Finite-State Computational Morphology - Treatment of the Zulu Noun

L. Pretorius\textsuperscript{a} S. E. Bosch\textsuperscript{b}

\textsuperscript{a}Department of Computer Science and Information Systems, University of South Africa, Pretoria, South Africa, pretol@unisa.ac.za
\textsuperscript{b}Department of African Languages, University of South Africa, Pretoria, South Africa, boschse@unisa.ac.za

Abstract

Morphological analysis is a basic enabling application for further kinds of natural language processing, including part-of-speech tagging, parsing, translation and other high-level applications. Automated morphological analyzers exist for many of the European languages, but have not been reported for any of the indigenous languages of southern Africa. Our project in computational morphological analysis/generation includes the production of an automated morphological analyzer/generator for Zulu, using finite-state methods and tools. In this paper we elaborate on the use of finite-state methods in computational morphology, and report on our treatment of the Zulu noun.

Keywords: natural language processing, computational morphology, finite-state technology, morphological analysis, agglutinating languages, Zulu, noun
Computing Review Categories: 1.2.7

1 Introduction

Advances in research and in the production of sophisticated applications in natural language processing often rely on automated morphological analysis. Such applications include, for example tokenization, part-of-speech tagging, shallow syntactic parsing, and machine translation [4, 6, 7]. Computational aids for morphological analysis already exist for many European languages, including English, French, German, Spanish, Portuguese and Italian, while significant work has already been done for Basque, Turkish, Arabic, Finnish, Swedish, Norwegian, Danish, several East European languages, for example Hungarian, as well as for Swahili, a member of the Bantu language family\textsuperscript{1}.

The status quo according to [6, p. 96] is that morphological analyzers still remain to be written for all but the commercially most important languages.

This is also the case for Zulu and the majority of other languages in the Bantu language family which up to this stage have not received much attention in terms of natural language processing. The processing of these languages, which are characterized by complex morphological structures, particularly requires specialized tools for the automatic analysis of word-forms, as well as for most other electronic corpus-based analyses (see for example [12]).

Against this background, the aim of this article is to show how the challenges posed by an automated morphological analysis of a language such as Zulu can be addressed within the framework of finite-state methods. For the purposes of this discussion, we restrict ourselves to the morphological analysis of the word category noun in Zulu.

Section 2 of this paper provides a short exposition of the morphological structure of the Zulu noun and the language-specific challenges posed by a computational analysis of this word category. Section 3 gives an overview of some aspects of finite-state methods and tools that are relevant in computational linguistics in general, and in computational morphology in particular. In section 4 the use of finite-state technology (i.e. methods and tools) in automating the morphological analysis of the Zulu noun is demonstrated by means of a simple example. Finally, concluding remarks and proposals concerning future research possibilities are made.

2 The morphological structure of the noun in Zulu

The noun in Zulu is made up of two parts, namely a noun prefix and a noun stem. Nouns are typically categorized into eighteen noun classes, as determined

\textsuperscript{1}In linguistic studies “Bantu language family” refers to a specific family of languages which is spoken on the southern half of the African continent, the individual languages of which share common linguistic features. The term “Bantu” was introduced in language studies as far back as 1857 by a German philologist, Wilhelm Bleek.
by the noun prefixes. For example, a referent such as “person” is categorized differently from the referent “bread”, since the word for “person” is umuntu in Zulu, and uses the prefix uma-, while the word for “bread” is isinkwa, and uses the prefix isi-. These noun prefixes, have for ease of analysis, been divided into classes with numbers by scholars who have worked within the field of the Bantu language family, so that a noun such as umuntu is referred to as a class 1 noun, and a noun such as isinkwa is referred to as a class 7 noun. We shall follow Meinhof’s [15, 193:48ff] numbering system of the noun class prefixes, as listed in table 1.

The two examples given so far, would thus be morphologically analyzed as follows:

<table>
<thead>
<tr>
<th>Class</th>
<th>Prefix</th>
<th>Class</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>uma-</td>
<td>2</td>
<td>aba-</td>
</tr>
<tr>
<td>1a</td>
<td>u-</td>
<td>2a</td>
<td>o-</td>
</tr>
<tr>
<td>3</td>
<td>uma-</td>
<td>4</td>
<td>imi-</td>
</tr>
<tr>
<td>5</td>
<td>i(i)-</td>
<td>6</td>
<td>ama-</td>
</tr>
<tr>
<td>7</td>
<td>isi-</td>
<td>8</td>
<td>isi-</td>
</tr>
<tr>
<td>9</td>
<td>iN-</td>
<td>10</td>
<td>isiN-</td>
</tr>
<tr>
<td>11</td>
<td>u(lu)-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>u(bu)-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>uku-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>pha-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>uku-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>pha-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>uku-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>mu-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The eighteen noun classes

2.2 Noun stems

As already indicated, the noun in Zulu consists of a noun prefix and a noun stem. When we refer to the stem of a noun, we include

a) so-called primitive stems which cannot be reduced to a simpler form, and thus do not require suffixes, e.g. -ntu, -kwa; and

b) stems which are derived from other word categories, and which can be further analyzed into roots plus their suffixes, e.g.

2a: -akhi > -akhi- + i (root ‘build’ + suffix)

2b: -khulameli > -khulum- + -el- + i
   (root ‘talk/speak’ + suffix + suffix)

A noun stem may therefore be subdivided into a root plus suffixes. The root is generally regarded to be “the core element of a word, the part which carries the basic meaning of a word.” [16, p.170]. In the case of the noun, we consider the root to be the underived part that occurs after the noun prefix and which may therefore either be a part which does not require any suffixes for completeness, e.g. -ntu, -kwa: or it may be a part such as e.g. -akhi- or -khulum- which needs certain suffixes to form a complete noun. In view of the general definition of morphemes, namely that they are minimal meaningful units of which words are composed, it should be noted that roots are also regarded as morphemes. (Hyphens on either side of roots indicate where prefixes and/or suffixes need to be added for completeness). On morphological grounds we can therefore distinguish between two types of nouns:

- nouns formed from roots which do not require suffixes, and
- nouns formed from roots which do require suffixes.
2.3 Noun suffixes

Suffixes are morphemes which occur after a root or stem, such as for instance deverbal suffixes, diminutive and augmentative suffixes.

Nouns derived from verb roots, for example require a noun prefix as well as a deverbal suffix, in order to be complete nouns. The following are examples of nouns that can be formed from the verb root *khulum-* ‘speak, talk’:

3: *isi-khulum-i* ‘orator’
   *um-khulum-i* ‘speaker’
   *in-k(h)ulum-o* ‘speech’

The deverbal suffixes above are -i and -o. Note that such nouns may even have more than one suffix if the deverbal is formed from a verb root that has been extended, e.g.

4: *um-khulum-el-i* ‘advocate; one who incedes/speaks for’

The diminutive suffix *-ana* may be suffixed to noun stems in order to express ‘smallness’, ‘shortness’ or even “the young of”. With the suffixation of *-ana*, various phonological changes occur, depending on the nature of the final syllable of the noun stem. It is such phonological changes that need to be accounted for in the automatic morphological analysis of Zulu. Examples of nouns to which the diminutive suffix has been added, are:

5: *indlu* ‘house’ > *indlwna* ‘small house’
   *umfufa* ‘river’ > *umfudlana* ‘small river’
   *ufudu* ‘tortoise’ > *ufujana* ‘small tortoise’

Another suffix which may be added to noun stems is the augmentative suffix *-kazi*, which is sometimes used to express “bigness” or “greatness”, and in certain instances to express feminine gender, for example:

6: *intaba* ‘mountain’ > *intabakazi* ‘big mountain’
   *imnu* ‘sheep’ > *imnukazi* ‘ewe’

2.4 Challenges posed by a computational analysis of Zulu morphology

In the foregoing discussion on the morphology of the noun in Zulu, we have shown that the basis from which a noun is constructed, is the root. The root is the constant core element in words or word forms while the rest is inflection and derivation. Therefore, morphological analysis is essential for any kind of information retrieval. Let us look at the complex nature of nouns derived from the monosyllabic noun root *-zi* ‘village’, for instance:

7: *umazi* ‘village’
   *imazi* ‘villages’
   *emzini* ‘in the village’
   *umzana* ‘small village’

Without morphological analysis the identification of the noun root *-zi* ‘village’ in such examples is very difficult. Anyone with some knowledge of Zulu can imagine how many occurrences of the syllable *-zi-* one would come across by performing a simple find and replace exercise on a text corpus of one million plus words.

The following short text reveals a number of occurrences of the syllable *-zi-* (underlined), however, it is only these occurrences printed in bold which correspond with the root *-zi-* ‘village’.


So if for argument sake we wish to retrieve certain information from a corpus, for instance all occurrences of the monosyllabic root *-zi* in Zulu for purposes of lemmatization, frequency analysis, concordances, dictionaries or language learning, each word form in the corpus needs to be morphologically analyzed, before reliable feed-back can be obtained.

According to Hurskainen [8, p.633]

Many tasks in the process of retrieving language-specific information from text can be carried out in more than one way, by using suitable tools nowadays abundantly available. What cannot be substituted by commercial, shareware, or public domain tools is the morphological parser.

In other words, the morphological analysis needs to be language-specific. Morphological rules that are applicable to one language, are not necessarily applicable to another. In general, however, the specific challenges posed by a computational analysis of morphology are twofold, namely:

1) *morphotactics*, or word-formation rules, which means that morphemes that make up words cannot combine at random, but are restricted to certain combinations and orders. A morphological analyzer needs to know which combinations of morphemes are valid.

2) *morphological alternations*, which means that one and the same morpheme may be realized in different ways depending on the environment in which it occurs. Again, a morphological analyzer needs to recognize the correct form of each morpheme.
Table 2: Morphology of the noun

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Basic Prefix</th>
<th>Root</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>si</td>
<td>klahda</td>
<td>ana</td>
</tr>
<tr>
<td>u</td>
<td>mu</td>
<td>ntu</td>
<td>kazi</td>
</tr>
<tr>
<td>i</td>
<td>si</td>
<td>hlah</td>
<td>o</td>
</tr>
<tr>
<td>i</td>
<td>zi</td>
<td>thand</td>
<td>w+a</td>
</tr>
</tbody>
</table>

Let us have a brief look at some of the implications of morphotactics and morphological alternations in Zulu. Firstly, according to morphotactics or word-formation rules, every word needs to be analyzed into its morphological components, that is its lexical form/base or root as well as its inflectional/derivative affixes. There are numerous possibilities of morpheme concatenations in Zulu nouns for which possible legal combinations would need to be determined, as indicated in table 2. Noun roots/stems are preceded by a noun prefix which can be divided into a prefix and a basic prefix, and may be followed by nominal suffixes such as the diminutive, locative, augmentative etc. Furthermore noun class prefixes cannot combine freely with any noun stem.

Secondly, let us briefly consider morphological alternations, or the rules that determine the form of each morpheme in Zulu. Words are just concatenations of morphemes, but raw concatenation often gives us abstract ‘morphophonemic’ not-yet-correct words. There are ‘alternations’ between the raw concatenations and the desired final words.

If we look at the words in example 8, it might seem improbable to a non-Zulu speaker, that these two words actually share a common root morpheme, namely the lexical item -thab- ‘chance/opportunity’.

8a: ithaba (i(li)-thaba) ‘chance/opportunity’
8b: ithunshana (i(li)-thab(a)-ana) ‘small chance/ opportunity’

In words such as those in example 9, we are dealing with a root morpheme -nsima ‘field’ which occurs with two different suffixes.

9a: ensimini (e-nsim(u)-mi) ‘in the field’
9b: insingana (i(n)-nsima-ana) ‘the little field’

What makes it difficult to identify the root morpheme in the two sets of examples in 8 and 9 is the fact that certain morphophonological processes have taken place which render the method of comparing word forms and then identifying the indivisible, meaningful and recurrent element as the lexical item, impractical and virtually impossible.

A morphological analyzer would have to take certain morphophonological processes into account, that is where changes in the sounds of morphemes are based on surrounding phones. It would, for instance have to be specified in the analyzer that palatalization could occur with certain verbal extensions (passive), noun suffixes (diminutives) and so forth, as in examples 8 and 9.

Other morphophonological processes that would have to be provided for in the analyzer are vowel elision, vowel coalescence and consonantization which often occur across morpheme boundaries, e.g.

10a: Vowel elision
   inja-ana > injana

10b: Vowel coalescence
   na-ubaba > nobaba

10c: Consonantization
   abantu-ana > abantuwa

It will be shown that these challenges can be addressed within the framework of computational analysis by means of finite-state technology in which both morphotactics and morphological alternations are modelled between abstract underlying strings and their surface spellings. The result is a system (lexical transducer) that maps from surface words to analyses, and vice versa.

3 Finite-state technology for computational morphology

Finite-state technology (FST) has been the subject of extensive research and interest for the past almost fifty years. Its mathematical elegance and efficient implementation have been well documented in the vast literature available. In this section we give a brief overview of those finite-state notions that provide the basis for finite-state computational morphology. We focus on the ideas, the rigorous mathematical exposition may be found elsewhere (see, for example, [2, 14]).

Regular expressions were first defined by Kleene in 1956, in 1959 Rabin and Scott published their classical paper on finite automata, and the first reference to finite-state transducers was by Ginsburg in 1962 (see, for example, [3, 13]).

Definitions:
Regular expressions over the finite alphabet Σ and the regular sets they denote are defined recursively as follows:

1. φ is a regular expression denoting the regular set φ, or {}.
2. ε, the empty string, is a regular expression denoting the regular set {ε}.
3. a in Σ is a regular expression denoting the regular set {a}.
4. If p and q are regular expressions denoting regular sets P and Q, respectively, then

   p + q
   p . q
   p∗
(a) \((p \mid q)\) is a regular expression denoting \(P \cup Q\).
(b) \((pq)\) is a regular expression denoting \(PQ\).
(c) \((p)^*\) is a regular expression denoting \(P^*\).

(5) Nothing else is a regular expression.

A finite automaton or FA is a quintuple \((S, \Sigma, \delta, s, F)\):

(i) a finite set \(S\) of states, one of which is the start state, \(s\), and some are final states, \(F\),
(ii) an alphabet \(\Sigma\) of possible input letters, and
(iii) a transition function \(\delta\), which maps \(S \times \Sigma\) to \(S\).

Every FA defines or accepts some language, namely the set of all the strings which are accepted by the machine, i.e. all strings which will end up in a final state when they are run on the FA. An FA is a deterministic machine.

A finite-state transducer is a device much like a finite automaton, except that its purpose is not to accept strings or languages, but to transform input strings into output strings. Informally, it starts in a designated initial state and moves from state to state, depending on the input, just as a finite automaton does. On each step, however, it emits a string of zero or more symbols, depending on the current state and the input symbol. The state diagram (or network) for a deterministic finite-state transducer looks like that of a deterministic finite automaton, except that the label on the arrow (transition) looks like \(a/w\), or \(a : w\), which means ‘if the input symbol is \(a\), follow this arrow and emit output \(w\)’. The regular relation associated with a finite-state transducer is the set consisting of the pairs of strings \((a, w)\) which label its arrows.

In the 1950s significant mathematical results were obtained in formal language theory, including the following equivalence theorem:

**Theorem:**
The following statements are equivalent:

(1) \(L\) is a regular set.
(2) \(L\) is a regular (or right-linear) language.
(3) \(L\) is a finite automaton language.
(4) \(L\) is a nondeterministic finite automaton language.
(5) \(L\) is denoted by a regular expression.

The (recursive) definition of regular expressions (sets) implies that for any given alphabet arbitrarily complex regular expressions (or sets) may be obtained from the atomic ones by the repetitive use of the operations of choice \((\mid)\), concatenation and repetition \((\ast\), Kleene star\). By the equivalence theorem similar operations may also be performed on automata, as well as on transducers and relations. This means that we can build finite automata that accept arbitrarily large regular sets of strings - which is the case for the words in a real natural language.

Since computational morphological analysis / generation, by definition, involves both input and output, the appropriate finite-state device to use is the transducer. We therefore briefly discuss one particularly useful operation on transducers, viz. composition. (When a finite-state automaton is used in a composition, it is interpreted as a transducer, representing the associated identity relation.)

Let \(A\), \(B\), and \(C\) be sets, and \(R\) and \(S\) relations on \(A \times B\) and \(B \times C\), respectively. The composition of these two relations, denoted by \(R \circ S\), is a relation on \(A \times C\), such that

\[ S \circ R = \{(a, c) \mid \exists b \in B \exists (a, b) \in R \text{ and } (b, c) \in S\}. \]

The composition of two transducers \(T_1\) and \(T_2\) is again a transducer. It corresponds to the regular relation which is the composition of the regular relations associated with \(T_1\) and \(T_2\), respectively.

In theoretical computer science regular expressions and finite automata were studied in the context of the Chomsky hierarchy of grammars and languages, the formal basis for computer programming languages, compiler construction, and computability via Turing machines.

Apart from being mathematically elegant and pleasing, finite-state methods are of practical use. The set/rel/tion representation embodies the mathematical notion, the expressions are easily understood and manipulated by humans, and the automaton / transducer representation is suitable for efficient computer implementation [3, 5, 13, 17]. Early applications were in text manipulation systems, including lexical analyzers for compilers, text editors, and file manipulation systems.

Although the first application of finite-state transducers to natural language parsing was developed as early as 1958-1959 [9], Johnson (1972), Kaplan and Kay (1981 and 1994), Koskenmaki (1983) and Karttunen (1983) pioneered the use of finite-state transducers in computational morphology [4, 6, 11], which resulted in the Xerox finite-state calculus, “to this day the dominant finite-state paradigm”, [11, p.4]. The growing interest in finite-state methods in all aspects of natural language processing, is echoed by the increase in the variety of finite-state tools available for building and manipulating large-scale finite-state natural language systems [10]. The work reported on in this paper makes use of the Xerox finite-state software tools [4].

## 4 Applying FST to the Zulu noun

Research in computational morphology in organizations such as Xerox Research Centre Europe, is based on the fundamental insight that the complexities of
word-formation rules as well as morphological alternations, can be handled with the help of finite-state networks.

The Xerox finite-state calculus is a powerful, sophisticated, state-of-the-art ‘programming language’ for building finite-state solutions to a variety of problems in natural language processing. While a detailed discussion thereof is beyond the scope of this paper, we use a simple example to demonstrate its use in automating morphological analysis / generation.

4.1 A simple example

We introduce the Xerox finite-state tools by means of a simple example. Let us consider the one word language

\[ L = \{ \text{um fazi 'married woman'} \}. \]

Our aim is to build a computational morphological analyzer/generator for \( L \). In particular, we need to model the appropriate morphotactics and alternation rules for \( L \).

- The morphological analysis of \( \text{um fazi} \):
  
  \[ u-, \quad \text{noun preprefix, class 1} \]
  
  \[ -mu-, \quad \text{noun prefix, class 1} \]
  
  \[ -fazi, \quad \text{noun stem}. \]

- The alternation rule: In the case of polysyllabic noun stems, the basic noun prefix \(-mu-\) is reduced to the shortened form \(-m-\).

The allowed or legal combinations of morphemes (i.e. morphotactics) as well as the alternation rules can be encoded as finite-state networks, the automation of which we will discuss below.

The morphological analysis tags we use are shown in table 3.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPrePre1</td>
<td>noun preprefix class 1</td>
</tr>
<tr>
<td>NPrePre2</td>
<td>noun preprefix class 2</td>
</tr>
<tr>
<td>NPrePre5</td>
<td>noun preprefix class 5</td>
</tr>
<tr>
<td>NPrePre11</td>
<td>noun preprefix class 11</td>
</tr>
<tr>
<td>BPre1</td>
<td>basic prefix class 1</td>
</tr>
<tr>
<td>BPre2</td>
<td>basic prefix class 2</td>
</tr>
<tr>
<td>BPre5</td>
<td>basic prefix class 5</td>
</tr>
<tr>
<td>BPre11</td>
<td>basic prefix class 11</td>
</tr>
<tr>
<td>WRoot</td>
<td>word root</td>
</tr>
<tr>
<td>NomSuf</td>
<td>nominal suffix</td>
</tr>
<tr>
<td>DimSuf</td>
<td>diminutive suffix</td>
</tr>
<tr>
<td>CausExt</td>
<td>causative extension</td>
</tr>
</tbody>
</table>

Table 3: Morphological analysis tags

4.1.1 Automating the morphotactics

The Xerox software tool for specifying the required natural-language lexicon and the morphotactic structure of the word in this lexicon, is called \texttt{lexc}, for

lexicon compiler. A \texttt{lexc} source file is produced with a text editor and compiled into a finite-state network. The \texttt{lexc} source file, example.lex, for our example is shown in example 4.1.

```
! example.lex
Multichar_Symbols [NPrePre1] [BPre1]
    [WRoot] "U "MU "ER
LEXICON Root
    NomPrefixes;
LEXICON NomPrefixes
    u[NPrePre1]mu[BPre1]:"U"MU WRoot;
LEXICON WRoot
    fazi NomSuf;
LEXICON NomSuf
    [WRoot]:"ER NomSufContents;
LEXICON NomSufContents
    #;
```

Example 4.1: The \texttt{lexc} input file, example.lex.

A closer look at this file shows the intuitive correspondence between the cascading LEXICONS and states of a network, while the individual entries may be thought of as representing the labels on the arrows of such a network.

This network generates morphotactically well-formed but still rather abstract strings. Such strings are sometimes called ‘underlying’ morphophonemic or, in finite-state tradition, lexical strings. Rules are then needed to map the abstract lexical strings into properly spelled surface strings, as they occur in natural language.

4.1.2 Automating the alternation rule

The alternation rule is formulated as a regular expression, and compiled into a finite-state network by means of the Xerox tool \texttt{xfst}. Our \texttt{xfst} source file, example.script, is shown in example 4.2.

```
! example.script
define Cons [b|c|d|f|g|h|j|k|l|m
    |n|p|q|r|i|s|t|v|w|x|y|z];
define Vowel [a|e|i|o|u];
define Syllable [Cons+ Vowel];
define ruleumu
    %"MU -> m || _ [Syllable Syllable %"ER
    | Syllable %"ER Syllable]  
    .o. %"U -> u
    .o. %"MU -> m || _ Vowel
    .o. %"MU -> m u
    .o. %"ER -> 0;
read regex ruleumu;
```

Example 4.2: The \texttt{xfst} input file, example.script.

In the regular expression implementing our alternation rule we need to differentiate between monosyllabic and polysyllabic noun stems in order to determine the appropriate form of the basic noun prefix.
The Xerox construct used in this instance is conditional replacement.

4.1.3 Building and using the analyzer/generator

As a final step in the development of this specific type of morphological analyzer, the two finite-state networks are then combined together into a single network, namely a so-called lexical transducer, that contains all the morphological information about the language being analyzed, including derivation, inflection, alternation, compounding etc.

We use composition to combine these two networks, yielding the final finite-state network, stored as file example.fst. Finally we apply our morphological analyzer / generator to the word of language L, obtaining

\[ u[NPrePre1]mu[BPrel]fazi[WRoot] \]

from

\[ umfazi, \]

and vice versa, as expected.

4.1.4 An interactive session

In example 4.3 we show an interactive session for building a morphological analyzer / generator for the language L, which includes all the user inputs, but only selected system responses. The prompts are in boldface, the system responses in sans serif, and the user inputs in typewriter font. The \$ is the Linux prompt.

```
$ xfst
xfst[0]: source example.script
xfst[1]: save stack exrul.fst
xfst[1]: quit
$ lexc
lexc> compile-source example.lex
lexc> read-rules exrul.fst
lexc> compose-result

... Done.
29 states, 29 arcs, 3 paths.
Minimizing ... Done.
17 states, 17 arcs, 3 paths
lexc> save-result example.fst
lexc> lookup umfazi
Use (s)ource or (r)esult [r]: r
NOTE: Using RESULT.
ul[NPrePre1]mu[BPrel]fazi[WRoot]
lexc> lookupdown ul[NPrePre1]mu[BPrel]fazi[WRoot]
Use (s)ource or (r)esult [r]: r
NOTE: Using RESULT.
ulmazi
lexc> quit
$
```

Example 4.3: Building a morphological analyzer / generator for L: an interactive session.

While this example demonstrates the steps followed in modelling the morphological structure of the simple language L, it does not reflect the increasing complexity of the modelling process as the language L is extended to include the entire Zulu vocabulary. The research challenge is to use the capabilities of the Xerox finite-state tools to model the morphological structure of Zulu in such a way that all the Zulu words are included (generated) and correctly analyzed, but that all character strings that do not represent words in the real language be excluded.

4.2 Towards a prototype

Our ultimate aim is that the lexc input should define a comprehensive lexicon of the Zulu language, as well as all the word formation rules (morphotactics) that apply in the language. The xfst input should contain all the alternation rules that are required to produce well-formed strings in the Zulu language.

At present our prototype provides for Zulu nouns based on a selected lexicon, which covers most of the identifiable challenges such as morphotactics and morphological alternations. Once the noun prototype is evaluated and proved to be functional, the lexicon will be extended, not only to include all nouns in the language but also to deal with extensions to the noun by means of various prefixes such as the copulative and adverbial prefixes, as well as possessive concords.

The following are examples of types of nouns that are already being handled by the current prototype. The symbol ↓ indicates the bidirectional use of the morphological analyzer / generator:

```
a) ikati
   ↓
   i[NPrePre5]li[BPrel]kati[WRoot],

b) abantwana
   ↓
   a[NPrePre2]ba[BPrel]ntu[WRoot]
       ana[DimSuf],

c) ufuwana
   ↓
   u[NPrePre1]lu[BPrel]fudu[WRoot]
       ana[DimSuf],

d) umfundisi
   ↓
   u[NPrePre1]mu[BPrel]fund[WRoot]
       is[CausExt]i[NomSuf]
```

5 Concluding remarks

The results of our initial research show that finite-state technology may be used successfully to implement morphological analysis / generation of nouns in an agglutinating language such as Zulu.

The main challenges posed by the complexity of nouns, namely morphotactics and morphological al-
ternations, can both be handled by means of finite-state networks which, when composed, result in a so-called lexical transducer containing all the morphological information of the Zulu noun.

Future research will aim at extending the functional noun prototype to other word categories, and at the same time developing a comprehensive (electronic) lexicon of the Zulu language. This work on the computer analysis of the Zulu noun should therefore be regarded as an initial step towards the much bigger goal of a comprehensive automated analyzer / generator of Zulu morphology, and eventually of other indigenous languages spoken in southern Africa.

Indeed, the development of African languages at technological level is of crucial importance, especially in the light of Cole’s [6, p.xvi] prediction that

Languages for which no adequate computer processing is being developed, risk gradually losing their place in the global Information Society, or even disappearing, together with the cultures they embody, to the detriment of humanity’s great assets: its cultural diversity.

6 Acknowledgements

The authors would like to acknowledge the financial support of the National Research Foundation (NRF) for the project Computer Analysis of Zulu Morphology; the initial financial support from the University of South Africa’s Grant for Foreign Visits, and the encouragement and enthusiasm of Prof. Chris Swanepoel, Vice-Dean of the Faculty of Arts, University of South Africa; the contribution of Xerox Research Centre Europe (XRCE), whose software and training are a vital component of this project; the continued support and expert advice of Dr. Ken Beesley (XRCE); the encouragement and additional financial support of Prof. Paula Kotze, director of the Centre for Software Engineering (CENSE) and head of the Department of Computer Science and Information Systems, University of South Africa.

References


