we can generate and analyze word forms such as

(ma:Du + isu + i + koLLu + i + biDu + i + ho:gu + O + id + anu)
'ma:DisikoMDubiTTuho:danu'

Also, networks have simple visual representations, making it easy for people to read and understand.

2.2 The Process:

Here we are concerned with the saMdhi processes that take place between affixes and the root/stem. While in some languages the rules of saMdhi may be fairly simple and straight forward, there are languages where the saMdhi processes are quite involved. Our model incorporates a general and powerful process component and is thus suitable for all types of languages.

The most common saMdhi process in Kannada is the deletion (lo:pa) of final vowel. If a vowel initial suffix combines with a vowel final stem/root, the last vowel of the stem/root is deleted. Thus, *ma:Du + id => ma:Did.*

It must be noted that Kannada has no consonant ending words. Even in the case of loan word such as car, an enunciative vowel -u is added to make it *'ka:ru'*. A large number of Kannada words end with -u. However, the final -u in many words is only an enunciative vowel and is not real. Thus the root is *'ma:D'* and it is simply concatenated with 'id' to form *'ma:Did'* - there is really no lo:pa taking place. None the less traditionally the citation forms in dictionary include the enunciative vowel and hence a lo:pa saMdhi will have to be carried out in the process component.

In some cases, a:gama saMdhi also occurs. For example, *'bare' + 'itu' => 'bareyitu'*. Similarly, an -a ending human noun gets an 'n' inflectional increment: *'huDuga' + 'annu' => 'huDuganannu'.* The process component checks for such conditions and then applies the appropriate saMdhi rules for analysis or generation as the case may be.

For example, the dative case suffix in Kannada is *'ige'* but it has the following variations: Neuter nouns ending with -a take *'kke'* and -e and -i ending words get *'ge'*. Thus we get the dative forms

mara	=>	*marakke*
ta:yi	=>	*ta:yige*
taMde	=>	*taMdege*
mu:gu	=>	*mu:gige*

huDuga => huDuganige

Note that although the suffix in question in an vowel initial suffix, the final vowel in the root is not lost in the last example because of the inflectional increment that gets in between.

The process component also serves as a base for incorporating all exceptions and idiosyncratic variations. This strategy of capturing the variations inside the process component has the advantage that the network needs to show only the default suffixes - the rule is segregated from the exceptions. The network depicts only the major rules and thus becomes simpler and more economical. Keeping the basic linguistic requirements in mind, one has the freedom to divide the work between the network and process components as appropriate.

In the MORPH tool, this strategy of subsuming all the variations within the process component is extensively used. In Kannada, the suffix *'iru'* (which is also the verb 'be') indicates perfectual aspect. When combined with the past tense suffix 'id', it becomes *'idd'*. This single rule not only takes care of past perfect forms but also serves in the derivation of the paradigm of the verb *'iru'* itself. Normally *'iru'* is considered an irregular verb (Sridhar, S.N., 199O) and it is suggested that the paradigm be listed. *'iru'* is peculiar in the sense that there is an additional tense form: *'iddenu'* in past, *'idde:ne'* in non-past, *'irutte:ne'* in future/habitual and *'iruvenu'* also, future/habitual. This apart, generation of past tense forms makes use of the same rule: *'iru' + 'id' => 'idd'* and so you get *'iddenu'* etc. In Kannada maximum variations occur in the past tense forms and here is where we have saved.

3. MORPH: A Morphological Analysis/Generation Tool:

MORPH is an implemented system for morphological analysis and generation based on the network and process model. MORPH has been used to develop an analyser cum generator for Kannada. A lexicon with more than 10,OOO entries has been developed. MORPH uses a non-deterministic network and can generate multiple solutions where required through backtracking. Thus the verb form *'ma:Di'* can indicate either the plural imperative form or the past verbal participle of the root *'ma:Du'*. MORPH internally uses Roman notation but the input and output can be in Indian scripts - a GIST to Roman interconversion tool is part of MORPH. MORPH can generate a specified word form as also an entire paradigm. This stands in sharp contrast to other tools where one starts by

listing the paradigms by hand. MORPH has been implemented in PROLOG.

MORPH is currently being used as part an Machine Aided Translation System for translating from English to Kannada for the Government of Karnataka. MORPH's ability to do analysis as well as generation comes in very handy in the post editing tool. For example, using the post editing tool one can simply select the word *'toMdarege'* and substitute with *'kaSTakke'*. The word *'toMdarege'* is analysed for the root *'toMdare'* and singular dative inflection, *'toMdare'* replaced with the user selected word *'kaSTa'* and its dative singular form *'kaSTakke'* generated on the fly. The Kannada MORPH system is currently giving 60 to 70% performance on general texts. MORPH has also been tested for suitability for several other languages such as Telugu, Tamil and Bengali. Tamil MORPH is also giving 60 to 70% performance as on date.

4. Conclusions:

The division of labour between the network and the process components seems to be justified on several counts. The network component is declarative in nature. On the other hand, the processes of making and breaking saMdhi are more procedural. The order in which the various constraints are checked differs from analysis to generation. While generating a word form from a given root, we would know the grammatical and semantic features of the root to start with. When we are analysing a complete word form, these aspects can only be verified after an analysis is produced. Hence the process component needs parallel but distinct procedures for making and breaking saMdhi. On the other hand, the network component is inherently bidirectional. Thus the network and process model seems to hold promise as computational model for morphological analysis and generation.

The implemented system MORPH has demonstrated these advantages of the Network and Process model. While there are several other models proposed which appear to be similar on the surface, MORPH has some unique features that set it apart from the others:

- MORPH has been shown to be computationally viable and highly efficient
- provides for both analysis and generation
- can even generate complete paradigms
- provides automatic menus
- provides for input/output in Indian scripts

- segregates the declarative and procedural components, making it easier to develop grammars
- can easily handle complex saMdhi rules
- can incorporate a variety of constraints and exceptions
- can be phonologically enabled
- is suitable for a wide variety of languages
- MORPH systems have already been built and tested for several languages

More work is on to improve the Kannada MORPH system as also to develop a universal visual interface for the linguist using which morphological analysers and generators can be developed by linguists themselves without having to write any computer programs whatever.

References

1.Upadhyaya U.P., 1976, A Comparative Study of Kannada Dialects (Bellary, Gulbarga, Kumta and Nanjangud Dialects), Prasaranga, Mysore

2.Rajapurohit B.B., 1982, Accoustic Characteristics of Kannada, Central Institute of Indian Languages, Mysore

3.Sridhar, S.N., 199O, Kannada, Routledge

4.Hopcroft J.E., Ullman J.D., 1979, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley

5.Roche E., Schabes Y. (Eds), 1997, Finite State Language Processing, MIT Press