Finite State Solutions For Reduplication In Kinyarwanda Language

Jackson Muhirwe
Makerere University
Uganda jmuhirwe@cit.mak.ac.ug

Trond Trosterud
University of Troms
trond.trosterud@hum.uit.no

Abstract

Reduplication, the remaining problem in computational morphology is a morphological process that involves copying the base form wholly or partially. Reduplication can also be classified as either bounded or unbounded reduplication. Some solutions have been proposed for bounded reduplication. Some of the proposed solutions use ordered replace rules while others use simultaneous two-level rules. In our attempt to solve both bounded and unbounded reduplication we used a combination of two-level rules and replace rules. All our experiments were are carried out on Kinyarwanda an under-resourced language with complex agglutinative morphology.

1 Introduction

Reduplication is known to many computational morphologists as the remaining problem. Unlike concatenative morphology, which involves concatenation of different components to create a word, reduplication involves copying. Reduplication is therefore non-concatenative, and involves copying of either the whole word or part of the word. The reduplicated part of the word could be a prefix or part of the stem or even a suffix. This copying is what makes reduplication an outstanding problem. Depending on the language, reduplication may be used to show plurality, iterativity, intensification or completeness (Kimenyi, 2004). Some of the notable examples of reduplication in computational morphology that have been reported include Kinande, Latin, Bambara (Roark and Sproat, 2007); Tagalog and Malay (Beesley and Karttunen, 2003; Antworth, 1990). In these cases, one language may be exhibiting full stem reduplication while another may be exhibiting partial stem reduplication (Syllable).

Reduplication may generally be divided into two: bounded and unbounded. Bounded reduplication is the kind that involves just repeating a given part of the word. Unbounded reduplication differs from bounded reduplication in that bounded reduplication involves copying of a fixed number of morphemes. Unbounded reduplication is considerably more challenging to deal with compared with bounded reduplication. Unbounded reduplication has received little attention from researchers no wonder it is yet to be fully solved (Roark and Sproat, 2007). In principle, finite state methods are capable of handling bounded reduplication, and here some solutions have been proposed. In this paper we present our attempt to solve both bounded and unbounded reduplication in Kinyarwanda a typical Bantu language. Kinyarwanda is the national and official language of Rwanda. It is closely related to Kirundi the national language of Burundi. It is the mother tongue of about 20 million people living in the great lakes region of East and Central Africa. Kinyarwanda is a less privileged language characterised by lack of electronic resources and insignificant presence on the Internet. The language has an official orthography where tones, long vowels and consonants are not marked. Kinyarwanda is agglutinative in nature, with complex, mainly prefixing morphology. Verb forms may have slots of up to 20 affixes to be attached to the root on both sides: left and right. Reduplication is a common feature and generally all verbs undergo some form of reduplication. Adjectives and adverbs tend to undergo full word reduplication, as we shall see in section 2.

2 Kinyarwanda Reduplication

Kinyarwanda exhibits full word reduplication, full stem reduplication and partial stem reduplication or syllable reduplication. Full word reduplication involves copying of the whole word, this phenomenon has been observed mainly in adjectives and adverbs. Full stem reduplication involves copying a full stem of either a verb or a noun. Part of a stem is copied in partial stem reduplication. To a large extent this copying is uniform (the large number of example given below show that) but there are also cases of un uniformity. There are cases when a nasal (n or m) and an associative morpheme is inserted between the copied morpheme and its base form. Kinyarwanda language exhibits also cases of suffix reduplication attested mainly in verb extensions which are not considered in this paper.

For our discussion in this section we shall look at full word reduplication full stem reduplication and partial stem reduplication will be considered last.

For readers with an orientation towards theoretical linguistics we shall categorise our examples according to whether they are lexical or grammatical, but for implementation purposes this will not be considered. Lexical reduplication is concerned with words that may appear in the dictionary. Reduplicated words may appear in a dictionary as distinct words from the original word which underwent reduplication. Grammatical reduplication is concerned with words or sentences that are reduplicated based on grammatical rules. For instance, only monosyllabic verbs are reduplicated, bisyllabic and polysyllabic are never reduplicated (Kimenyi, 2004).

2.1 Full Word Reduplication

All adjectives, adverbs and numerals may undergo full word reduplication. In this case, the complete word is copied to form a new word.

Adjectives

munini ”big” > *munini**munini*** ”big”
muto ”small” > *muto**muto*** ””small /young”
mashya ”new” > *mashya**mashya*** ”very new”

Adverbs

vuba ”fast” > *vuba**vuba*** ”very fast”
buhoro ”slowly” > *buhoro**buhoro*** ”very slowly”
buke ”little” > *buke**buke*** ”very little”

Numerals

rimwe ‘’one’ *rimwe**rimwe*** ’one by one or once in a while’
kabiri ’two’ *kabiri**kabiri*** ’two by two’
gatatu ’three’ *gatatu**gatatu*** ’three by three’

2.2 Full Stem Reduplication

This involves reduplication of the whole stem resulting in a new word with a different meaning from its parent. This kind of reduplication has been observed in both verbs and nouns and can be both at lexical and grammatical level. Formally, it differs from word reduplication in that the verb and noun class prefix does not participate in the reduplication, whereas word reduplication reduplicates the class prefix as well, cf. *ka-biri**ka-biri*** vs. *gu-taga=**taga***.

Verbs differ from nouns in that all verbs may be reduplicated. In many cases, the resulting reduplicated verb keeps the same basic meaning, but adds iterativity , continuity, etc. In other cases, the result is a change in meaning. For nouns, the situation is different. Here, reduplication is semantically restricted to meaning ”kind of”, ”associated to”, and only a subset of the nouns undergo reduplication.

In our transducer, we open for reduplication for all verbs, whereas reduplicating nouns are singled out as a separate group in the lexicon.

In all the verb cases we see iterativity, continuity of events or an activity done many times. In the noun examples it may be noticed that reduplication refers to the description of an object, to what an object does, or to an association based upon the original meaning.

2.2.1 Grammatical Reduplication

The examples given below mainly concern lexical reduplication. Grammatical reduplication involves reduplication of existing word forms, thereby forming new words with different meanings. Grammatical reduplication may be realized at word level or at sentence level. Here we shall consider reduplication at word level only; sentence level processes are outside the scope of a morphological transducer. The reader is advised to consult Kimenyi (1986) for sentence level reduplication.

Also in this category it is the whole stem that is reduplicated. Most of the examples belonging to this category are of verb reduplication.

Examples include the following:
kugenda ”to walk” $>$ *kugenda**genda*** ”to walk around”
kubunda ”to bend” $>$ *kubunda**bunda*** ”to walk bending”
kubumba ”to mould” $>$ *kubumba**bumba*** ”to continue moulding”
guhonda ”to knock” $>$ *guhonda**honda*** ”to knock repeatedly”

Notice from the examples above that this type of reduplication is limited to two-syllable stems, and most of these verbs end with a nasal cluster NC. Two syllable verbs referring to continuous events are never grammatically reduplicated, e.g gukunda ”to love”, kwanga ”to hate” guhinga ”to cultivate”. They may undergo lexical reduplication, though. So, in an analysis invoking semantic disambiguation, trisyllabic reduplicated verbs will be discarded as candidates for grammatical reduplications.

2.3 Partial Stem Reduplication

In this case the initial reduplicated syllable has the form CV, VC or CVN.

Verbs
*kuje**je**ta* ”to drop /leak”
*guse**se**ra* ”to go through a fence with a bent back”
*kuba**ba**ra* ”to fill pain”
*kunyu**nyu**za* ”to suck”

Nouns
*ise**se**me* ”nausea”
*inge**ge**ra* ”crook”
*umuru**ru**mba* ”greed”
*ibijo**jo**ba* ”rain drops”

2.4 Unbounded Reduplication

This is still a challenge, and it involves two cases in Kinyarwanda, nasal insertion and the insertion of the associative between the reduplicates.

2.4.1 Nasal Insertion

These cases may be few but they do exist. The majority of the cases are verbs. Few nouns exhibit this kind of behaviour.

Verbs
*guto**n**toma* ”to make pig’s noise”
*kuvu**m**vura* ”to talk (insulting)”
*guta**n**tamura* ”to tear up”

Nouns
*igipa**m**para* ”a useless thing”

2.4.2 Associative insertion

Associative insertion has mainly been observed in demonstratives when they reduplicate. An associative infix such as *na* ”and” and *nga* ”such and such” is inserted between the reduplicates.

Demonstratives
*uyu**ng**uyu* ”this one”, abangaba ”these ones”
*aha**ng**aha* ”Here”, *aha**na**ha* ”such and such a place”
*iki**ni**ki* ”this and this one”.

3 The proposed approach

In order to handle the different issues presented above we used a hybrid approach. The hybrid approach is a combination of two-level rules and replace rules. These two formalisms represent the state of the art and practice in computational morphology. The two formalisms are powerful, well designed and well understood.

3.1 Two-level Formalism

The two-level formalism has been the dominant formalism in Computational Morphology since its invention by Koskenniemi in 1983 (Koskenniemi, 1983). Since then the approach has been used to develop morphological analysers for very many languages around the world, including the Bantu language Swahili (Hurskainen, 1992). This formalism has been the major motivation force behind renewed interests in computational morphology since 1983. The two-level formalism is based on two-level rules which are applied to a lexicon to facilitate lexical to surface level mappings. The two-level rules are compiled either by hand (Antworth, 1990) or by machine (Karttunen, 1992) into finite state networks.

The rule network may now be applied to a lexicon that has been compiled into a finite state network. A two-level based morphological analyser is developed by composing the two-level rule network with the lower side of the finite state lexicon network. The two-level rules are symbol to symbol rules which apply to the lexicon in parallel. The developer does not have to worry about the order of the rules. The only problem is that rules tend to conflict. With computerised compilers, such conflicts are no longer a problem. The compiler shows which rules are conflicting, so that the developer can resolve them. The output from a two-level morphological analyser is never affected by the order of the rules.

Two-level rules are generally of the form
CP OP LC _ RC
where CP = Correspondence Part; OP = Operator; LC = Left Context; RC = Right Context

There are four different kinds of rules that may be used to describe morphological alternations of any language.

1. a:b => LC _ RC. This rule states that lexical //a// can be realized as surface b ONLY in the given context. This rule is a context restriction rule

2. a:b <= LC _ RC This rule states that lexical //a// has to be realized as surface b ALWAYS in the given context. This rule is a surface coercion rule.

3. a:b <=> LC _ RC this is a composite rule which states that lexical //a// is realized as surface be ALWAYS and ONLY in the given context.

4. a:b /<= LC _ RC This is an exclusion rule that states that lexical //a// is never realized as surface //b// in the given context.

These rules may be compiled into finite state acceptors either by hand or automatically using one of the available Two-level rule compilers. For the purpose of this research we used the Xerox Finite State Tools.

3.2 Replace Rules

On the other hand the replace rules were introduced by Karttunen in 1995 motivated by the rewrite rules model developed by Kay and Kaplan (1994). Replace rules were easily accepted by computational linguistics because that is how linguistics has been done every where. It was so natural for linguistics to take up this formalism.

The replace rules are regular expressions that make it possible to map the lexical level strings to surface level strings. Replace rules have been very popular in Computational Morphology and have been used to develop many morphological analysers.

Replace rules are compiled into a finite state network and this network is applied to the lower side of the lexicon network to map the lower level strings to the surface level strings. It is worthy noting that replace rules are feeding rules and therefore apply in a cascade. Each rule uses the result of the preceding rule. Because of this, a linguist writing language grammar using replace rules notation must order rules in a proper way, otherwise the results may not be right. For implementation purposes, replace rules have one clear advantage over two-level rules. They can map symbols to symbols; symbols to strings; strings to symbols; and strings to strings. Replace rules are very handy when it comes to writing string to string mappings. In this case you write only one rule instead of the many rules you would otherwise have to write while using two-level rules. Replace rules take the following four forms:

Unconditional replacement

```
A -> B
```

Unconditional parallel replacement (Several rules with no contexts)

```
A1 -> B1, A2 -> B2, ...... An -> Bn
```

Conditional replacement. (One rule with contexts)

```
UPPER -> LOWER || LEFT _ RIGHT
```

Conditional parallel replacement.

```
UPPER1 -> LOWER1 ||
LEFT1 _ RIGHT1 ,,  UPPER2 -> LOWER2 ||
LEFT2 _ RIGHT2,,..,,UPPERn -> LOWERn ||
LEFTn _ RIGHTn
```

3.3 Comparison of the two Formalisms

- Replace rules are organised vertically in a cascade and feed each other. Two-level rules, on the other hand side, are organised horizontally and apply in parallel.
- Because replace rules are feeding rules, they must be properly ordered. Order is not important in two-level rules and would not affect the output.
- Replace rules conceptually produce many intermediate levels when mapping from lexical to surface level.
- Since two-level rules apply simultaneously, there is no ordering problem. The only problem that arises are conflicts that the linguist must deal with. But as we said earlier, this is no longer a problem since current two-level compilers can detect the rule conflicts and then the grammar writer can deal with them accordingly.

3.4 Towards a Hybrid Approach

As much as we have seen that these two formalisms have differences, they all work very well and are efficient at doing what they were designed to do. Networks compiled from these two networks have the same mathematical properties (Karttunen and Beesley, 2005), and none of the formalisms can be claimed to be superior over the other, per se. It is further claimed that choosing between two-level rules and replace rules is just a matter of personal choice. This is true as far the general areas of application of each of these rules are concerned. Our experience has shown that two-level rules are much easier to learn and conceive how they work. This experience is also shared by Trosterud and Uibo who also while working on Sami found it much easier to learn two-level rules but again proposed that it would be possible to combine both formalisms (Trosterud and Uibo, 2005). Independently, Muhirwe and Barya (2007) also found it easier to learn two-level rules and they used them to develop their Kinyarwanda Noun morphological analyser. Beesley and Karttunen also realised that each one of these rules has strong points and weak points. There are incidences where it is much easier to use two-level rules and there are other incidences where it is easier to use replace rules over two-level rules (Beesley and Karttunen, 2003). Let us look at an example to strengthen our argument. In solving limited partial stem reduplication in Tagalog, Antworth used two level rules to model the solution. This same example was repeated by Beesley and Karttunen (2003). Efforts to rewrite the solution using replace rules resulted in many rules. We used this approach to solve the problem of partial reduplication in Kinyarwanda.

```
Alphabet %+:0 b c d f g h j k l m n
p q r s t v x y z a e i o u;
Sets
C = b c d f g h j k l m n p q r s t
 v x y z;
V = a i e o u  ;

Rules
"R for realisation as Consonant"
R:CC <=> _ E: %+: CC;
where CC in C;

"E realisation as vowel"
E:VV <=> _ %+: (C:) VV;
where VV in V;
```

Replace rules have an edge over two-level rules when it comes to string to string mapping. When the strings are of unknown length, two-level rules cannot be applied, and we will have to use special compilation routines from the xfst toolbox. In other words, replace rules are more appropriate if the mapping requires replacement of a string, whereas two-level rules are more appropriate when only symbols are involved, and especially when sets of symbols are involved. Based on this we decided to combine the two approaches to take advantage of each formalism's strength.

4 Implementation

At the onset, we wanted to solve three problems: Full wordform reduplication (we will follow established practice and refer to it as word reduplication), stem reduplication and first syllable or partial stem reduplication. Our hybrid approach was used as follows. We used the two-level rules to solve the problem they are best at solving: partial stem reduplications. Beesley and Karttunen's compile-replace algorithm was then used to handle full word and full stem reduplication.

4.1 Full word and full stem Reduplication

The full word and full stem reduplication was handled by use of the replace rules and the compile-replace algorithm. The compile-replace algorithm is based on the insight that any string S can be reduplicated using regular expressions of the form {S}ˆ2. The central idea behind the application of the compile-replace algorithm therefore is looking for a way to enclose the stem with the delimiters { and }ˆ2. This was done by enclosing the whole stem with ˆ[{S}ˆ2ˆ] in the lexicon, and given a reduplication context, the compile-replace algorithm is applied to the lower side of the lexicon network, doubling the stem. When the reduplication context is not present, the delimiters were simply deleted. As an example, take a look at part of thelexc lexicon below:

```
LEXICON Root
0:ˆ[{    AdjRoots;
0:ˆ[{    AdvRoots;
```

This continues to the adverb and adjective or to any other sublexicon

```
LEXICON AdjRoots
kinini    AdjSuff;
kito      AdjSuff;
muto      AdjSuff;
```

Lastly we can add the suffix

```
LEXICON    AdjSuff
+Adjective:0    Redupli;

LEXICON    Redupli
+Reduplic:}ˆ2ˆ]    #;
%+unmarked:0       #;
```

After compiling the lexicon and applying the compile-replace algorithm to the lower side, the alternation rules can then be applied to constrain the surface realisation of the reduplicated words. In this case most of the surface alternation rules were written using replace rules formalism.

4.2 Partial stem reduplication

The solution provided by Antworth in PC Kimmo is a good solution to handling limited length reduplication. We therefore adapted this solution to provide a solution to first syllable reduplication in Kinyarwanda. The rules we used were presented in the previous section. We used the two-level rules because of their convenience, but, as noted, one will get the same result by using replace rules. These two-level rules were compiled into a finite state network and then intersected using the two-level compiler *twolc*. The rule network was then applied to the lower side of the lexicon network to produce the required output on the surface. In the lexicon we had to include a feature that would interact with the rules to cause reduplication:

```
Lexicon PSPrefix
[Redupli]:RE+  PVRoot;
Lexicon PVRoot
jeta    VFinal;
```

In Kinyarwanda, the partial stem reduplication is of three types, CV, VC and CVN reduplication. We thus made three different templates, all modeled upon the rule shown here.

4.3 Emerging Problems in Kinyarwanda reduplication

The solution provided above for partial reduplication seemed to work very well until we tested the results, and then we found that there were some interesting challenges.

1. some stems reduplicate and cause insertion of a nasal. For example /gu + kama/ > /gukankama/ /gu + toma/ > / gutontoma/

2. there were cases of complex consonants which when present makes the reduplication problem harder. Evan Antworth's solution was for fixed length CV reduplicates and it is in this case rendered inefficient (Antworth, 1990). Examples /gucyocyora/
/kunyunyuza/
/gushwashwanya/

3. when demonstratives reduplication, a presentative affix /nga/ is inserted in the middle of the reduplicates

In order to solve the first challenge, we carried out more negative tests and looked for cases of words that were not recognized. Of these we identified reduplicates where a nasal is inserted and we found that such cases are not very frequent. The majority of verbs and nouns undergo full stem reduplication, for which the provided solution was adequate. The remaining few undergo partial stem or first syllable reduplication. There are also cases of stems that undergo both full stem and partial stem reduplication, but these were not a challenge at all. So our solution to the nasal insertion challenge was to write a rule that would insert a nasal between the reduplicating prefix and the base stem.

```
[] -> n || _ [ [t o m a] |
[k a m a] | ....| [v u r a]]
```

The second problem involving complex consonants was solved by representing each complex by a multicharacter symbol that is not used in the lexicon. For example, in /kunyunyuza/ there is a complex consonant ny which is part of the reduplicate. We represent all occurrences of ny with N and the following rule will be applied lastly to effect the surface realisation.

```
N -> ny
```

The third problem was solved by using replace rules. The problem of reduplication of demonstratives was partly solved by application of the compile-replace algorithm and replace rules. We used a replace rule to insert /nga/ in all the reduplicated demonstratives.

```
[] -> [ n g a ] || _ demo .#.
```

4.4 Evaluation and tests

The partial, full stem and full word reduplication lexica were compiled and composed together in a finite state network. We applied the network of all the rules described above for all the different issues to the lower side of the lexicon network. We then carried out tests for both analysis and generation. We did both negative testing and positive testing. Positive involved testing the system on the words that were part of the lexicon. These we found were all correctly analysed. Below are some of the results:

```
apply up> ikigorigori
iki[CL7-SG]gori[stem_redupli][noun]
apply up> kugendagenda
ku[Inf]genda[stem_redupli][verb]
Demonstratives with nga insertion
aka[DEM-12][dem_redupli][Demonst]
akongako
ako[DEM-12][dem_redupli][Demonst]
Nasal insertion
gutontoma
ku[Inf][Redupli]toma[verb]
Complex consonants
kunyunyuza
ku[Inf][Redupli]Nuza[verb]
Full stem reduplication
gusomasoma
ku[Inf]soma[stem_redupli][verb]
Full word reduplication
muninimunini
munini+Adjective+Reduplic
```

Negative testing involved selecting words from our untagged corpus of Kinyarwanda. Since these words were not part of the lexicon, they were not recognized and were then duly added to the lexicon. Adding a new word to the lexicon is very easy since it only involves identifying the reduplicating part of the word and it is then added to the appropriate sub-lexicon. This testing will be continued as we discover new reduplicated words.

The tests indicated above were manual tests. We created another test set to be carried out automatically. In this case we created a test file with about 100 known reduplicated forms of different word categories in Kinyarwanda. The results indicated that the earlier problems due to unbounded reduplication: complex consonants, insertion of nasals and the prefix /nga/ have now been fully solved.

5 Conclusion

The solutions provided in this paper have demonstrated that existing extended finite state methods are sufficient to handle all forms of reduplication in Kinyarwanda. The hybrid approach proposed in this paper makes it easy to handle all forms of reduplication problems attested in Kinyarwanda language. This approach could also be used with other problems in morphological analysis. The finite state developer can solve morphological problems using the most appropriate approach depending on whether what is being replaced is a symbol or a string.

References

Antworth, E.,L. 1990. *PC-KIMMO: a Two-level Processor for Morphological Analysis*. No. 16 in Occasional Publications in academic computing. Dallas: Summer Institute of Linguistics.

Beesley, K. AND Karttunen L. 2003. *Finite State Morphology: CSLI Studies in Computational Linguistics*. Stanford University, CA: CSLI Publications.

Guthrie, M. 1971. *Comparative Bantu*. Vol. I-IV. Farnborough: Gregg International.

Hurskainen, A. 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 (1992)

Karttunen, L., Kaplan, R., and Zaenen, A., (1992). *Two-level morphology with composition.* Xerox Palo Alto Research Center - Center for the Study of Language and Information. Stanford University.

Karttunen, L. 1995. *The replace Operator*. Proceedings of ACL-95, pp 16-23, Boston Massachusetts.

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.

Kay, M., Kaplan, R. 1994. Regular Models of Phonological rule systems Computational Linguistics, Special issue on Computational phonology, pg 331-378

Kimenyi, A. 1986. *Syntax and semantics of reduplication: A semiotic account La Linguistique* Vol 22 Fasc 2/1986

Kimenyi, A. 2004. *Kinyarwanda morphology* In the International Handbook for inflection and word formation vol2.

Koskenniemi, K. 1983. *Two-level morphology: a general computational model for word-form recognition and production*. Publication No. 11. University of Helsinki: Department of General Linguistics.

Muhirwe, J. and V. Baryamureeba 2007. Towards Computational Morphological Analysis for Kinyarwanda. Proceedings of the 1st International conference on Computer science and informatics,Feb 2007, Nairobi, Kenya.

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

Trosterud, T and Uibo, H. 2005. *Consonant gradation in Estonian and Smi: two-level solution* In: Inquiries into Words, Constraints and Contexts. Festschrift in the Honour of Kimmo Koskenniemi 60th anniversary. CSLI Publications 2005.