Nordic Journal of African Studies 19(1): 1–22 (2010)

Finite State Methods in Morphological Analysis of Runyakitara Verbs

Fridah KATUSHEMERERWE
Makerere University, Uganda
&
Thomas HANNEFORTH
Potsdam University, Germany

ABSTRACT

Previously, there has been a lack for an automatic analyser and generator for the word forms of Runyakitara. In this paper, we present a computational model for grammatical Runyakitara verbs. This model, RUNYAGRAM, is based on freely-available open-sourced finite-state methods and, in particular, the fsm2 interpreter. It captures the morphotactic structures with non-recursive context-free grammars supported by fsm2 and morpho-phonological alternations with a finite composition of commonly used context-dependent string rewriting rules. Their combination results into a finite state transducer that can be exported and used in numberless software-developing platforms. The obtained transducer is an important building-block that can be employed in comprehensive morphological analysers, syntactic parsers, spell-checkers, text-to-speech synthesizers, and machine translation systems. Currently, 86% of the verb forms are recognized. It is possible to increase the coverage, or alternatively, to adapt the approach of the RUNYAGRAM system to related languages.

Keywords: *morphological analysis, finite state methods, Runyakitara verb.*

1. INTRODUCTION

One of the core enabling technologies required in natural language processing applications is a morphological analyzer. It is an established fact in computational linguistics that a morphological analyzer is a starting point for many natural language processing applications (Pretorius & Bosch, 2003; Yona & Wintner, 2005).

Computational morphology deals with automatic word-form recognition and generation. The general challenges posed by a computational morphological analyzer, as described by Prestorious and Bosch, (2003), are twofold:

- 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 (morphotactics) are valid.

- Morphemes may be realized in different ways depending on their context. A morphological analyzer needs to recognize the morphophonological changes between lexical and surface forms (morphophonological alternation). Automatic morphological analyzers and generators must take into consideration the above issues.

Comprehensive morphological analyzers are available for well documented languages such as English, Swedish, German, Arabic, and Finnish (Karttunen & Beesley, 2005:77). Considerable work has also been achieved in employing finite state methods for Bantu language analysis: the Kiswahili morphological analyzer (Hurskainen, 1992; 1996; 2004); the Zulu analyzer prototype (Pretorius & Bosch, 2003), Lingala verb morphology (Karttunen, 2003), Ekegusii verb morphology (Elwell, 2005), Kinyarwanda (Muhirwe & Trosterud, 2008), and Setswana verb morphology (Pretorius, Berg, & Pretorius, 2009).

However, given the fact that Bantu languages are more than five hundred in number, almost all are still not treated. Although Bantu languages are classified as largely agglutinative and exhibit significant inherent structural similarity, they differ substantially in terms of their phonological features implying that each Bantu language requires an independent morphological analyzer.

Runyakitara is one of those under-resourced Bantu languages with no computational morphology. Bernsten (1998) splits Runyakitara into four major dialects: Runyankore, Runkiga, Runyoro, and Rutooro. Guthrie (1967) groups these four dialects into two languages belonging to Narrow Bantu branch of the Niger-Congo family, Nyankore-Kiga (E.13) and Nyoro-Ganda (E.11). For purposes of this paper, Runyakitara will be taken to mean two major language clusters mentioned above: Runyoro-Rutooro and Runyankore-Rukiga, denoted by R-R in the following.

Runyakitara is spoken by approximately six and half million (6,500,000) people in nineteen districts of Western Uganda. As a major language in Uganda, some parts of Tanzania and Democratic Republic of Congo, it is important that R-R is given computational attention because it has a large number of speakers, a language of media in western Uganda (two regular newspapers – one online) and a rich history and culture which should be preserved. Besides the language is a medium of instruction in lower levels of primary education in Western Uganda and we shall consider how computational efforts may add value to the language education. The morphology of a verb in R-R, as has been stressed by other Bantu researchers, (Hurskainen, 1992; Elwell, 2005) is one of the complex morphological systems known which means that it needs special attention.

2. RUNYAKITARA VERB MORPHOLOGY AND THE COMPUTATIONAL CHALLENGE

A verb in a typical Bantu language will take on many prefixes and suffixes. The Runyakitara verb morphology poses the following challenges to computational modeling: a. number of morphemes, b. morpheme order, c. morpheme combination, d. allomorphs, and e. vowel harmony. These are discussed in the sub-sections below.

2.1 NUMBER OF MORPHEMES INVOLVED

The Bantu verb template described in many studies suggests about 8 to 15 morpheme slots as follows:

Slot	**1**	**2**	**3**	**4**	**6**	**7**		**8**	**9**
Meaning	Pre-initial	Initial	Post-initial	Tense marker	OM	Verbal base		Final	Post-final
Morpheme	NEG	SM	NEG	Tense	Object marker	Root	Verb ext.	Mood, aspect, NEG	

Table 1. Bantu Verb Template (Nurse & Philippson, 2003).

The above generic template raises many questions if one considers it with respect to R-R morphology: what is considered a morpheme on the template? If verb extension, (in Slot 7) is a morpheme, does it mean that such extensions as causative, applicative, passive, etc are allomorphs of the same morpheme? This and many other questions prompted us to devise an R-R verb template to cater more specifically for a number of morphemes present in the language.

There are many morphemes involved in the formation of R-R verbs; therefore, it is important to expand the template. These can be broadly classified as prefixes, (morphemes left of Slot 0) root (Slot 0) and suffixes (morphemes after Slot 0). The following template shows morphemes involved in the formation of Runyakitara verbs:

-7	-6	-5	-4	-3						-2		-1	0	1							2			3	
Ng1	Asp	Sp	Ng 2	Tense/aspect markers						Object pronous		Asp	R	Verb extension morphemes (VEXT)							Verb end (VE)			Pf1	Pf2
				Inf	Hab		Pf	ff	Rp	Op1	Op2	ref		Ca	Apl	Rec	Pas	Int	Stat	Rev	Ind	subj	past		
ti	ni	18	ta	ku	Ø		aa	ria	ka	18	18	e		es is iz y	er ir	an	w ebw ibw	erer irir	ek ik	uk ur uur	a	e	ire	ho mu yo	ga

Table 2. Runyakitara Verb Template.

Note: *Slot 0 represents root, to the right of 0 are suffixes to the root. Slot 1 is for verb extensions as: Ca – causative, Apl – applicative, Rec – reciprocal, Pas – passive, Int – intensive, Stat – stative, Rev – reversive. Slot 2 represents Verb end: Ind – indicative, subj – subjunctive, past – past tense. Slot 3 indicates post final morphemes: pf1 – post-final 1; pf2 – post-final2. On the left of zero, -1 Asp – aspect, -2 – object pronouns, -3 Tense/aspect markers, -4 – Ng2 – Negative 2, -5 Sp – subject prefix; -6 Asp – aspect; -7 Ng1 – Negative 1. For a more description and examples, refer to Appendix A.*

Runyakitara has typical characteristics of template morphology as it is outlined by Spencer. As observed by Spencer (1991), template morphology poses a computational challenge. According to Spencer, template morphology is a morphological system where a verb stem or root consists of obligatory affix(es) as well as a set of optional affix(es). The combinations of morphemes make automatic analysis difficult because one has to sort out first which affixes fit to the root to form specific verb forms.

Adding to the number of morphemes involved, subject and object pronouns mark agreement with the noun classes in question. In case the subject is not included, they serve as subject and object pronouns. These markers appear on the verb root as prefixes to the root. R-R has eighteen (18) noun classes, therefore subject and object pronouns add up to 18 in each case. In addition, R-R is a type 3 language according to the classification given by Maho (2007), which means that it allows two or more objects in the construction. Evidence in Runyakitara shows that the language can have a double object construction, that is, a verb can have a marker for both direct and indirect objects in the same construction. An example in this case is *mu-mu-n-kwat-ire* (you grab/hold him for me), where *mu-n* indicate double objects representing **him** and **me**. This will add to the number of morphemes, indicating that a number of morphemes is large enough to pause a challenge.

2.3 Morpheme Combination

Much as some studies have been carried out on combination of morphemes in Bantu languages, (Hayman, 2007) limited research is available for Runyakitara morpheme combination. This is specifically in reference to verb extensions. As earlier noted by Hayman, (2007) verb extensions are difficult to analyze mainly because of various functions and also, they are numerous and often occur in long successions. Runyakitara has seven (7) verbal extensions which can be added to the root individually or in combination. For example, one can have a verb with verb extensions such as:

reeb-a (see)
reeb-es-a (see with),
reeb-an-a (see each other),
reeb-w-a (be seen),
reeb-es-an-a (make each other to see),
reeb-an-is-a (make to see each other),
reeb-es-an-is-ibw-a (be made to make them see each other).

In the last example, [***es, an, w, is, ibw***] are all verb extensions playing different roles. The order of causative morphs ***es*** and ***is*** in the above example is different and there is no study available that has established the combination of verbal extensions in Runyakitara, and the order in which they can follow one another.

2.2 MORPHEME ORDER

Although the Bantu verb template is presumed to present a fixed order of morphemes, and provides Slot 4 in Table 1, for example, as a slot for tense aspect markers, some morphemes in Runyakitara violate the order. Specific cases are: progressive ***ni,*** reflexive ***e*** and past ***ire*** which violate the order of Bantu template. As indicated on Runyakitara template in Table 2, ***ni*** comes before the subject marker in the construction while other tense/aspect markers follow the subject marker e.g.

ni*-ba-mu-reeb-a* (they are seeing him)
*ba-****ka****-mu-reeb-a* (they saw him [last year or some months back]).
*Ba-mu-reeb-****ire*** (they saw him [yesterday])

In the above verb constructions, ***ni, ka***, and ***ire*** are tense/aspect markers but appear in different positions in respect to the root.

Also, the order of verb extensions on the template does not necessarily mean that it is the order of their construction. That is to say, extensions can attach to verbs depending on the argument structure. So, there is not fixed order in which they are supposed to appear in the construction of the verb. For example, a verb root

reeb-a	(see)
reeb-es-a	(see with)
reeb-an-a	(see each other)
reeb-es-an-a	(make each other to see)
reeb-an-is-a	(make … to see each other)
reeb-an-is-ibw-a	(be made to make … see each other).
reeb-er-a	(see for)
reeb-er-an-a	(see for each other)

All this indicates that there is a lot of flexibility regarding which morphemes precede and follow one another because is and es are all causatives.

2.4 ALLOMORPHY

Runyakitara has various allomorphs, that is, different realizations of the same morphemes. A case in point here is a causative morpheme which has four different realizations [es/is/iz/s/y]. Applicative, passive, stative and reversive morphemes are no exception. All these pose a challenge to computational modeling.

2.5 Vowel Harmony

Katamba, (1984) analyses vowel harmony of verb extensions in Luganda, a language closely related to Runyakitara. His analysis, which classifies the vowels involved in harmony as mid and nonmid gives an understanding of existence of vowel harmony in the language but does not aid much when it comes to formalizing morphemes for computational purposes. The reason is that it is difficult to identify the location of mid and nonmid vowels in the string. The suggestion provided by Morris and Kirwan (1972) of a penultimate syllables is useful here. Penultimate syllable is a syllable preceding the final. Penultimate, which means before last, can easily aid one to locate the vowel in question. For example, in the word ***bo-ro-go-ta***, (flow of water) the penultimate is '**go**'. This aided in understanding that when a penultimate syllable is **e, o**, the causative extension will be ***es***. On the other hand, when the penultimate syllable is ***a, i*** or ***u***, the causative extension will be ***is*** or ***iz***. The same applies to applicative, intensive and stative.

Given the nature of Runyakitara morphology, it was important to carefully select the formalization approach appropriate to the structure. Therefore, a phrase structure grammar was identified to appropriately handle the concatenative nature of Runyakitara morphology. Rules proposed by Selkirk (Spencer, 1991), were applied, written as W+A for suffixing; and A+W for prefixing. However, it was clear that, the rules Selkirk proposed only account for concatenative nature of morphology. It was important therefore to also think of the way of handling morpho-phonological and orthographical processes. However, they are helpful for Runyakitara concatenative morphology.

3. Formalization and Implementation

Given the nature of Runyakitara morphology, it was important to carefully select an appropriate approach. The concatenative nature of Runyakitara morphology can be captured with Phrase Structure Grammar (PSG) along the lines of Selkirk (Spencer, 1991) who proposes phrase-structure like rules written as W+A for suffixing and A+W for prefixing. However, it was clear that the rules Selkirk proposed only account for concatenative morphology. It was important therefore to also think of the way of handling morpho-phonological and orthographical processes. Because recursion is not needed, we describe both the concatenative rules and phonological processes in the framework of finite-state acceptors (FSA) / transducers (FST). Our approach relies heavily on the closure properties of these automata under intersection, composition, and substitution (see Hopcroft & Ullman, 1979, Kaplan & Kay, 1994).

The implementation is done using *fsm2* (Hanneforth, 2009), a scripting language within the framework of finite state technology. Finite-state technology is considered the preferred model for representing the phonology and morphology of natural languages (Wintner, 2008). The model has been used to computationally analyze natural languages such as English, German, French, Finnish, Swahili, to mention a few cases (Beesley and Karttunen, 2003), and its main advantage is that it is bidirectional – it works for both analysis and generation. It is on this basis that the technology was selected to be applied on the morphological grammatical analysis of R-R.

Fsm2 was chosen as a resource tool for a morphological grammar of R-R due to a number of reasons:

- It supports a full-set of algebraic operations defined on both unweighted and weighted finite state automata and weighted finite state transducers (Hanneforth, 2009). Algebraic operations are useful to design complex morphological analyzers in a modular way.
- *fsm2* supports a number of equivalence transformations which change or optimize the topology of a weighted automation without changing its weighted language or relation, that is an automata, can be minimized, determinized, optimized etc;
- *fsm2* uses symbol signatures which map symbols to numbers that are internally recognized by the automata. Symbol signatures are useful in language modeling since every word in a language constitutes an alphabet symbol, and a task of a developer is to define symbols that represent morpheme and their categories.
- *fsm2* provides an efficient way of compiling morphological grammars where the co-occurrence of roots and inflectional affixes common in Runyakitara is easily accounted for.
- *fsm2* is open-source software. The source code can be downloaded at *www.fsmlib.org*.

fsm2 is able to load lexicons, grammars and replace rules defined by the morphology developer. It is able to automatically transform various rule formats into transducers.

3.1 THE STRUCTURE OF RUNYAGRAM

RUNYAGRAM is built on a modular basis comprising of a special symbol module/file, a grammar module and a replacement rule module. The three are composed together, and the result is a single finite state transducer.

The following diagram demonstrates the overall architecture of RUNYAGRAM:

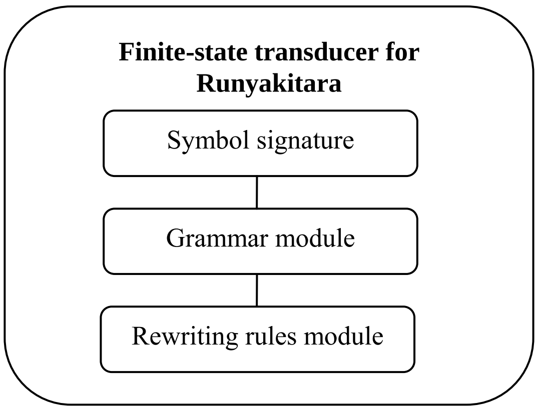

Figure 1. Sketch of the architecture of RUNYAGRAM.

The output RUNYAGRAM generates can be used as an input for other applications such as:

- a spell checker for Runyankore-Rukiga
- a dictionary since RUNYAGRAM outputs lemmas
- a syntax analyzer for Runyakitara
- a language learning system for vocabulary and grammar depending on how it can be developed.

The remaining subsections illustrate the construction of the subanalyzer for verbs in Runyakitara. The subanalyzer for nouns is created in a similar fashion.

3.2 Symbol Signature

Like the AT&T Lextools (see Roark and Sproat, 2007), *fsm2* uses a *symbol signature* to define the basic entities of the grammatical description. Fig. 2 shows some sample entries.

```
Letter     a b c d e f g h i j k l m n o p q r s t u v w x y z
Category:  VERB_ROOT_SIMPLE1 Simple1
Category:  VERB_PREF_TENSE Tense
```

Figure 2. Sample entries of the RUNYAGRAM symbol signature.

The entries are of one of two types:

1. **Supertype** – **subtype** definitions
2. **Category** definitions. A category consists out of a category name and a (perhaps) empty list of features.

The first line in Fig. 2 defines `Letter` as the supertype of the subtypes `a`, `b`, `c`, etc. The following lines define two categories `VERB_ROOT_SIMPLE1` and

VERB_PREF_TENSE, with features Simple1 and Tense defined elsewhere in the signature. Features themselves are again treated as supertypes, having their subtypes as their values.

Each symbol in the signature – whether type or category name – is mapped by *fsm2* to a unique integer used internally in the compiled automata.

3.3 WORD GRAMMAR

For specifying the morpheme order, we do not use the "classical" *continuation-class mechanism* of Koskenniemi (1984), but instead employ a *context-free word grammar* for that purpose[1]. We think that using a grammar is a much more natural way of defining orderings and groupings of elements compared to the continuation-class method which basically amounts to hand-coding a finite state automaton within the lexicon. Since the generative capacity of context-free grammars is beyond the one inherent in finite-state automata, we restrict ourselves to a subset of context-free grammars along the lines of quasi-context free grammars by Mohri & Pereira (1996). This subset may include left- or right recursive rules, but rules out all forms of center-embedding.

In the *fsm2* framework, grammar rules have the form $A \rightarrow \beta$, where A is a designated nonterminal symbol and β is an arbitrary regular expression (which may even use intersection or negation).

The compilation approach is based on an ordering of the nonterminals of the grammar, creating finite-state automata (FSA) for each grammar symbol and substituting the FSA for the individual grammar symbols into the rules right-hand sides in the previously computed order. In the grammar rules right-hand sides, morphemes of Runyakitara are interleaved with grammatical categories bearing grammatical information for the morphemes preceding them.

The grammar module consists of a set of quasi context-free rules accounting for the concatenative nature of Runyakitara morphology. The grammar contains a large number of rules, but we present a sample, exemplifying the principles underlying the overall organization of the grammar. We follow the approach of taking a verb from a minimum number to a maximum number of morphemes. This was done to ensure that every verb form is accounted for. Fig. 3 shows some (simplified) sample rules of the verb sub-grammar.

[1] A *context-free grammar* (see Hopcroft & Ullman, 1979) is a 4-tuple $\langle\Sigma,N,S,P\rangle$ where Σ is a finite set of alphabet symbols, N is a finite set of non-terminal symbols (phrase symbols), $S \in N$ is the start (sentence) symbol of the grammar and P is a set of rules $A \rightarrow \beta$, where $A \in N$ and $\beta \in (N \cup \Sigma)^*$. That means: the left-hand side of a grammar rule is restricted to a single phrasal symbol, whereas the right-hand side can contain an arbitrary combination of alphabet and phrasal symbols.

```
# Verb structure rules

# Minimum number of morphemes a verb takes
[VERB]        →    [VROOT] [VEND]

# Maximum number of morphemes a verb takes
[VERB]        →    [VPREFNEG] [VPREFSP] [VPREFTM] [VPREFOP] \
                   [VROOT] [VEXT] [VEND] [POSTV]

# Morpheme insertion rules (morphemes are in bold-face)
[VROOT]       →    (reeb|teer|kwat|shom)\
                         VERB_ROOT_SIMPLE Simple=simpleverb]
[VEND]        →    a     [VERB_END_IND Ind=mood]
[VPREFNEG]    →    ti    [VERB_PREF_NEG Neg=polarity]
[VPREFSP]     →    a     [VERB_PREF_SPM3S Spm3s=agrmt3]
[VPREFTM]     →    aa    [VERB_PREF_PERF Perf=perfective]
[VPREFOP]     →    bu    [VERB_PREF_OPM13 Opm13=objectprefix13]
[VEXT]        →    es    [VERB_PREF_CAUS Caus=causative1]
[POSTV]       →    mu    [VERB_SUFF_POST Post=postverbal]
```

Figure 3. Sample rules of the verb grammar (Nonterminals are enclosed in square brackets: `[VPREFNEG]` = verb prefix negative; `[VPREFSP]` = verb subject prefix; `[VPREFTM]` = verb prefix tense marker; `[VPREFOP]` = verb prefix object marker; `[VROOT]` = verb root; `[VEXT]` = verb extension; `[VEND]` = verb end; `[POSTV]` = Verb suffix post verbal. Symbols after morphemes in bold-face indicate categorical information. | means disjunction.)

To compile a grammar like the one in Fig. 3 into an unweighted finite-state acceptor, the grammar rules are converted into a directed graph according to the following principle: for all nonterminals A and B, if there exists a rule $A \rightarrow \ldots B \ldots$, then the graph contains an edge $A \rightarrow B$. After this preprocessing step, a *topological order* (cf. Cormen et al., 2001) of the resulting graph is computed (in case the graph is cyclic – which means that the underlying grammar is recursive – the (acyclic) *component graph* of all *strongly connected components* is used instead[2]). All the right hand sides of all grammar rules which share the same left-hand side are disjunctively combined and for every nonterminal A we compute a finite-state acceptor $FSA(A)$ representing all the right-hand sides for A. In a final step, each nonterminal A is substituted by its corresponding FSA in *reverse topological order*, beginning with the FSAs for the grammar rules which do not have further nonterminals in their right-hand sides. Note that the grammar need not be in a special format (right-linear etc.) to apply this procedure.

To illustrate these steps, Fig. 4a shows the FSA for nonterminal `VERB`, while Fig. 4b shows the FSA for `VROOT` according to our grammar fragment. The FSA

[2] At this stage, also the regularity check takes place: all nonterminals in a strongly connected component (there may be more than one in case of mutual recursion) must occur *either right- or leftlinear* in the subgrammar restricted to these nonterminals. This for example rules out rules like $S \rightarrow a\ S\ b \mid c$ which generates a non-regular language.

for VROOT of Fig. 4b is substituted into the one in Fig. 4a, replacing the two occurrences of VROOT (transitions 0 → 1 and 5 → 6). All other symbols in Fig. 4a are replaced in a similar way by their corresponding automata, yielding a finite-state acceptor representing the whole grammar fragment.

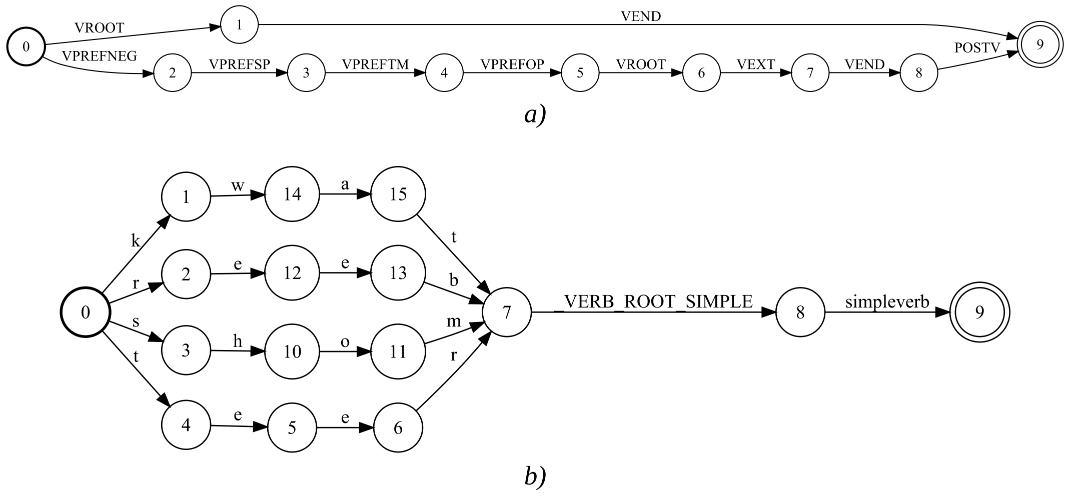

Figure 4. FSAs corresponding to grammar rules of Fig. 3. a) FSA for VERB, b) FSA for VROOT.

The grammar fragment in Fig. 3 accounts for verb forms like *teera* (beat), *reeba* (see), *kwata* (catch) and *shoma* (read). However, we need also to cater for *shutama* (sit), *gyenda* (go), etc, which are not indicated in the fragment. The grammar fragment is simplified since it would be computationally too expensive to include the complete set of Runyakitara verb stems, because it would result in grammars with several ten thousand rules. We therefore partitioned the set of verb stems into eight equivalence classes, each class containing all verb stems which participate in the same word-grammatical constructions and represented by a unique symbol in the grammar. After compiling the word grammar into a finite-state acceptor $\boldsymbol{A_G}$, a final processing step then substitutes each equivalence class denoting symbol by the set of its corresponding verb roots. This also simplifies adding new verb roots, since the grammar automaton remains unchanged and only the final substitution has to be recomputed. Nevertheless, the compilation of the grammar with approx. 330 rules with subsequent substitution takes less than a quarter of a second on a modern CPU, resulting in a finite-state acceptor with ≈ 800 states and ≈ 1,200 transitions.

The language (in the technical sense) generated by the grammar is still a set of morpheme concatenations forming strings but some are still abstract concatenations (morphotactics) without proper phonological and orthographical representation. Fig. 5 shows some strings described by the grammar.

```
a     [VP_SPM3S Spm3s=agrmt3s]
aa    [PERF Perf=perfective]
bu    [VP_OPM13 Opm13=agrt13]
reeb  [VERB_ROOT_SIMPLE]
a     [VERB_END End=indicative]

a     [VP_SPM3SSpm3s=agrmt3s]
aa    [PERF Perf=perfective]
bu    [VP_OPM13 Opm13=agrt13]
reeb  [VERB_ROOT_SIMPLE]
a     [VERB_END End=indicative]
mu    [POST Post=postverbial]
```

Figure 5. Some elements of the language generated by the verb grammar (morphemes are in bold face, strings like `End=indicative` indicate feature-value pairs).

Both **`a-aa-bu-reeb-a`** and **`aa-bu-reeb-a-mu`** are valid underlying forms in Runyakitara, representing correct grammatical information, but are not correctly spelt and well pronounced words. The grammatical forms are **`yaabureeba`** and **`yaabureebamu`**. This calls for a change in the first **a** to **y**.

To deal with this kind of allomorphic variation, we switch from the *Item-and- Arrangement* model inherent in the grammar approach to a more process-oriented *Item-and-Process* model (see Hockett, 1954 for a description of these models).

3.4 Context-Dependent Rewriting Rules: Morpho-Phonological and Orthographical Rules

Rewriting rules cover morpho-phonological and orthographical issues and are of the abstract form:

$$\alpha \rightarrow \beta \;/\; \gamma \;\_\; \delta$$

This means that an instance denoted by α is replaced by an instance β, if α is preceded by a γ and followed by a δ. It is well-known (Johnson, 1972, Kaplan & Kay, 1994) that rules of this kind stay within the realm of regular devices if certain conditions apply: (i) α, β, γ and δ must denote regular languages and (ii) rules are not allowed to apply on their own output.

For example, the replacement rule

```
a  → y  /  _  [VP_SPM3S Spm3s=agrmt3s] aa [PERF Perf=perfective]
```

states that **a** is replaced by **y**, whenever **a** (a verb prefix marker for third person singular) occurs before aa (verb prefix marker for perfective). This kind of rule will change **a-aa-reeb-a** to **y-aa-reeb-a** (he has seen), a well formed R-R word.

We developed a set of 34 context-dependent replacement rules for R-R verb. The rules in this category are able to delete, substitute, and insert symbols in the string as long as the context is clearly defined. Each replacement rule $\boldsymbol{RR_i}$ – which corresponds to an infinite *regular relation* (see Kaplan & Kay, 1994) – is compiled into a finite-state transducer, and all resulting rule transducers are in turn composed resulting in one big transducer representing all the rules simultaneously (o denotes composition):

$$\boldsymbol{RR} \ =_{def}\ \boldsymbol{RR_1} \circ \boldsymbol{RR_2} \ldots \circ \ldots \boldsymbol{RR_k}$$

In terms of computational complexity, compiling these kinds of rules is the most expensive step of the whole construction[3]. Compilation needed approx. 1.5 seconds, yielding a finite transducer $\boldsymbol{RR}$ with ≈ 170 states and ≈ 93,000 transitions.

To apply the replacement rules to the strings generated by the grammar, both finite-state machines are composed:

$$\boldsymbol{A_G} \circ \boldsymbol{RR}$$

All the allomorphic changes performed by the combined rule transducer $\boldsymbol{RR}$ manifest themselves on the output tape of $\boldsymbol{A_G} \circ \boldsymbol{RR}$. But these changes have to occur at the surface, input level. We achieve the desired effect by *inverting* the transducer, that is, by switching input and output tape. But before doing so, we have to get rid of the categorical information (introduced in the stem and affix lexicons) still present on both tapes of the transducer. For that purpose, we define a simple unconditional rewriting rule which replaces each category by ε, the empty string, effectively deleting all categories:

$$\texttt{[<Category>]} \ \rightarrow \varepsilon$$

Here `<Category>` is a special meta-symbol, denoting all the grammatical categories defined in the symbol signature.

The transducer for the Runyakitara verb morphology is then defined as follows ($^{-1}$ denotes inversion):

$$(\boldsymbol{A_G} \circ \boldsymbol{RR} \circ (\texttt{[<Category>]} \rightarrow \varepsilon))^{-1}$$

This transducer maps Runyakitara verb forms (incorporating all the allomorphic changes) to sequences of underlying forms interleaved with categorical information about these morphemes (see the next section for example output).

[3] This is due to the various complementation operations for restricting the replacements to the correct contexts (*P-iff-S*-operator, see Kaplan & Kay, 1994).

3.5 Sample Output

The output of the system includes morphemes, their categories and features. Fig. 6 shows some sample output.

```
mukakubaasa:     mu      [VERB_PREF_SPM2P Spm2p=agrmt2p]
                 ka      [VERB_PREF_FPAST Fpast=remotepast]
                 ku      [VERB_PREF_OPM15 Opm15=agrt15]
                 baas    [VERB_ROOT_SIMPLE Simple=simpleverb]
                 a       [VERB_END_IND Ind=mood]

mukakubaaga:     mu      [VERB_PREF_SPM2P Spm2p=agrmt2p]
                 ka      [VERB_PREF_FPAST Fpast=remotepast]
                 ku      [VERB_PREF_OPM15 Opm15=agrt15]
                 baag    [VERB_ROOT_SIMPLE Simple=simpleverb]
                 a       [VERB_END_IND Ind=mood]

zizigyegyenesa:  zi      [VERB_PREF_SPM10 Spm10=agrmt10]
                         [VERB_PREF_PRESENT Present=habitual]
                 zi      [VERB_PREF_OPM10 Spm10=agrt10]
                 gyegyen [VERB_ROOT_SIMPLE1 Simple1=simpleverb1]
                 es      [VERB_EXT_CAUS Caus=true]
                 a       [VERB_END_IND Ind=mood]

zizigyegyenera:  zi      [VERB_PREF_SPM10 Spm10=agrmt10]
                         [VERB_PREF_PRESENT present=habitual]
                 zi      [VERB_PREF_OPM10 Spm10=agrt10]
                 gyegyen [VERB_ROOT_SIMPLE1 Simple1=simpleverb1]
                 er      [VERB_EXT_LOC Loc=prep]
                 a       [VERB_END_IND Ind=mood]

zizigyegyenera:  zi      [VERB_PREF_SPM10 spm10=agrmt10]
                         [VERB_PREF_PRESENT Present=habitual]
                 zi      [VERB_PREF_OPM10 Opm10=agrt10]
                 gyegyen [VERB_ROOT_SIMPLE Simple1=simpleverb1]
                 er      [VERB_EXT_APPL Appl=prep]
                 a       [VERB_END_IND Ind=mood]

zizigyegyenerera:zi      [VERB_PREF_SPM10 Spm10=agrmt10]
                         [VERB_PREF_PRESENT Present=habitual]
                 zi      [VERB_PREF_OPM10 Opm10=agrt10]
                 gyegyen [VERB_ROOT_SIMPLE Simple1=simpleverb1]
                 erer    [VERB_EXT_INT Int=degree]
                 a       [VERB_END_IND Ind=mood]
```

Figure 6. Sample output of RUNYAGRAM.

From the above output, taking the first word as an example, ***mu-ka-ku-baas-a*** (**you managed it;** 'tense starts from last month onwards') has morphemes ***mu-***

serving as subject prefix marker for class two and it is a plural marker serving agreement function; ***ka-*** is a tense marker in remote past**,** ***ku-*** is an object prefix marker for class 15 also serving as agreement, ***baas-*** is a verb root for simple verbs, and ***-a*** is a verb end for indicative mood.

A regular verb in Runyakitara outputs up to fifty thousand (50,000) forms. The *fsm2* script for creating the verb subanalyzer can be found in Appendix B.

4. TESTING

Testing is one of the complex tasks in morphological analyzer development (Beesley and Karttunen, 2003), therefore it needs a lot of care and patience. One of the important aspects of *fsm2* is the testing functionality to aid developers test and debug morphological analyzers. The *fsm2* testing functionality can be described as:

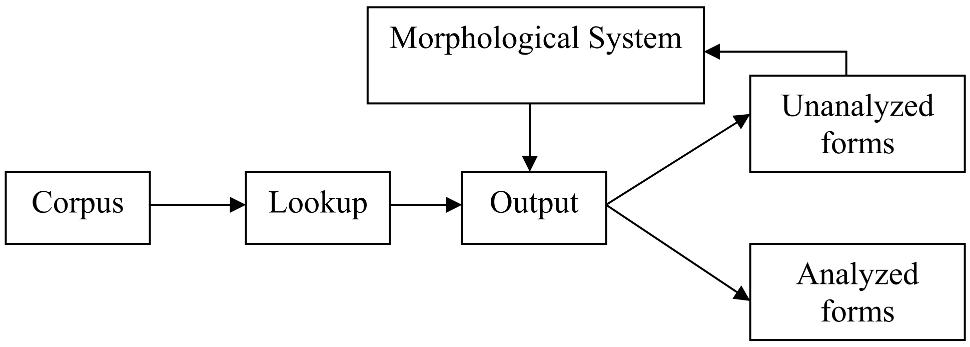

Figure 7. Testing process in *fsm2*.

Applied to Runyakitara, a list of 3971 Runyankore-Rukiga verbs was extracted from the dictionary and a Runyakore orthography reference book (Taylor, 1985). That constituted our testing raw material. Using the lookup operation provided by fsm2, the words were looked up in the analyzer and the results were stored in two files: one with the analyzed forms and another containing the unanalyzed forms. The unanalyzed forms were re-examined for further consideration into the morphological system.

The following table presents results of RUNYAGRAM:

Corpus	3971 tokens	Percentage
Analyzed forms	4604	86%
Non-analyzed forms	559	14%
Precision (correctly analyzed)	3820	82%

Table 3. Testing Results.

The above results indicate that the verb system of R-R in its current development has so far registered success by analyzing 86% of running text.

The precision for the system is at 82%. This is a positive remark on the ability of *fsm2* to analyze verb morphology of R-R.

7. Conclusion and Future Research

This work demonstrates the application of finite state approach in the analysis of Runyakitara verb morphology. Language specific knowledge and insight have been applied to classify and describe the morphological structure of the language, and quasi context-free and rewriting rules have been written to account for grammatical verbs of Runyakitara.

The research results presented above describe the first efforts aimed at building a morphological analyser of Runyakitara, a Bantu language. RUNYAGRAM results from the combination of *Item-and-Arrangement* and *Item-and-Process* models as proposed by Hockett, (1954; 1959). It is evident that the models are applicable to Runyakitara morphology.

Specifically, this work demonstrates:

1. The first computational description of the orthography of the Runyakitara verbs
2. A proof that the *fsm2*-based approach (context-free grammar + rewriting rules) is applicable to a morphologically complex Bantu language like R-R.
3. An enrichment of the common Bantu template to account for the more specific situation in R-R.

Future research: the entire plan for this research is to cater for all Runyakitara word categories to be analyzed by *fsm2*. This will result into a comprehensive morphological analyser of Runyakitara. The morphological analyzer will be an input for many other planned applications such as learning systems and machine translation.

Acknowledgments

The authors would like to acknowledge the continued support and expert advice of Prof. Arvi Hurskainen (Helsinki, Finland) and Prof. John Nerbonne (University of Groningen, The Netherlands). We are especially grateful to an anonymous reviewer for his elaborate and astute comments on an earlier version of this paper. Finally, we would like to thank the editor, Axel Fleisch, for his work.

This paper is a result of the six month visiting research period in Germany that was supported by DAAD (The German Academic Exchange Service). We are grateful.

REFERENCES

Beesley, K.R. & Karttunen, L. 2003.
Finite state morphology. CSLI Publications.

Bernsten, J. 1998.
Runyakitara, Uganda's 'New' Language. **Journal of multilingual and multicultural development** 19(2): 93-107.

Cormen, T.H. & Leiserson, C.E. & Rivest, R.L. & Stein, C. 2001.
Introduction to Algorithms. 2nd Edition. Cambridge, Mass.: MIT Press.

Elwell, R. 2005.
'Finite-state Methods for Bantu Verb Morphology'. In: Nicholas Gaylord, Stephen Hilderbrand, Heeyoung Lyu, Alexis Palmer and Elias Ponvert (eds.), *Texas Linguistics Society 10: Computational Linguistics for Less-Studied Languages*. CSLI Publications.

Guthrie, M. 1967.
An Introduction to the Comparative Linguistics and the Pre-history of Bantu Languages. Gregg International Publishers Ltd.

Hayman, 2007.
'Niger Congo Verb Extension: Overview and Discussion'. In: Doris L. Payne and Jaime Peña (eds.), *Selected Proceedings of the 37th Annual Conference on African Linguistics*, pp. 149-163. Somerville, MA: Cascadilla Proceedings Project.

Hanneforth, T. 2009.
'fsm2 - A Scripting Language for Creating Weighted Finite-state Morphologies'. In: C. Mahlow & Piotrowski (eds.), *State of the Art in Computational Morphology*, pp. 48-63. Heidelberg: Springer Berlin.

Hockett, C.F. 1954.
Two Levels of Grammatical Description. **Word** 10: 210-31.

1958 *A course in Modern Linguistics*. New York, MacMillan.

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

Hurskainen, A. 2004.
Swahili Language Manager: a Storehouse for Developing Multiple Computational Applications. **Nordic Journal of African Studies** 13(3): 363-397.

1992 *A Two-level Computer Formalism for the Analysis of Bantu Morphology: an Application to Swahili*. **Nordic Journal of African Studies** 1(1): 87-119.

1996 'Disambiguation of morphological analysis in Bantu languages'. *Proceedings of COLING-96,* pp. 568-573.

Kaplan R.M. & Kay M. 1994.
Regular Models of Phonological Rule Systems. **Computational Linguistics** 20(3): 331-378.

Karttunen, L. 2003.
‘Computing with Realizational Morphology’. In: Alexander Gulbekh (ed.), *Computational Linguistics and Intelligent Text Processing*, pp. 205-216. Lecture Notes in Computer Science, 2588.

Karttunen, L. & Beesley, K.R. 2005.
‘Twenty-five years of Finite-State Morphology’. In: *Inquiries into Words. A Festschrift for Kimmo Koskenniemi on his 60th Birthday*. CSLI Studies in Computational Linguistics. Stanford CA: CSLI 2005: 71-83.

Katamba, F. 1984.
A Non-linear Analysis of Vowel Harmony in Luganda. **Journal of Linguistics** 20(2): 257-275.

Koskenniemi, K. 1984.
‘A General Computational Model for Word Form Recognition and Production’. *Proceedings of COLING,* pp. 178-181.

Maho, J.F. 2007.
‘A Linear Ordering of TAM/NEG Markers in the Bantu Languages’. *SOAS working papers in linguistics*, Vol. 15, pp. 213-225.

Mohri, M. & Pereira F.C.N. 1996.
Dynamic Compilation of Weighted Context-free Grammars. AT&T Labs – Research.

Morris & Kirwan, 1972.
A Runyakore Grammar. Kampala: East African Literature Bureau.

Muhirwe, J. & Trosterud, T. 2008.
‘Finite State Solutions for Reduplication in Kinyarwanda Language’. *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pp. 73–80, Hyderabad, India, January 2008. Asian Federation of Natural Language Processing.

Roark, B. & Sproat, R. 2007.
Computational Approaches to Morphology and Syntax. Oxford University Press.

Nurse, D. & Philippson, G. 2003.
The Bantu languages. Routledge Language Series.

Pretorius, L. & Bosch, E.S. 2003.
Finite-State Computational Morphology: An analyzer prototype for Zulu. **Machine Translation** 18(3): 195-216.

Pretorious, R., Berg, A. & Pretorious, L. 2009.
‘Setswana Tokenisation and Computational Verb Morphology: Facing the Challenge of a Disjunctive Orthography’. *Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages* – AfLaT 2009, pages 66–73, Athens, Greece, 31 March 2009.

Spencer, A. 1991.
Morphological Theory. Wiley-Blackwell publishers.

Taylor, C. 1985.
Descriptive Grammars, Nkore-Kiga. London, Groom Helm.

Wintner, S. 2008.
Strengths and Weaknesses of Finite-state Technology: A Case Study in Morphological Grammar Development. **Natural Language Engineering** 14(4) (October 2008): 457-469.

Yona, S. & Wintner, S. 2005.
A Finite State morphological grammar for Hebrew. *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pp. 9-16.

About the authors: *Fridah Katushemererwe* is an Assistant Lecturer at the Institute of Languages, Makerere University and a PhD student at the Faculty of Computing and Information Technology, Makerere University. Currently researching in the area of Intelligent Computer Assisted Language Learning Systems (ICALLs) specifically, utilizing natural language processing techniques to develop an ICALL for Runyakitara. *Dr. Thomas Hanneforth* is an Associate professor for computational linguistics at the Linguistics Dept. at Potsdam University.

APPENDIX A – DETAILED DESCRIPTION OF RUNYAKITARA MORPHOLOGY

Slot	**Meaning**	**morpheme**	**Word formed**	**Gloss**
0	Verb root	Vroot	*gyend*-a	go
1	Verb extensions (VEXT)	Ca – causative (es) Apl – applicative (er) Rec – reciprocal (an) Pas – passive (w) Int – intensive (erer) Stat – stative (ek) Rev - reversive	gyend-*es*-a gyend-*er*-a gyend-*an*-a reeb-*w*-a gyend-*erer*-a gyend-*ek*-a teek-*uur*-a *Also possible:* gyend-es-ebw-a gyend-an-is-a gyend-an-is-ibw-a	make to go go for go with be seen go specifically for - remove (on stack)
2	Verb end (VE)	Ind – indicative (a) Subj – subjunctive (e) Past – past tense (ire)	y-aa-gyend-*a* n-gyend-*e* n-gyenz-*ire*	he has gone may I go I went
3	Post final	Pf1 – adverbial (ho, yo, mu) Pf2 – mitigator (ga)	gyend-a-*yo* ti-n-ka-gyend-a-*ga*	go there I have never gone
4	Aspect marker	Asp – reflexive (e)	ku-*e*-reeb-a	to see oneself
5	Object pronouns	Op1 – object pronouns (18) Op2 – object pronouns (18)	*ba*-gyend-e mu-*mu*-*n*-reeb-er-e	Let them go You see him for me
6	Tense/aspect markers	Inf – infinitive (ku) Hab – habitual (ø) Pf – perfective (aa) Ff – far future (ria/rya) Rp – remote past (ka)	*ku*-gyend-a n-gyend-a n-*aa*-gyend-a n-*dya*-gyend-a n-*ka*-gyend-a	to go I go (everyday) I have gone I will go (far future) I went (last year)
7	Negation marker	Neg2 – negative (ta)	ku-*ta*-gyend-a	not to go
8	Subject pronouns	Sp – subject pronouns (18)	*n*-aa-gyenda *tw*-a-gyend-a	I have gone we have gone
9	Aspect marker	Asp – progressive (ni)	*ni*-ba-gyenda	they are going (now)
10	Negation marker	Neg1 – negative 1 (ti)	*ti*-baa-gyend-a	they have not gone

APPENDIX B – *FSM2* SCRIPT FOR CREATING THE VERBAL ANALYZER

```
# Define a macro mapping verb equivalence class symbols (%SYMBOL%)
# to sublexicons stored in a file called %SYMBOL%.lex.
macro verb_substitution(%SYMBOL%)
      load lexicon %SYMBOL%.lex
      optimize
      map %SYMBOL%
endmacro

# Load symbol signature
load symspec ../../symbols/rr.sym

# Load all verb roots stored in a number of files and associate them
# with a symbol denoting the verb's equivalence class (VERBFORMx).
# This creates a substitution map associating each verb class with
# the verb roots in this class

call verb_substitution(VERBFORM1)
call verb_substitution(VERBFORM2)
call verb_substitution(VERBFORM3)
call verb_substitution(VERBFORM4)
call verb_substitution(VERBFORM5)
call verb_substitution(VERBFORM6)
call verb_substitution(VERBFORM7)
call verb_substitution(VERBFORM8)

# Compile the verb subgrammar and optimize it
load grammar verbs
optimize

# Perform the verb root substitution and optimize the result
substitute
optimize

# Compile the rewriting rules
# and compose them with the result of the step before
load contextrules verbs.rules
compose

# Delete all the category information from the lower tape
regex "[<category>] --> []"
compose

# Finally, swap input and output tape of the transducer
invert
optimize
```